УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

ЗБОРНИК РАДОВА
ФАКУЛТЕТА ТЕХНИЧКИХ НАУКА

Едиција: Техничке науке - зборници
Година: XXXIV
Број: 1/2019

Нови Сад

Едиција: „Техничке науке – Зборници“
Година: XXXIV *Свеска*: 1

Издавач: Факултет техничких наука Нови Сад
Главни и одговорни уредник: проф. др Раде Дорословачки, декан Факултета техничких Наука у Новом Саду

Штампа: ФТН – Графички центар ГРИД, Трг Доситеја Обрадовића 6, Нови Сад

CIP-Каталогизација у публикацији
Библиотека Матице српске, Нови Сад

378.9(497.113)(082)
62

ЗБОРНИК радова Факултета техничких наука / главни и одговорни уредник Раде Дорословачки. – Год. 7, бр. 9 (1974)-1990/1991, бр.21/22 ; Год. 23, бр 1 (2008)-. – Нови Сад : Факултет техничких наука, 1974-1991; 2008-. – илустр. ; 30 цм. –(Едиција: Техничке науке – зборници)

Месечно

ISSN 0350-428X

COBISS.SR-ID 58627591

ПРЕДГОВОР

Поштовани читаоци,

Пред вама је прва овогодишња свеска часописа „Зборник радова Факултета техничких наука“.

Часопис је покренут давне 1960. године, одмах по оснивању Машинског факултета у Новом Саду, као „Зборник радова Машинског факултета“, а први број је одштампан 1965. године. Након осам публикованих бројева у шест година, пратећи прерастање Машинског факултета у Факултет техничких наука, часопис мења назив у „Зборник радова Факултета техничких наука“ и 1974. године излази као број 9 (VII година). У том периоду у часопису се објављују научни и стручни радови, резултати истраживања професора, сарадника и студената ФТН-а, али и аутора ван ФТН-а, тако да часопис постаје значајно место презентације најновијих научних резултата и достигнућа. Од броја 17 (1986. год.), часопис почиње да излази искључиво на енглеском језику и добија поднаслов «Publications of the School of Engineering». Једна од последица нарастања материјалних проблема и несрећних догађаја на нашим просторима јесте и привремени прекид континуитета објављивања часописа двобројем/двогодишњаком 21/22, 1990/1991. год.

Друштво у коме живимо базирано је на знању. Оно претпоставља реорганизацију наставног процеса и увођење читавог низа нових струка, као и квалитетну организацију научног рада. Значајне промене у структури високог образовања, везане за имплементацију Болоњске декларације, усвајање нове и активне улоге студената у процесу образовања и њихово све шире укључивање у стручне и истраживачке пројекте, као и покретање нових мастер и докторских студија, доносе потребу да ови, веома значајни и вредни резултати, постану доступни академској и широј јавности. Оживљавање „Зборника радова Факултета техничких наука“, као јединственог форума за презентацију научних и стручних достигнућа, пре свега студената, обезбеђује услове за доступност ових резултата.

Због тога је Наставно-научно веће ФТН-а одлучило да, од новембра 2008. год. у облику пилот пројекта, а од фебруара 2009. год. као сталну активност, уведе презентацију најважнијих резултата свих мастер радова студената ФТН-а у облику кратког рада у „Зборнику радова Факултета техничких наука“.

Поред студената мастер студија, часопис је отворен и за студенте докторских студија, као и за прилоге аутора са ФТН или ван ФТН-а.

Зборник излази у два облика – електронском на web сајту ФТН-а (www.ftn.uns.ac.rs) и штампаном, који је пред вама. Обе верзије публикују се сваки месец, у оквиру промоције дипломираних мастера. Електронска верзија Зборника од почетка 2019. године излази у новом облику, са директним приступом сваком раду, са јединственим DOI (Digital Object Identifier) бројевима. На тај начин побољшавамо видљивост радова и повећавамо квалитет часописа у целини.

У овом броју штампани су радови студената мастер студија, сада већ мастера, који су радове бранили у периоду од 06.09.2018. до 05.10.2018. год., а који се промовишу 27.01.2019. год. То су оригинални прилози студената са главним резултатима њихових мастер радова.

Известан број мастер кандидата објавили су радове на некој од домаћих научних конференција или у неком од часописа. Њихови радови нису штампани у Зборнику радова.

Велик број дипломираних инжењера–мастера у овом периоду био је разлог што су радови поводом ове промоције подељени у две свеске.

У овој свесци, са редним бројем 1., објављени су радови из области:

- машинства,
- електротехнике и рачунарства,
- грађевинарства и
- саобраћаја.

У свесци са редним бројем 2. објављени су радови из области:

- графичког инжењерства и дизајна,
- архитектуре,
- инжењерског менаџмента,
- инжењерства заштите на раду и заштите животне средине,
- мехатронике,
- математике у техници,
- геодезије и геоматике и
- управљања ризиком од катастрофалних појава и пожара.

Уредништво се нада да ће и професори и сарадници ФТН-а и других институција наћи интерес да публикују своје резултате истраживања у облику регуларних радова у овом часопису. Ти радови ће бити објављивани на енглеском језику због пуне међународне видљивости и проходности презентованих резултата.

У плану је да часопис, својим редовним изласком и високим квалитетом, привуче пажњу и постане довољно препознатљив и цитиран да може да стане раме-уз-раме са водећим часописима и заслужи своје место на СЦИ листи, чиме ће значајно допринети да се оствари мото Факултета техничких наука:

„Високо место у друштву најбољих“

Уредништво

SADRŽAJ

STRANA

Radovi iz oblasti: Mašinstvo

Radovi iz oblasti: Elektrotehnika i računarstvo

Radovi iz oblasti: Građevinarstvo

Radovi iz oblasti: Saobraćaj

Zbornik radova Fakulteta tehničkih nauka Novi Sad

UDK: 621.791
DOI: https://doi.org/10.24867/01AM01Madzar

OPTIMIZACIJA ELEKTROLUČNOG ZAVARIVANJA TOPLJIVOM ELEKTRODOM U ZAŠTITNOM GASU PRI ZAVARIVANJU MPM SENDVIČ LIMOVA

OPTIMIZATION OF GAS METAL ARC WELDING DURING WELDING OF MPM SANDWICH SHEET METAL

Milko Madžar, Sebastian Baloš, *Fakultet tehničkih nauka, Novi Sad*

Oblast - MAŠINSTVO

Kratak sadržaj – *Pojavom sve strožijih zahteva za smanjenje buke i vibracija, pojavljuju se i novi materijali koji mogu da ispune te zahteve. Jedan od tih materijala jesu metal – polimer – metal (MPM) sendvič paneli korišćeni kao antiakustična barijera, za kućišta indstrijskih generatora i elektromotora. Zavarivanje ovih panela je zbog prisustva polimernog sloja problematično i može da rezultuje deformacijama, odnosno razdvajanjem čeličnih ploča. U ovom radu, prikazana je optimizacija tehnologije zavarivanja, sa ciljem da se na što ekonomičniji način, izvrši zavarivanje, uz što manje isparavanje polimernog međusloja.*

Ključne reči: *MPM lim, zavarivanje, MAG postupak.*

Abstract – *With increase of demands for lowering noise and vibration levels, there are many new materials developed that can satisfy these demands. One of these materials is metal – polimer – metal (MPM) sandwich panel used as anti-aqoustic element in industrial generators and electric motor housings. Problems related to welding refer to the polymer interlayer, that is, buckling and separating of steel plates. In this paper, welding technology optimization is shown, so that the joining process is as economical and with as low evaporation of polymer interlayer as possible.*

Keywords: *MPM plate, welding, MAG procedure.*

1. UVOD

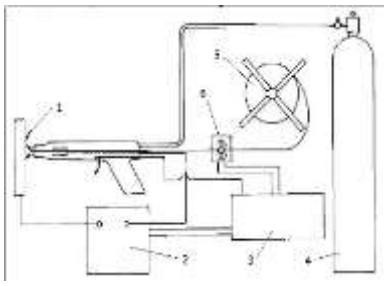
Zavarivanje predstavlja proces u kome se dva materijala spajaju u neraskidivu vezu. Ovaj proces može se ostvariti sa dodatkom dodatnog materijala ili bez dodatnog materijala. Zavareni spoj mora ispuniti definisane tehnološke zahteve, mehaničke osobine, a u pojedinim slučajevima i da poseduje određene estetske zahteve [1].

MPM limovi se koriste kao kućišta velikih industrijskih elektromotora i električnih generatora, delova panela za vozila, za poklopce motora i kućišta menjača, itd. Njihova osnovna prednost u odnosu na jednoslojne čelične limove jeste sposobnost smanjenja buke i upijanja vibracija. Najčeći postupci zavarivanja MPM limova su MIG/MAG i elektrootporni postupak [2].

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Sebastian Baloš.

1.2 MIG/MAG postupak zavarivanja

Elektrolučno zavarivanje topljivom elektrodom u zaštitnom gasu (slika 1) je proces spajanja osnovnog i dodatnog materijala topljenjem pri čemu se zagrevanje vrši električnim lukom, dodatni materijal je u koturu i može se primenjivati inertni ili aktivni zaštitni gas koji smanjuje potencijal jonizacije prostora između elektrode i štiti od prodora atmosferskih gasova.
Ako je zaštitni gas inertan (argon Ar, helijum He ili neka mešavina gasova koja se ponaša kao inertni gas) postupak se naziva MIG zavarivanje. Zaštitni gas može biti i aktivan (CO_2 ili mešavina gasova koja ima funkciju kao aktivan gas) tada se taj postupak naziva MAG zavarivanje [3].
U ovom radu, prikazana je optimizacija tehnologije zavarivanja postupkom MIG/MAG, sa ciljem da se na što brži i ekonomičniji način izvrši zavarivanje MPM limova, uz što manje isparavanje polimernog međusloja, čime se stvaraju preduslovi za smanjenje deformacija čeličnih limova.



Slika 1. *Elektrolučno zavarivanje topljivom elektrodom u zaštitnom gasu: 1. Zaštitni gas; 2. Izvor struje; 3. Komandni ormar; 4. Boca sa zaštitnim gasom; 5. Kotur sa el. žicom; 6. Dovod žice [1].*

2. EKSPERIMENTALNI DEO

2.1. Zavarivanje MPM limova

MPM limovi u ovom radu se sastoje iz dve čelične ploče debljine po 3 mm. Između njih se nalazi sloj vinil kopolimernog monomera, debljine 0,05 mm, koji povezuje dva čeilčna lima. Hemijski sastav limova je prikazan u tabeli 1, a mehaničke osobine u tabeli 2. U tabeli 1 dat je C ekvivalent (C_{EKV}) koji je dobijen na osnovu sledeće jednačine [4]:

$$C_{EKV} = C+Mn/6+(Cr+Mo+V)/5+(Ni+Cu)/15\ [\%] \quad (1)$$

Tabela 1. *Hemijski sastav MPM limova [mas. %].*

% C	% Si	% Mn	% S	% Cr	% P	% Al
0,03	0,011	0,185	0,009	0,049	0,011	0,039
% Cu	% Ni	%Fe	C_{EKV}			
0,019	0,025	ostatak	0,074			

Tabela 2. *Mehaničke osobine MPM limova.*

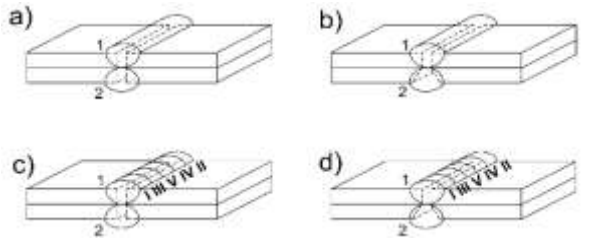
Granica tečenja Re [MPa]	Zatezna čvrstoća Rm [MPa]	Izduženje A [%]
176	281	40

U svrhu eksperimentalnog ispitivanja, MPM limovi su isečeni na sledeće dimenzije: 250 x 200 mm. Limovi su spojeni pomoću MAG postupka, po dužoj strani pravougaonika, koristeći uređaj EWM Vega 500, i Boehler EMK 6 punu elektrodnu žicu za zavarivanje sa hemijskim sastavom prikazanim u tabeli 3.

Tabela 3. *Kataloški hemijski sastav elektrodne žice Boehler EMK 6*

% C	% Si	% Mn
0,1	0,9	1,4

Primenjeni su različiti parametri zavarivanja, od kojih su najvažniji tipovi zavara prikazani na slici 2. Korišćene su četiri vrste sučeonih šavova: I šav sa 1 mm zazora, kombinacija I šava (gornja ploča) i V šava (donja ploča). Zavarivanje u jednom prolazu i zavarivanje u segmentima. Kod kombinacije I i V šava, prvi prolaz je izveden na gornjem delu šava čime se omoguća isparavanje polimera kroz donji deo šava.



Slika 2. *Sučeoni zavari: a) I šav, zavarivanje u jednom prolazu sa obe strane; b) kombinacija I šava (gornja ploča) i V šava (donja ploča), zavarivanje sa obe strane; c) I šav, zavarivanje u segmentima sa obe strane; d) Kombinacija I i V šava, zavarivanje u segmentima sa obe strane.*

Parametri procesa zavarivanja su prikazani u tabeli 4, dok je plan zavarivanja prikazan u tabeli 5. Korišćene su dve vrste zaštitnog gasa, CO_2 za prva četiri uzorka (1-4), i mešavina C18 (Ar + 18 % CO_2) za sledeća četiri uzorka (5 – 8).

Ispitivanje izrađenih epruveta na: zatezanje, savijanje i makro ispitivanja izvršena su na Fakultetu Tehničkih Nauka u Labaratoriji za ispitivanje materijala na Departmanu za proizvodno mašinstvo.

Ispitivanja zateznih karakteristika su urađena na proporcionalnim epruvetama širine 8 mm i paralelne dužine 70 mm na mehaničkoj kidalici VEB ZDM 5/91. Zatezanje je izvršeno na po tri epruvete od svakog uzorka.

Tabela 4. *Parametri procesa zavarivanja*

Vrsta zavarivanja	DC (+)
Struja zavarivanja	150 A
Brzina zavarivanja - I šav	42 cm/min
Brzina zavarivanja - V šav	14 cm/min
Nagib elektrode	60 o
Zaštitni gas	CO2 or C18 mix

Tabela 5. *Plan zavarivanja.*

Oznaka	zavarivanje kao na slici:	Brzina zavarivanja [cm/min]		Zaštitni gas
		Gornji prolaz	Donji Prolaz	
1	Slika 1a	42	42	CO_2
2	Slika 1b		14	CO_2
3	Slika 1c		42	CO_2
4	Slika 1d		14	CO_2
5	Slika 1a		42	C18
6	Slika 1b		14	C18
7	Slika 1c		42	C18
8	Slika 1d		14	C18

Za ispitivanje savijanjem korišćen je trn ϕ24 mm, oslonci su bili na rastojanju od 36,5 mm, ispitivanje je vršeno do ugla savijanja od 180^o. Ispitivane epruvete su širine 15 mm, a dužine 100 mm. Savijanje je izvršeno na po dve epruvete od svakog uzorka kako bi ispitali obe strane zavarenog spoja.

Makro i mikro struktura je ispitana nakon standardne metalografske pripreme koja se sastoji iz sečenja, brušenja na brusnim papirima različite granulacije (120, 220, 320, 500, 1000, 1500, 2000), poliranja dijamantskom suspenzijom (granulacije 6, 3, 1 μm) i nagrizanja sa 3 % azotnom kiselinom (Nital).

Tvrdoća je ispitana metodom po Vikersu na uređaju VEB HPO-250, sa opterećenjem od 10 kg po odgovarajućoj šemi. Izvršena su po tri merenja tvrdoće u svakoj zoni (osnovni materijal (OM), zona uticaja toplote (ZUT) i metal šava (MŠ).

Uticaj zavarivanja na polimerni međusloj je ispitan nakon sečenja čeličnih limova po metalu šava i njihovog razdvajanja kako bi pristupili polimernom sloju.

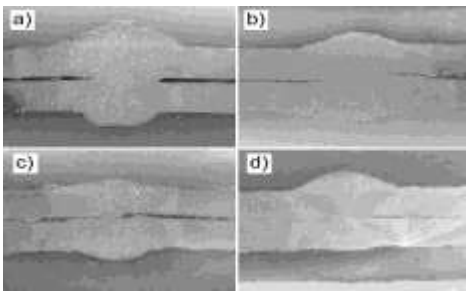
Merenja su izvršena od zavarene ivice lima do karakterističnih rastojanja. Ova karakteristična rastojanja odgovaraju istopljenom, oštećenom i međusobno povezanom poli-meru.

3. REZULTATI I DISKUSIJA

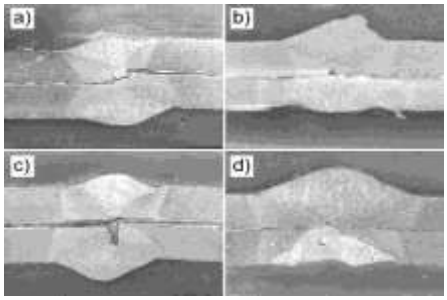
3.1 Makro ispitivanje

Makro slike spojeva su prikazane na slikama 3 i 4. U svim uzorcima osim u uzorcima 6, 7 i 8 (slika 3b, c, d) je postignuta puna penetracija.

Usled toga, u uzorcima koji su zavareni sa C18 zaštitnim gasom je postignuta manja, nepotpuna penetracija. Na slikama 3 i 4 se može videti tipičan šav sa zonom stubastih kristala. Takođe se može primetiti šira zona uticaja toplote.



Slika 3. *Uzorci zavareni sa zaštitnim gasom CO_2: a) uzorak 1; b) uzorak 2; c) uzorak 3; d) uzorak 4.*



Slika 4. *Uzorci zavareni sa zastitnim gasom C18: a) uzorak 5, b) uzorak 6; c) uzorak 7; d) uzorak 8.*

3.2 Ispitivanje tvrdoće

Rezultati ispitivanja tvrdoće (HV10) su prikazani u tabeli 6. Tvrdoća osnovnog materijala je relativno ujednačena i odgovara zateznoj čvrstoći koja je prikazana u tabeli 2. Sve vrednosti za tvrdoću dobijene u ZUT-u imaju vrednost ispod 300HV, međutim, postoje značajne varijacije. Kod uzoraka sa V šavom, koji su zavareni zaštinim gasom C18, u segmentima, je dobijena nešto veća tvrdoća u ZUT-u i MŠ-u, što je verovatno rezultat potpunog izlaganja isparenom ugljeniku iz polimernog međusloja.

Tabela 6. *Rezultati tvrdoće HV10.*

Uzorak, br. prolaza		Tvrdoća HV10														
		OM			ZUT			ŠAV			ZUT			OM		
1	1	109	117	109	114	118	115	169	155	152	121	121	113	107	115	115
	2	121	117	117	115	118	115	169	161	154	123	117	115	121	115	117
2	1	107	115	121	127	150	131	152	151	146	123	123	116	117	117	115
	2	109	107	119	118	120	127	169	178	153	112	114	121	109	115	115
3	1	121	109	109	147	129	126	189	191	187	129	127	121	117	117	121
	2	119	121	107	131	126	146	176	175	182	133	128	133	109	117	117
4	1	121	107	117	120	125	119	194	188	180	131	130	124	117	117	117
	2	115	119	121	117	120	120	157	155	160	132	121	131	119	115	119
5	1	121	121	109	131	131	140	206	206	201	126	127	126	117	115	115
	2	121	109	117	133	125	131	192	197	199	124	131	130	117	115	114
6	1	109	109	121	136	143	144	187	182	160	133	126	120	117	118	117
	2	121	115	119	125	118	114	177	170	180	152	142	143	115	121	117
7	1	117	107	109	144	146	169	206	206	185	136	128	128	121	106	115
	2	121	119	115	132	146	147	159	183	179	143	133	122	117	107	104
8	1	117	121	119	122	123	146	198	189	192	138	126	117	115	115	104
	2	109	115	117	122	122	125	224	227	240	129	123	133	117	115	115

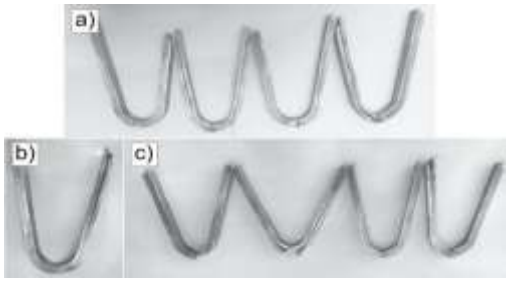
3.3 Ispitivanje zatezanjem i savijanjem

Retultati ispitivanja savijanjem i zatezanjem dati su u tabeli 7. Pri ispitivanju zatezanjem kod uzoraka 1-4 i 7 do loma je došlo u OM-u. Kod uzoraka 5 i 6 do loma je došlo u OM-u i MŠ-u, a kod uzorka 8 je do loma došlo samo u MŠ-u.

Nakon savijanja uočene su prsline na uzorcima 5, 7 i 8, kod uzorka 8 pored prsilna došlo je i do loma epruveta, dok kod ostalih uzoraka nisu uočene prsline, slika 5.

Tabela 7. *Rezultati zatezanja i savijanja*

Uz. br.	Ispitivanje zatezanja		Ispitivanje savijanja
	Zatezna čvrstoća [MPa]	Mesto loma	
1	287±6	OM, OM, OM	Nema prslina ili loma
2	292±3	OM, OM, OM	Nema prslina ili loma
3	292±3	OM, OM, OM	Nema prslina ili loma
4	294±6	OM, OM, OM	Nema prslina ili loma
5	235±35	OM, OM, MŠ	Četiri pukotine
6	243±55	OM, MŠ, OM	Nema prslina ili loma
7	268±11	OM, OM, OM	Jedna prslina
8	161±29	MŠ, MŠ, MŠ	Tri prsline, jedan lom



Slika 5. *Ispitivanje savijanja sa prslinom i prelomom: a) uzorak 5; b) uzorak 7; c) uzorak 8*

3.4 Ispitivanje mikrostrukture

Na slikama 6 i 7 je prikazana mikrostruktura uzoraka 1 i 8, usled njihove velike razlike u mehaničkim osobinama. Sve mikrostrukture su slične, osim u zoni ponovnog zagrevanja, koja nije prisutna kod uzorka 8.

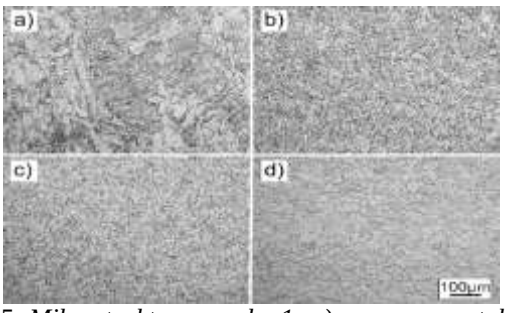
3.5 Ispitivanje debljine polimernog sloja nakon zavarivanja

Oblast isparavanja, topljenja i međusobnog povezivanja je prikazana na slici 8, dok su vrednosti izmerenih rastojanja date u tabeli 8.

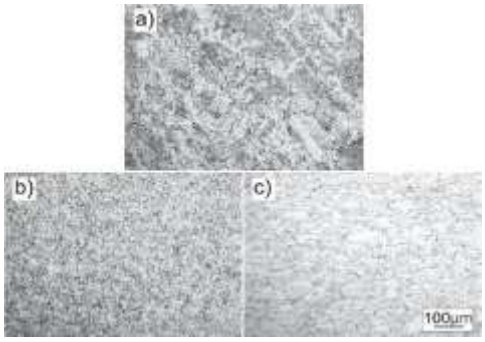
Rastojanja isparavanja, topljenja i među-sobnog povezivanja kod uzoraka 1-4 su veće u poređenju sa uzorcima 5-8, verovatno usled primene CO_2 kao zaštitnog gasa pri zavarivanju, nasuprot C18, koji ima manji sadržaj kiseonika.

Najveće oštećenje polimernog sloja je uočeno kod uzorka 2, najverovatnije zbog visokog unosa toplote, koja je uticala na dobijanje viših mehaničkih osobina.

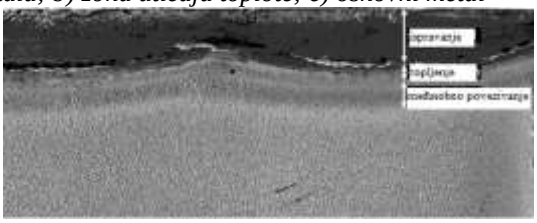
Primenom C18 kao zaštitnog gasa dobija se manje rastojanje isparavanja polimernog međusloja, ali se takođe dobijaju i lošije mehaničke osobine.



Slika 5. *Mikrostruktura uzorka 1: a) zavar-zona stubastih kristala; b) zavar-zona ponovnog zagrevanja-; c) zona uticaja toplote; d) osnovni metal*



Slika 6. *Mikrostruktura uzorka 8: a) zavar-zona stubastih kristala; b) zona uticaja toplote; c) osnovni metal*



Slika 8. *Oblast isparavanja, topljenja i međusobnog povezivanja kod uzorka 4.*

Tabela 8. *Rastojanja isparavanja, topljenja i medjusobnog povezivanja.*

Uz.br	Rastojanje isparavanja (prosek) [mm]	Rastojanje topljenja (prosek) [mm]	Rastojanje međusobnog povezivanja (prosek) [mm]
1	6,5-13,2 (9,85)	2,2-5,3 (3,75)	5,7-7,3 (6,5)
2	7,5-15,1 (11,3)	3,8-5,3 (4,55)	7,7-9,6 (8,65)
3	7,1-12,2 (9,65)	2,8-4,7 (3,8)	3,6-6,8 (5,2)
4	5,3-11,8 (8,55)	3,1-3,5 (3,3)	5,0-8,8 (6,9)
5	4,6-8,0 (6,3)	1,6-3,0 (2,3)	3,6-4,8 (4,2)
6	3,6-10,3 (6,95)	4,4-6,2 (5,3)	4,6-4,7 (4,65)
7	8,0-8,2 (8,1)	2,2-2,3 (2,25)	4,4-5,5 (4,95)
8	4,0-5,0 (4,5)	1,8-3,2 (2,5)	1,5-1,7 (1,6)

Ako se uzme da su sa pogleda strukturnog integriteta mehaničke osobine najvažnije, uzorci 1-4 imaju prednost u odnosu na uzorke 5-8. Optimalni uzorak, osim kratkog rastojanja isparavanja polimernog međusloja treba da ima i najkraći utrošak vremena.

Iako je brzina zavarivanja koristan indikator, prekidanje šavova kod uzoraka 3 i 4 takođe zahteva dodatno vreme, naročito kod uspostavljanja luka i kod preklopa šavova. Kako su karakteristična rastojanja slična kod uzoraka 1 i 3, uzorak 1 se znatno brže zavaruje pošto se zavarivanje izvodi u jednom prolazu.
Usled toga, uzorak 1 se može smatrati kao optimalni uzorak od svih ispitanih uzoraka.

4. ZAKLJUČAK

U skladu sa rezultatima eksperimenta, mogu se izvući sledeći zaključci:

- Puna penetracija je postignuta u uzorcima 1-4, koji su zavareni sa CO_2 zaštitnim gasom.
- Najveća zatezna čvrstoća i zadovoljavajući rezultati savijanja su dobijeni kod uzoraka 1-4. Ovo je razumljivo usled pune penetracije dobijene na ovim uzorcima. Kod uzoraka koji su zavareni sa C18 zaštitnim gasom, dobijene su značajno lošije mehaničke osobine.
- Uzorci 5-8, iako su njihova rastojanja isparavanja manja u odnosu na uzorke 1-4, imaju niže mehaničke osobine.

Način izrade uzorka u jednom prolazu sa obe strane (uzorak 1) predstavlja izbor tehnologije zavarivanja, jer poseduje optimalnu kombinaciju mehaničkih osobina i tehnološke jednostavnosti u smislu brzine pripreme uzorka i brzine zavarivanja.

LITERATURA:

[1] Palić Vlastimir: Zavarivanje 1, Fakultet Tehničkih Nauka Novi Sad, 1987.
[2] http://antiphon.se/wp-content/uploads/antiphon_mpm_engelsk09_id340.pdf (23.06.2018)
[3] Grupa autora, Zavod za zavarivanje, Postupci zavarivanja i oprema za zavarivanje, Beograd 2004.
[4] Ivan Hrivnjak, Zavarljivost čelika, IRO Građevinska knjiga, Beograd, 1982., 189.
[5] https://www.resale.de/ewm-wega-500-schweissger%C3%A4t/Nr-6073010 (23.06.2018)

Kratka biografija:

Milko Madžar rođen u Novom sadu 1991. god. Master rad na Fakultetu Tehničkih Nauka iz oblasti mašinstva – Optimizacija elektrolučnog zavarivanja topljivom elektrodom u zaštitnom gasu pri zavarivanju MPM sendvič limova.

Sebastian Baloš rođen je u Somboru 1974. god. Doktorirao je na Fakultetu Tehničkih Nauka 2010. godine iz oblasti Materijali i tehnologija spajanja. 2011. stekao zvanje docenta, a 2016. u vanrednog profesora.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE00Benka

IZAZOVI U RAZVOJU FLASH MULTIPLATFORMSKIH APLIKACIJA

CHALLENGES IN THE DEVELOPMENT OF FLASH MULTIPLATFORM APPLICATIONS

Vladislav Benka, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad opisuje načine renderovanja Flash aplikacija korišćenjem svojih metoda. Pokazuje na njihove procese, pravila i ograničenja. Pored toga, postoji i primer implementacije Stage3D API-ja kao i nekoliko reči o Starling Freamwork-a.*

Ključne reči: *Flash, GPU, Stage3D, Android, iOS*

Abstract – *This work describes ways to render Flash application by using its own methods. It points to their processes, rules and limitations. In addition, there is also an example of the implementation in Stage3D API, as well as a few words about Starling freamwork.*

Keywords: *Flash, GPU, Stage3D, Android, iOS*

1. UVOD

Pojavom iOS i Android mobilnih uređaja, Adobe Systems je nastavio da održava svoju poziciju u web svetu, napravivši Flash Player plugin za mobilne uređaje koji nije dobro prihvaćen od iOS Android platforme.

Zbog toga Adobe Systems proširuje svoj Adobe AIR višeplatformski sistem sa AIR SDK 2.0 bibliotekom (maja 2010.). Ovom bibliotekom AIR po prvi put ima podršku za mobilne uređaje (iOS i Android).

Pošto su u to vreme na tržištu bili mobilni uređaji sa dosta slabim CPU-om, animacije novonastalih aplikacija bile su suviše spore. Zbog toga Adobe Systems uvodi GPU mod za renderovanje ekrana sa AIR SDK 2.5 bibliotekom (oktobra 2010.). GPU mod podržan je samo za AIR aplikacije na mobilnim uređajima.

Dolaskom Adobe AIR SDK 3.0 biblioteke (oktobra 2011.), Adobe Systems uvodi novi Direct mod za renderovanje ekrana, mobilnih i desktop uređaja. Isti ovaj mod uvodi se i u Adobe Flash Player 11.

Od tada postoje tri moda za renderovanje Flash aplikacija:

- CPU mod - podržan je za sve AIR platforme, kao i u Flash Player plugin-u. U ovom modu renderovanje ekrana vrši se preko CPU-a.
- GPU mod - podržan je samo za AIR mobilne platforme (iOS i Android). Kompozicija se vrši uz pomoć GPU-a, a rasterizovanje se vrši preko CPU-a.
- Direct mod - podržan je za sve AIR platforme, kao i u Flash Player plugin-u. U ovom modu renderovanje je u potpunosti direktno preko GPU-a.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivetić, red. prof.

2. CPU MOD ZA RENDEROVANJE EKRANA

U CPU modu renderovanje ekrana se vrši u potpunosti preko CPU-a, metodom DisplayList. Ova metoda pokazala je odlične performanse od samog početka i postala konkurentna u web svetu. Ona se zapravo svodi na iscrtavanje samo dela ekrana, dela koji se menja. Proces renderovanja sadrži tri osnovna koraka, a to su:

- Calculating dirty regions. Renderovanje započinje skeniranjem DisplayList-a, u potrazi za objektima markiranih kao Dirty (prljavi). To su zapravo objekti kojima se izmenila neka od vizuelnih osobina kao što su pozicija (x,y), nagib (rotation), veličina (scale) ili neki Blending filter. Posle skeniranja, kreira se najviše do tri pravougaona kontejnera koji sadrže Dirty objekte. Ove kontejnere nazivamo još Dirty Region i to su zapravo delovi ekrana koje je potrebno ponovo iscrtati.
- Rendering Dirty Region. Za svaki Dirty Region, Flash Player treba da utvrdi šta treba uraditi sa novim pikselima unutar kontejnera. Za ovaj postupak, kreira se scena u internom baferu zvanom Display Buffer i tu postoje sledeće faze:
  - Building edges from DisplayObjects. Flash Player ponovo skenira DisplayList, tražeći objekte koji se preklapaju sa Dirty Region-om. Rastavlja se svaki nađen objekat u set ivica i ispuna (ovo mogu biti čiste boje, gradijentne boje ili slike). Ako objekat u sebi sadrži druge objekte, ovaj proces se izvršava ponovo unutar child objekta i tako rekurzivno.
  - Rasterizing Edges. Flash Player konvertuje ivice i ispune u piksele. Dirty Region se deli na skup horizontalnih linija zvanih Scan Lines, za svaki vertikalni piksel. Sada Flash Player prolazi ovim linijama, piksel po piksel i određuje njihovu boju zasnovanu na ispunu tog područja između ivica koje ih vežu. Vreme rasterizovanja zavisi od veličine Dirty Region-a (broja piksela) i kompleksnosti objekta (objekat u objektu).
  - Applying Filters. Na kraju ovog koraka dodaju se filteri kao što su senka, sjaj, itd. Oni se dodaju na rasterizovanu sliku, kreirajući novu sliku. Postoji i mogućnost da objekat u sebi ima child objekat na kome je primenjen filter. U tom slučaju child objekat se prvo mora rasterizovati, pa primeniti filter i kao rezultat dobijamo sliku child objekta koji se koristi za određivanje ivica i ispuna parent objekta (Building edges from DisplayObjects).
- Copying to Scene - na kraju kada renderovanje prođe, Flash Player kopira piksele sa Display Buffer-a na ekran. Vreme izvršavanja ovog koraka zavisi od broja

piksela, kao i od neželjenog zaključavanja ekrana kojim upravlja operativni sistem uređaja.

Najbolja praksa kod stvaranja aplikacije korišćenjem CPU moda jeste pridržavanje određenih pravila, a to su:

- Korišćenje manjeg broja objekata koji se animiraju.
- Smanjivanje dela ekrana (dirty regions) gde se animacija izvršava, a koju CPU treba da iscrta.
- Smanjiti kompleksnost objekata, tj. da objekti nisu sastavljeni od objekata koji su sastavljeni od objekata i tako dalje.
- Smanjiti korišćenje ili potpuno ne koristiti alpha (transparentnost) svojstva objekta.

3. GPU MOD ZA RENDEROVANJE EKRANA

GPU mod je pre svega optimizovan za brzo pomeranje bitmap-a, pa stoga radi maksimalnog ubrzanja i poboljšanja performansa aplikacije, treba shvatiti kako konvertovati, do tad teške, vektorske objekte u brze rasterizovane bitmap-e.

U Flash-u postoje dve metode za konvertovanje postojeće vektorske grafike u bitmap-e, a to su:

- Caching metoda - keširanjem vektora kao bitmap-a,
- Draw metoda - crtanjem vektora kao bitmap-a.

Uključivanjem MovieClip-ovog svojstva cacheAsBitmap dobijamo da se MovieClip posmatra kao bitmap, bez obzira šta se unutar njega nalazi. Sada više nije potrebno ponovno rasterizovanje MoviClip-ovog sadržaja za vreme njegovog pomeranja. Kod za keširanje MovieClip-a je vrlo jednostavan i prikazan na Listingu 1.

```
vektorMC.cacheAsBitmap = true;
```

Listing 1. *Uključivanje svojstva cacheAsBitmap*

Keširanje u velikoj meri poboljšava performanse, ali uz sledeća upozorenja. Ako se unutar MoviClip-a koji keširamo vrši neka animacija, Flash Player će morati ponovo da rasterizuje i kešira MovieClip. Pošto je proces keširanja sistemski intenzivan, to će stvoriti negativan efekat u performansama aplikacije. Ovako keširan MovieClip je spreman samo za pomeranje po x i y osi.

Kod skaliranja i rotiranja MovieClip-a dolazi do ponovnog zahteva za rasterizovanjem i keširanjem. Zato postoji još jedno svojstvo MovieClip-a koje treba uključiti radi dobijanja performansi kod transformisanja keširanog MovieClip-a.

```
vektorMC.cacheAsBitmap = true;
vektorMC.cacheAsBitmapMatrix = new
Matrix();
```

Listing 2. *Uključivanje svojstva cacheAsBimapMatrix*

Sa ovim uključenim svojstvom, keširan MovieClip može da se skalira, rotira i menja svoju transparentnost, bez zahteva za ponovnim keširanjem. CacheAsBitmapMatrix svojstvo postoji samo u GPU render modu.

Crtanje vektora kao bitmap-a (Draw metoda) je proces u kome se uz pomoć koda vrši preslikavanje podataka piksela vektorse grafike u BitmapData objekat i kreiranje bitmap-a na kraju.

```
import flash.display.BitmapData;
import flash.display.Bitmap;

var bmd:BitmapData = new BitmapData(vektorM
C.width, vektorMC.height, true, 0x00000000)
bmd.draw(vektorMC);
var bmp:Bitmap = new Bitmap(bmd);
addChild(bm);
```

Listing 3. *Crtanje vektora kao bitmap-a*

Prvo se kreira BitmapData objekat sa dimenzijom koja obuhvata vektorski MovieClip, sa transparentnim svojstvom i transparentnom pozadinskom bojom. Posle se poziva metoda draw koja kreira podatke za bitmap iz vektorskog MovieClip-a. Na kraju se kreira bitmap objekat uz pomoć novonastalog objekta BitmapData i dodajemo ga na DisplayList-u radi prikazivanja.

Draw metoda počinje sa koordinatama (0,0) vektorskog MovieClip-a, a skaliranje i rotiranje vektorskog MovieClipa ne utiče na draw metod, što znači da se uvek dobija bitmap vektorskog MovieClipa kakav on zapravo jeste.

BitmapData objekat koristi jedinstvenu referencu, što znači da je moguće imati jednu kopiju BitmapData u memoriji i poslati te podatke na bezbroj Bitmap objekata koje se nalaze na ekranu, bez da se za njih zauzima dodatna memorija.

Za Caching metod, Flash Player automatski oslobađa memoriju keširanog vektorskog objekta kada se on više ne nalazi na ekranu, dok se kod Draw metode, memorija BitmapData objekata mora osloboditi pomoću koda. Koristi se metoda dispose(), a preporučuje se i nuliranje reference objekta.

```
bmd.dispose();
bmd = null;
```

Listing 4. *Oslobađanje memorije BitmapData-e*

Aplikacije koje koriste GPU mod za renderovanje koriste više RAM memorije od aplikacija koje rade u CPU modu, stoga se mora jako voditi računa o kreiranju bitmap-a i njegovog oslobađanja iz memorije.

Kod GPU moda za renderovanje postoje nekoliko ograničenja, a to su:

- Ako GPU ne može da renderuje objekat, on se ne prikazuje, a takođe ne postoji mogućnost prebacivanja u CPU mod radi njegovog renderovanja.
- Blend modovi, kao što su: layer, alpha, erase, overlay, hardlight, lighten i darken, nisu podržani.
- Filteri na objektima nisu podržani.
- PixelBlender nije podržan.
- GPU renderovanje je nedostupno za stare android uređaje koji imaju slabu GPU jedinicu.

GPU mod je efikasan samo za bitmap-e, jednostavne vektore i display objekte koji imaju uključena svojstva cacheAsBitmap i cacheAsBitmapMatrix. Najbolja praksa kod stvaranja aplikacije u GPU modu jeste pridržavanje sledećih pravila:

- Ograničenje broja vidljivih objekata na stage-u.
- Ponovno korišćenje objekata.
- Korišćenje bitmap-a rezolucije od 2^n x 2^m.
- Uključivanje svojstva cache za vektorske objekte koji se ne menjaju često.
- Skaliranje bitmap-a na željenu dimenziju pre uvoženja u samu aplikaciju.
- Spuštanje parametra za stage kvalitet (LOW).

GPU mod je alternativa za renderovanje aplikacija na mobilnim uređajima uz potencijal pružanja vrhunske performanse, ali u isto vreme i teži način za kodiranje.

4. DIRECT MOD ZA RENDEROVANJE EKRANA

Pre pojave Direct moda za renderovanje, u Flesh-u su se razvijale 3D aplikacije pomoću softverskih 3D engine-a, gde se renderovanje vršilo preko CPU-a, i kao rezultat toga, aplikacije su bile jako spore.

Pojavom Direct moda za renderovanje i Stage3D API-a (kodno ime Molehill), 3D sadržaj se renderuje u realnom vremenu, direktno uz pomoć GPU-a.

Stage3D je API sa niskim nivoom pristupa GPU API-ju, posebno dizajniran da iskoristi maksimalnu snagu hardverskih 3D grafičkih jedinica uređaja. Neverovatno je brz, jednostavan i služi isključivo za 3D renderovanje.

Prebacivanjem procesa renderovanja na GPU jedinicu, oslobađa se CPU jedinica koja sad može da obavlja druge proračune. Ovo otvara mogućnost za kompleksnije aplikacije pošto su sada CPU i GPU 100% u upotrebi.

Kako 3D grafika funkcioniše, objašnjeno je u sledećem delu. Generalno, 3D scena definisana je grupom 3D geometrijskih objekata (mesh). 3D geometrijski objekat predstavljen je kao skup poligona (polies). Jedan poligon može biti sastavljen od jednog ili više trouglova (triangles), a svaki trougao sačinjen je uz pomoć tri tačaka (vertices). Dakle, 3D scena je skup definisanih tačaka i eventualno dodatnih informacija za renderovanje, kao što su boja ili tekstura.

Kod Stage3D API-a, podaci 3D scene se definišu i prosleđuju u GPU. Oni se smeštaju u GPU memoriju gde se obrađuju, trougao po trougao. Kao rezultat dobija se finalna renderovana slika, spremna za prikaz na ekranu.

3D renderovanje vrši se u GPU jedinici uz pomoć 3D programibilnog renderskog pipeline-a. 3D renderski pipeline je proces renderovanja, podeljen u više elementarnih operacija. Sastoji se od niza blokova, gde svaki blok izvršava jednu elementarnu operaciju, a postavljeni su kaskadno, tako da je izlaz jednog bloka ulaz sledećeg bloka.

Programibilni pipeline znači da podržava Shaders programe. To su mali programi koji se pokreću u GPU-u, pomoću kojih je moguće uticati na tok renderovanja 3D scene. Ovi Shader programi se, takođe, prosleđuju na GPU. Postoje dva tipa Shader programa, a to su:

- Vertex Sahader - programi koji utiču na transformaciju položaja tačaka.
- Fragment Shader - programi koji utiču na transformaciju boja trouglova.

Adobe Systems kreirao je dva Shading jezika za kreiranje programa:

- AGAL (Adobe Graphics Assembly Language) jezik - asemblerski jezik niskog nivoa.
- Pixel Blender 3D - jezik višeg nivoa, lakši za razumevanje, ali teži za potpunu kontrolu.

5. PRIMER STAGE3D API-JA

U sledećem delu je primer za prikaz obojenog kvadrata, korišćenjem Stage3D API-ja. Primarna klasa Stage3D API-ja jeste klasa Context3D, koja služi kao površina za 3D renderovanje. Zato je, na početku aplikacije, potrebno definisati event listener za CONTEXT3D_CREATE i ujedno pozvati zahtev za njeno kreiranje.

```
stage.stage3Ds[0].addEventListener(Event.CO
NTEXT3D_CREATE, configContext3D);
stage.stage3Ds[0].requestContext3D();
```

Listing 5. *Kreiranje Context3D-a*

Čim se Context3D kreira, vrši se njegova konfiguracija pozivom metode configureBackBuffer uz parametre:

- širine i visine površine za renderovanje,
- minimalnim nivoom anti-aliasing-a i sa
- parametrom true, za kreiranjem depth i stencil bafera.

```
var context:Context3D;
context=stage.stage3Ds[0].context3D;
context.configureBackBuffer(stage.StageWidt
h, stage.StageHeight, 0, true);
```

Listing 6. *Konfiguracija Context3D-a*

Posle konfiguracije Context3D-a, definišemo 3D geometrijski objekat, u ovom primeru obojeni kvadrat, koji sadrži četiri tačke i svaka od njih ima svoju boju. To činimo definisanjem Vertex Buffer-a, strukture koja sadrži sve informacije o tačkama geometrijskog objekta. Ovaj Vertex Buffer sadržaće 6 Vertex atributa. Tri atributa za položaj (x,y,z) i tri atributa za boju (r,g,b). Definisan Vertex Buffer držimo u vektor nizu. Zatim kreiramo VertexBuffer3D instancu (za 4 tačke sa po 6 atributa) i preko nje otpremamo Vertex Buffer na GPU.

```
var vertices:Vector.<Number>=
Vector.<Number>([
-0.3,-0.3,  0,  1,  0,  0, // x,y,z,r,g,b
-0.3, 0.3,  0,  0,  1,  0,
 0.3, 0.3,  0,  0,  0,  1,
 0.3,-0.3,  0,  1,  0,  1
]);
var vBuff:VertexBuffer3D;
vBuff=context3D.createVertexBuffer(4, 6);
vBuff.uploadFromVector(vertices, 0, 4);
```

Listing 7. *Definisanje VertexBuffer-a Context3D-a*

Pošto smo definisali Vertex Buffer, sledeći korak je definisanje Index Buffera. Geometrijski objekat se rastavlja na skup trouglova. Ovde se kvadrat rastavlja na dva trougla koja se definišu pomoću Index Buffer-a. Redosled unosa tačaka trougla mora biti u smeru kazaljke na satu, da bi renderovao lice slike.

Kao kod Vertex Buffer-a i ovde se Index Buffer čuva u vektor nizu. Kreiramo IndexBuffer3D instancu (2 trougla, 6 tačaka) i preko nje otpremamo Index Buffer na GPU.

```
var indices:Vector.<uint>=
Vector.<uint>([0, 1, 2, 2, 3, 0]);
var iBuffer:IndexBuffer3D;
iBuffer=context3D.createIndexBuffer(6);
iBuffer.uploadFromVector(indices, 0, 6);
```

Listing 8. *Definisanje IndexBuffer-a Context3D-a*

Sada su nam potrebni još Vertex i Fragment Shader-i. Vertex Shader jednostavno transformiše položaje tačaka prema transformacionoj matrici koju definišemo ActionScript-om, dok Fragment Shader transformiše boju. Korišćenjem AGAL-a kreiramo kod za Shader-e i pomoću Program3D klase otpremamo Shader-e na GPU.

```
var shaderP:Program3D;
var vShader:AGALMiniAssembler=new
AGALMiniAssembler();
vShader.assemble(
    Context3DProgramType.VERTEX,
    "m44 op, va0, vc0\n" + // pos
    "mov v0, va1" // copy color
);
var fShader:AGALMiniAssembler = new
AGALMiniAssembler();
fShader.assemble(
```

```
    Context3DProgramType.FRAGMENT,
    "mov oc, v0"
);
shaderP=context.createProgram();
shaderP.upload(vertexShader.agalcode,
fragmentShader.agalcode);
```

Listing 9. *VertexShaderar i FragmetShader*

Scena je sada spremna za rederovanje i preostala je implementacija funkcije za renderovanje. Ona se izvršava u svakom frejmu na sledeći način:

- Prvo se poziva metoda **Context3D::clear** koja setuje sve piksele (Color Buffer) na boju koju prosleđujemo. U primeru (1,1,1,1), to je bela boja bez transparencije.
- Setuju se Vertex Buffer-i, tj. tačke i boje. Tačke se setuju na vertex atributsku lokaciju 0 (va0), a boje na vertex atributsku lokaciju 1 (va1).
- Prosleđuje se Shader Program.
- Prosleđuje se transformaciona matrica radi menjanja položaja našeg geometrijskog objekta, pomoću metode **Context3D::setProgramConstantsFromMatrix**.
- Poziva se metoda **Context3D::drawTriangles** koja renderuje trouglove na površini za renderovanje.
- Na kraju kada smo završili sa renderovanjem našeg geometrijskog objekta poziva se metoda **Context3D::present,** koja govori Stage3D-u da je aplikacija završila sa renderovanjem i da je spremna da se prikaže na ekranu (Color Buffer).

```
context.clear(1, 1, 1, 1);

context.setVertexBufferAt(0,vBuff,0,Context
3DvertexBufferFormat.FLOAT_3);

context.setVertexBufferAt(1,vBuff,3,Context
3DVertexBufferFormat.FLOAT_3);

context.setProgram(sProgram);

var mat:Matrix3D = new Matrix3D();
mat.appendRotation(getTimer()/50,
Vector3D.Z_AXIS);

context.setProgramConstantsFromMatrix(Conte
xt3DProgramType.VERTEX, 0, mat, true);
context.drawTriangles(iBuff);

context.present();
```

Listing 10. *Funkcije za renderovanje*

6. STARLING FREAMWORK

Starling je ActionScript 3.0, 2D Framework, razvijen na osnovu Stage3D API-ja. Omogućava brz razvoj GPU akcelerovanih AIR aplikacija, bez direktnog korišćenja Stage3D API-ja. Starling koristi Stage3D API, koji je zapravo GPU API niskog nivoa, pokrenut preko OpenGL-a i DirectX-a na desktop platformi ili preko OpenGL-a ES2 na mobilnoj platformi.

Starling pojednostavljuje složenost niskog nivoa Stage3D API-ja i omogućava lako i intuitivno programiranje za sve. Jednostavan je i lak za učenje, pogotovo za Flash programere koji su već upoznati sa ActionScript konceptom. Starling API u sebi sadrži dosta klasa sa istim imenima i namenama kao u ActionScript API-ju.

Mnogi smatraju da je Stage3D API striktno namenjen za 3D sadržaje, međutim, u stvarnosti je moguće kreirati i 2D sadržaj. GPU je veoma efikasan kada se radi o iscrtavanju trouglova (triangles), tako da pomoću drawTriangles API-ja iscrtavamo dva trougla (kvadrat) na koje prilepimo teksturu, korišćenjem UV mapiranja. Takav objekat predstavlja se kao Starling Sprite, koji ima većinu osobina kao ActionScript Sprite. Napomena jeste da ako ovom Sprite-u menjamo teksturu u svakom frejmu, dobijamo animirani Starling-MovieClip. Ova tehnika menjanja teksture radi dobijanja animacije naziva se Sprite Sheets tehnika.

Da bi se predstavilo kako Starling smanjuje složenost Stage3D API-ja, moguće je uporediti gorenavedeni kod za prikazivanje kvadrata pomoću Stage3D API-ja i sledeći kod pisan pomoću Starling-a.

```
var tex:Texture=Texture.fromBitmap(new
embeddedBitmap());

var img:Image = new Image(texture);
img.x = 300;
img.y = 150;
addChild(img);
```

Listing 11. *Starling kod za iscrtavanje slike*

7. ZAKLJUČAK

U radu su predstavljeni 3 moda za renderovanja Flash aplikacija kao i Stage3D API, i Starling Freamwork. Tokom izrade ovog rada, vršilo se testiranje performansi ovih modova. Posmatrao se maksimalan broj grafičkih objekata koji se pomeraju na ekranu mobilnih uređaja. Ponašanje aplikacije pratio se pomoću Adobe Scout CC-a programa. Rezultati testova za iPhone 5c su sledeći:

- CPU mod, 200 objekata, 59.8fps.
- GPU mod, 700 objekata, 59fps.
- Starling Freamwork, 1200 objekata, 59fps.
- Stage3D API, 2650 objekata, 59.8fps.

Rezultati za Samsung S4 su:

- CPU mod, 20 objekata, 59.8fps.
- GPU mod, 160 objekata, 60fps.
- Starling Freamwork, 500 objekata, 58.2fps.
- Stage3D API, 2700 objekata, 60.1fps.

Ovim testovima moguće je potvrditi da Stage3D API ima sjajne performanse i da ga netreba izbegavati.

8. LITERATURA

[1] https://www.adobe.com/devnet/scout/articles/understanding-flashplayer-with-scout.html (pristupljeno u sept. 2018.)

[2] https://www.adobe.com/devnet/flashplayer/articles/how-stage3d-works.html. (pristupljeno u sep. 2018.)

[3] I. Thiabult, "Starling – Building GPU Accelerated Application ", O'Reilly Sebastopol, CA 95472, 2012.

Kratka biografija:

Vladislav Benka rođen 6. maja 1978. godine u Stuttgart, Nemačka. 1997. godine upisao je osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu. Položio je sve ispite propisane planom i programom.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.39
DOI: https://doi.org/10.24867/01BE00Repic

OSNOVNE TEHNIKE INTRA I INTER PREDIKCIJE U HEVC-U

BASIC INTRA AND INTER PREDICTION TECHNIQUES IN HEVC

Ivan Repić, Željen Trpovski, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu dat je pregled osnovnih tehnika primenjenih u intra i inter predikciji slike u standardu HEVC.*

Ključne reči: *HEVC, H.264/AVC, intra predikcija, inter predikcija, AMVP, TMVP*

Abstract – *In this paper we give an overview of the basic techniques applied to the intra and inter picture prediction in the HEVC standard.*

Keywords: *HEVC, H.264/AVC, intra prediction, inter prediction, AMVP, TMVP*

1. UVOD

Televizijski sistemi su razvijeni između dva svetska rata. To su bili analogni sistemi, razvijeni su paralelno po nekoliko različitih standarda. Zajednička osobina ovih standarda jeste diskretizacija signala po vertikali i vremenu, dok je signal po horizontali kontinualan. Krajem 20. veka došlo je do razvoja digitalnih televizijskih sistema. U postupku digitalizacije TV signala primenjuju se veoma složene tehnike i operacije. Video zapis se sastoji od sekvence slika. Da bi posmatrač stekao utisak da se radi o pokretnoj slici tj. da ne bi bilo treperenja ili isprekidanog pokreta, potrebno je da bude najmanje 25 slika u 1s, nezavisno od njihovog sadržaja i sličnosti sadržaja u slikama.

U digitalnim TV sistemima bi prenos celih slika bio neekonomičan zbog potrebne brzine digitalnog protoka. Zato su razvijeni složeni postupci kojima se za svaku sliku pronalazi njen suštinski sadržaj. Traži se sadržaj slike koji je neophodno preneti da bi se ova slika tačno reprodukovala u prijemniku. Predikcija slike je postupak kojim se sadržaj slike koju pripremamo za prenos, upoređuje sa drugim slikama, pre i posle nje.

Traže se sličnosti sa drugim slikama i iz date slike se izdvaja samo sadržaj koji je nov. To znači da se umesto prenošenja celih slika, prenosi samo sadržaj koji je nov (razlikuje se od sadržaja koji je već prenet), kao i svi podaci o sličnostima sa drugim slikama. Tema ovog rada jeste analiza predikcija slike u HEVC standardu. Postupak sadrži veoma mnogo složenih tehnika i detalja. Ovde je prikazan samo jedan deo osnovnih tehnika u okviru intra i inter predikcije slike.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Željen Trpovski, vanr. prof.

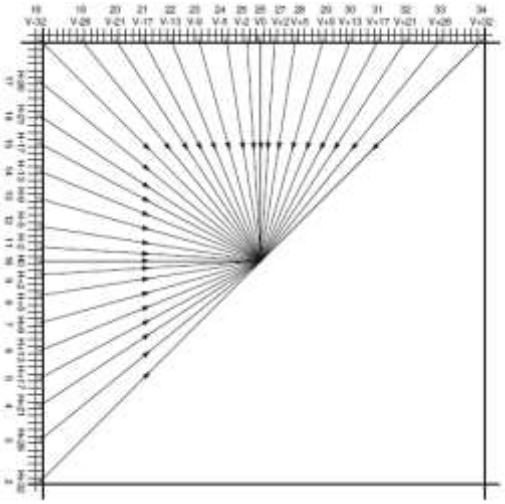
2. INTRA PREDIKCIJA

Intra predikcija predstavlja traženje sličnosti i pravilnosti unutar date slike. Sastoji se iz tri faze:

- formiranje niza referentnih uzoraka,
- predikcija uzorka i
- naknadna obrada dobijenih podataka.

Cilj predikcije jeste da ostvarimo visoku efikasnost kodovanja, uz istovremeno smanjenje računarskih zahteva u procesu obrade i reprodukcije. HEVC u intra predikciji ima set od 35 modova tj. metoda za modelovanje sadržaja slike. Za modelovanje direkcionih struktura koristi se 33 moda koji čine ugaonu predikciju. Za estimaciju glatkog sadržaja slike koriste se dva moda: planarni i DC. Svi modovi mogu se primeniti na blokove svih veličina koji su podržani: 4x4, 8x8, 16x16, 32x32 piksela. Ostvareno je unapređenje u odnosu na prethodni H.264 standard, koji radi sa blokovima veličina od 4x4 do 16x16. H.264 podržava 8 direkcionih modova i DC mod za blokove veličina 4x4 i 8x8, dok za blok 16x16 dozvoljava upotrebu 4 moda, a planarni mod se može primeniti samo na blok 16x16.

Na slici 1. prikazani su direkcioni modovi HEVC-a. Vidimo da modovi pokrivaju ugao od 180 stepeni, tako da je moguće ostvariti predikciju rotacije objekta. Ova rotacija se može ostvariti po tri ose, ali i kombinovano.

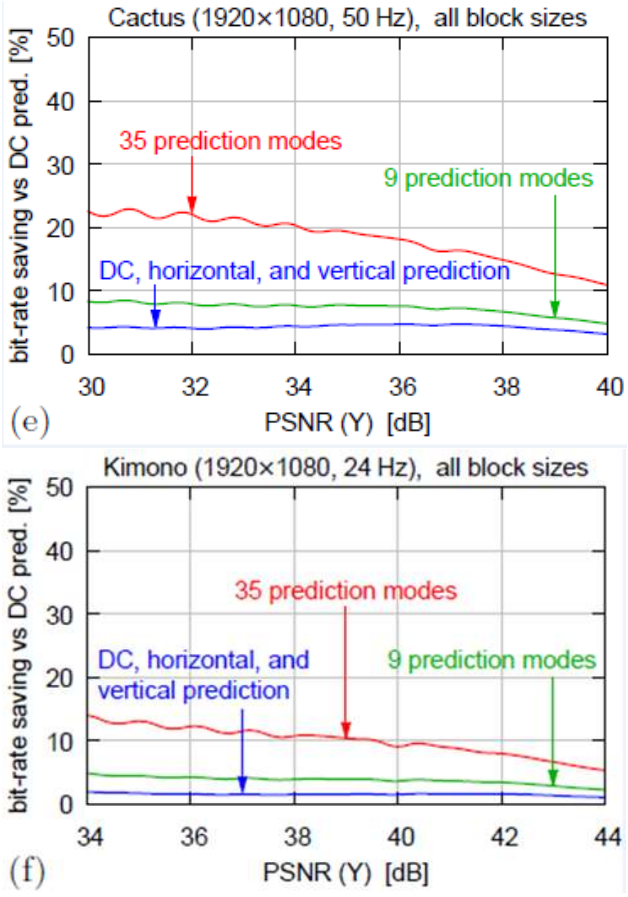


Slika 1. *Definicija ugla ugaone intra predikcije [1]*

U prirodi se mnogo češće pojavljuju vertikalne i horizontalne strukture, nego strukture u drugim pravcima. Sa slike 1. vidimo da su između skoro horizontalnih i skoro vertikalnih modova male razlike ugla, dok su te razlike veće za dijagonalne modove, čime se odlično prate prirodni procesi. Na slici 2. prikazano je kako se u intra predikciji menja zavisnost efikasnosti kodovanja od

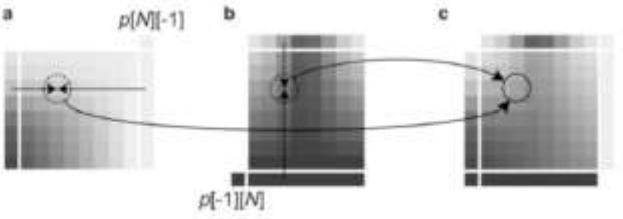
veličine predikcionog bloka pri transformacionom kodovanju. Efikasnost kodovanja raste sa povećanjem veličine predikcionog bloka.

Međutim, intra predikcija je najefikasnija kada radi sa manjim blokovima. Sa povećanjem veličine bloka i obrada postaje računarski kompleksnija. Malo povećanje efikasnosti kodovanja sa povećanjem veličine predikcionog bloka, često nije opravdano. Optimalan izbor veličine predikcionog bloka zavisi od osobina video sadržaja koji se prenosi i od potreba korisnika.



Slika 2. *Ušteda bitske brzine u slučaju kada su prisutni blokovi svih veličina koje HEVC dopušta [2]*

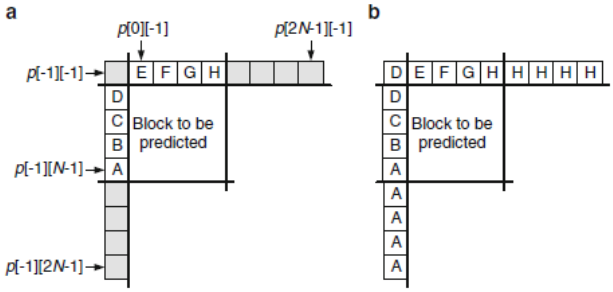
Svi modovi za predikciju koriste referentne uzorke koji su preuzeti iz ranije rekonstruisanih blokova. U toku DC predikcije posmatramo prosečne referentne uzorke koji se nalaze iznad i levo od bloka koji se prediktuje.
Uzimaju se konstantne vrednosti koje predstavljaju date prosečne referentne uzorke pomoću koji se određuje vrednost posmatranog uzorka.
Ugaona predikcija daje dobre rezultate kada radi sa jasnim ivicama, ali može stvoriti konture u glatkim oblastima.
DC predikcija takođe može stvoriti blokove u glatkim oblastima jer se dešava da se svi prediktovani uzorci zamene jednom vrednošću i tada na levoj i na gornjoj strani bloka nastaju diskontinuiteti. Problem ovih diskontinuiteta rešava planarna predikcija tako što se uzimaju prosečne vrednosti horizontalne i vertikalne predikcije. Na slici 3. prikazan je ovaj proces. Gornji-desni referentni uzorak p[N][-1] koristi se kao referenca za horizontalno filtriranje. Slično, donji-levi referentni uzorak p[-1][N] koristi se kao referenca za sve vertikalne operacije. Finalna predikcija vrednosti za svaki od uzoraka dobija se kada se uzme prosečna vrednost horizontalnih i vertikalnih predikcija [3].



Slika 3. *Primer planarne predikcije: (a) ilustruje kalkulaciju horizontalne komponente, (b) ilustruje generisanje vertikalne komponente i (c) daje primer finalne planarne predikcije dobijene uzimanjem vrednosti horizontalnih i vertikalnih komponenti [3].*

2.1 Generisanje referentnog uzorka

U intra predikciji, na osnovu moda predikcije vrši se proširenje vrednosti uzoraka pomoću grupe referentnih uzoraka. Može se dogoditi da pojedini ili svi referentni uzorci budu nedostupni za predikciju.
Za razliku od H.264, HEVC ima mogućnost zamene referentnog uzorka. Uzorci izvan slike, izvan „slice“ i izvan „tile“ se posmatraju kao nedostupni uzorci . Zatim, uzorci unutar date jedinice predikcije (PU) se tretiraju kao nedostupni, da ne bi došlo do propagacije greške. Naime, postoji mogućnost da su prethodno dekodovane slike pogrešno rekonsturisane. U slučaju kada nije dostupan nijedan referentni uzorak vrši se zamena svih referentnih uzoraka sa nominalnom prosečnom vrednošću uzoraka za datu dubinu bita. Na slici 4. vidimo proces zamene referentnog uzorka. Zamena se izvodi: pomoću prvog dostupnog referentnog uzorka, nedostupni uzorci oblika p[-1][y] gde je y = 2N-2 … -1, zamenjuju se sa referentnim uzorcima čije su vrednosti oblika p[-1][y+1] [17] i nedostupni uzorci oblika p[x][-1] gde je x = 0 … 2N-1, zamenjuju se referentnim uzorcima čije su vrednosti oblika p[x- 1][-1].



Slika 4. *Nedostupni referentni uzorci označeni su sivom bojom: (a) referentni uzorci pre procesa zamene (b) referentni uzorci posle procesa zamene [3].*

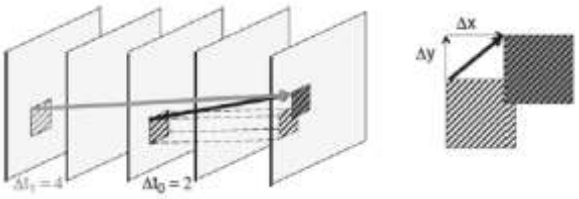
2.2 Filtriranje referetnih uzoraka

HEVC u intra predikciji izvodi uslovno filtriranje uzoraka primenom „smoothing“ filtra, da bi se izbegle skokovite promene vrednosti referentnih uzoraka koje bi mogle da stvore neželjene direkcione ivice na predikcijskom bloku.

Primena zavisi od veličine predikcijskog bloka i izabranog moda. Standardno se koristi 3-tap filtar [1 2 1]/4. Za blok veličine 32x32 se za generisanje referentnih uzoraka koristi linearna interpolacija između tri ugla referentnih uzoraka p[-1][63], p[-1][-1] i p[63][1].

3. INTER PREDIKCIJA

Inter predikcija se zasniva na vremenskoj povezanosti između slika. Izvodi se prediktor kompenzacije pokreta (MCP) za dati blok odbiraka (piksela). Ovo izvođenje se radi na osnovu prethodno dekodovanih blokova. koncept predikcije kompenzacije pokreta zasniva se na translaciji pokreta, prikazano na slici 5. Vektorom (Δx, Δy), gde su Δx i Δy horizontalni i vertikalni pomeraj u odnosu na položaj bloka koji posmatramo, označava se pozicija bloka u prethodno dekodovanoj slici. Radi preciznijeg modelovanja kretanja objekta (Δx,Δy) može se koristiti i necelobrojna (frakcionalna) preciznost. U tom slučaju, da bi izdvojili signal predikcije primenjuje se interpolacija na prethodno dekodovane slike.



Slika 5. *Koncept predikcije inter-slike i parametri koji koriste model translacionog kretanja [3].*

Pomoću prethodno dekodovanih slika pravimo listu referentnih slika, pri čemu svaka ima svoj referentni indeks Δt. Primenom modela translacije pokreta dobijaju se parametri koje nazivamo podaci pokreta. To su npr. vektori pokreta i referentni indeksi.

Imamo dve vrste inter predikcije: uni-predikcija i bi-predikcija. Kada koristimo bi-predikciju radi se sa dva seta podataka ($\Delta x_0, \Delta y_0, \Delta t_0$ i $\Delta x_1, \Delta y_1, \Delta t_1$). Na osnovu ova dva seta podataka formiramo dva MCP-a, koji najverovatnije potiču od različitih slika. Njihovom kombinacijom dobijamo finalni MCP, najčešće se uzima njihova prosečna vrednost. Pri primeni ponderisane predikcije možemo koristiti različite „težine" na svaki MCP. Na ovaj način možemo nadomestiti „izbledele" scene pomoću ponderisane predikcije. Pri primeni bi-predikcije koriste se dve odvojene liste (lista 0 i lista 1) za čuvanje referentnih slika. Bi-predikcija povećava digitalni protok i potrošnju memorije u „slice". HEVC ograničava potrošnju memorije tako što je propisao da PU koji sadrži luminentne predikcione blokove (PB) veličina 4x8 i 8x4 (mnogo blokova najmanje veličine) bude obrađivan samo uni-predikcijom.

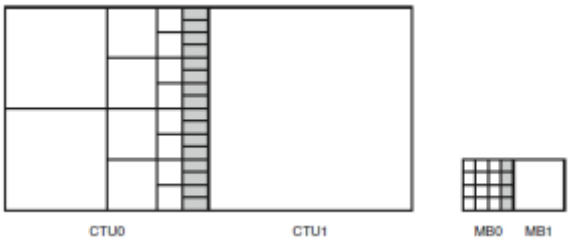
Estimacija pokreta koristi se u koderu radi dobijanja podataka pokreta. Nije definisana HEVC standardom. Ostavljeno je konstruktorima kodera da ostvare različite odnose kompleksnosti i kvaliteta u skladu sa njihovom primenom.

Korelacija podataka pokreta datog bloka sa podacima pokreta susednih blokova, omogućava da se podatak pokreta datog bloka prediktivno koduje na osnovu susednog podatka pokreta. Zato vektore pokreta u susednim blokovima koristimo kao prediktore.

3.1 Napredna predikcija vektora pokreta (AMVP)

U HEVC-u je unapređeno prediktivno kodovanje vektora pokreta uvođenjem napredne predikcije vektora pokreta (AMVP). AMVP obezbeđuje da najbolji prediktor svakog bloka bude signaliziran dekoderu. Prilagodljivo particionisanje koje se ostvaruje „quadtree" strukturom predstavlja unapređenje. Omogućava nam da koristimo velike blokove za oblasti slike u kojima nema detalja, npr. delovi slike na kojima se prikazuje nebo bez oblaka, dok za delove slike koji imaju puno detalja možemo izvršiti podelu na više malih blokova.

Međutim, pri primeni „quadtree" strukture mogu nastati i dva problema. Prvi je prikazan na slici 6. Vidimo da luminentni predikcioni blok veličine 64x64 nije podeljen i da sa leve strane, kao susedni blok ima blok sa najvećom mogućom podelom: 16 predikcionih blokova veličine 8x4. Znači da ima 16 kandidata za predikciju. HEVC je uveo AMVP da bi pojednostavio ovakav problem.

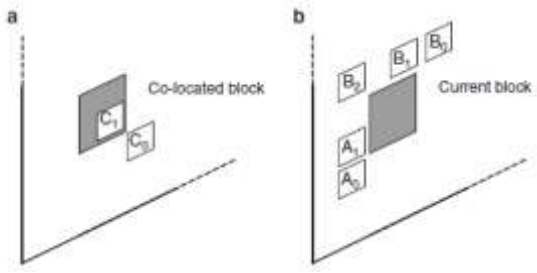


Slika 6. *Najgori scenario tj. maksimalan broj levih suseda u HEVC je prikazana na slici levo, na desnoj strani je najgori scenario u H.264 [3].*

3.1.1. Formiranje liste kandidata za AMVP

Izvođenje prediktora vektora pokreta (MVP) se zasniva na kompeticiji vektora pokreta. Prvo treba odrediti koji prediktori vektora pokreta će ići na listu kandidata. Na osnovu liste se izvodi vektor pokreta. Listu kandidata za AMVP čine dva prediktora vektora pokreta. Dobijaju se na sledeći način. Ako je dostupno 5 prostornih kandidata, onda se od njih izvode do dva prostorna kandidata.

Ako oba prostorna kandidata nisu dostupna ili su identična, onda se koriste dva vremenski kolocirana bloka: pomoću njih se izvodi jedan vremenski kandidat. Kada nisu dostupni prostorni, vremenski ili oba kandidata, imamo nula vektor. Na slici 7. prikazani su prostorni i vremenski kandidati. Pri traženju prostornog MVP kandidata posmatraju se samo prostorno susedni blokovi, sa leve i sa gornje strane posmatranog bloka. Blokovi desno i ispod posmatranog bloka se ne koriste jer nisu još uvek dekodovani. Vektor pokreta se skalira samo kada trenutna referentna slika i kandidat za referentnu sliku postoje u kratkom vremenskom trenutku. Kada tražimo vremenskog kandidata koristi se kolocirana slika, tj slika koja je već dekodovana.



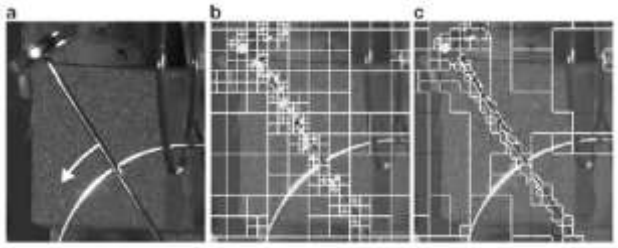
Slika 7. *Prediktor vektora pokreta i spajanje kandidata. (a) Vremensko. (b) Prostorno [3]*

Najbolji kandidati za traženje vremenskog prediktora vektora (TMVP) su C0 i C1. Prvo se traži podatak pokreta iz C0, pa ako nije dostupan, koristi se podatak pokreta iz C1. Skaliranje vektora pokreta obavezno je za TMVP. U

HEVC-u je moguće za svaku sliku navesti koja slika se posmatra kao kolocirana slika.

3.2 Spajanje blokova

Drugi problem koji nastaje primenom „quadtree“ particionisanja je nastanak neefikasnih granica blokova. Primer je prikazan na slici 8. Prikazano je klatno koje se kreće velikom brzinom ispred pozadine koja je „mirna“. U prirodi su česte ovakve scene. Dolazi do velikog „usitnjavanja“ oblasti oko klatna kako bi se „uhvatilo“ kretanje klatna. Kada se objekat relativno brzo kreće dolazi do naglih promena parametara pokreta i zato nastaje preterano particionisanje. Problem nastaje jer „quadtree“ struktura ne dozvoljava da se manji blokovi koji se nalaze u različitim velikim blokovima zajednički opisuju. Mogućnost da se blok podeli na četiri manja bloka, takođe može stvoriti problem neefikasnih granica.



Slika 8. *(a) Objekat koji se kreće (klatno), kretanje je označeno strelicom. (b) Stepen distorzije optimizovanog “quadtree” particionisanja. (c) Pokazane su samo granice koje odvajaju blokove sa različitim parametrima pokreta [4].*

HEVC tehnikom spajanja blokova rešava problem neefikasnih granica. Ova tehnika omogućava da se kodovanje pokreta izvrši jednostavnije sa smanjenim protokom bitova. Pomoću tehnike spajanja blokova, za dati sub-blok možemo upotrebiti parametre pokreta koje koriste susedni blokovi. Funkcioniše kao AMVP.

3.2.1. Konstrukcija liste spojenih kandidata

U AMVP postoji jedna referetna lista kandidata (koriste se samo vektori pokreta). Tehnika spojenih kandidata ima 1 ili 2 liste referentnih kandidata (koristi se vektor pokreta za svaku sliku i referentni indeks). Lista spojenih kandidata sadrži: do 4 prostorna kandidata (izvode se od 5 kandidata), 1 vremenski kandidat (izvodi se od dva vremenski kolocirana bloka). Mogu se koristiti i dodatni spojeni kandidati.

Prostorni kandidati

Posmatraju se isti prostorni susedi kako kod AMVP-a. Prvo se proverava da li je susedni blok dostupan i da li sadrži podatke pokreta. Zatim se vrši provera redudantnosti koja ima dve namene. Prva je izbegavanje da na listi budu kandidati sa redudantnim podatkom pokreta. Druga je da spreči spajanje dve particije koje bi se mogle izraziti i na još neki način.

Vremenski kandidat

Izvođenje je isto kao kod TMVP. Provera redudantnosti se ne vrši.

Dodatni kandidat

Lista spojenih kandidata ima fiksnu dužinu. Ako je nismo popunili sa prostornim i vremenskim kandidatima, onda koristimo dodatne kandidate.

To su kombinovani bi-predikcioni kandidati i kandidati nula vektora pokreta. Referentni indeks se inkrementira za svakog dodatnog kandidata sve dok se ne dostigne maksimalni dozvoljeni referentni indeks.

4. ZAKLJUČAK

Predikcija u HEVC-u nije donela revolucionarni skok u razvoju predikcije digitalnih TV sistema. Ona predstavlja očekivani napredak u odnosu na predikcije u prethodnom standardu H.264/AVC. Tako su poboljšane tehnike korišćene u prethodnim standardima i uvedene neke nove. Povećana je maksimalna veličina bloka za intra predikciju na 32x32, dok je u H.264/AVC bila 16x16. HEVC ima 33 moda za obradu direkcionih struktura, dok H.264/AVC ima 8 modova.

U predikciji hrominentnih i luminentnih blokova HEVC primenjuje sve modove, a H.264/AVC za predikciju hrominentnog bloka ima dva moda. HEVC kontinuitet na granicama blokova obezbeđuje primenom unapređenog planarnog moda i primenom naknadnog filtriranja. Mogu se nadomestiti svi nedostupni referentni odbirci (pikseli). Uvođenjem AMVP je inter predikcija unapređena. TMVP donosi poboljšanje. Tehnika spajanja je olakšala signalizaciju podataka pokreta. Pored navedenih tehnika, HEVC donosi još tehnika koje su unapredile predikciju. Npr. izvođenje najverovatnijeg moda za predikciju luminentne komponente. Izabrani mod se može primeniti i na hrominentnu komponentu. Uvedeni su regioni estimacije. Unapređena je frakciona interpolacija a uvedena su i još mnoga druga poboljšanja.

5. LITERATURA

[1] Lainema J, Bossen F, Han W-J, Min J, Ugur K (2012) Intra coding of the HEVC standard. IEEE Trans Circuits Syst Video Technol 22(12):1792–1801

[2] Schwarz H, Wiegand T, (2016) Video Coding: Part II of Fundamentals of Source and Video Coding

[3] Vivienne S,Budagavi M, Sullivan G (Eds) (2014) High Efficiency Video Coding (HEVC): Algorithms and Architectures

[4] Helle P, Oudin S, Bross B,Marpe D, Bici M, Ugur K, Jung J, Clare G,Wiegand T (2012) Block merging for quadtree-based partitioning in HEVC. IEEE Trans Circuits Syst Video Technol 22(12):1720–1731

Kratka biografija:

Ivan Repić rođen je u Prizrenu 1977. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Telekomunikacije odbranio je 2018.god.

Željen Trpovski rođen je u Rijeci 1957. godine. Doktorirao je na Fakultetu tehničkih nauka 1998. god. Oblast interesovanja su telekomunikacije i obrada signala.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.31
DOI: https://doi.org/10.24867/01BE02Jokic

OPTIMALNA REKONFIGURACIJA DISTRIBUTIVNIH MREŽA

OPTIMAL NETWORK RECONFIGURATION OF DISTRIBUTION SYSTEMS

Ivana Jokić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu su razmatrana dva osnovna algoritma za optimalnu rekonfiguraciju distributivnih mreža: algoritam izmene grana i algoritam najmanjih struja. Verifikacija njihovih mogućnosti realizovana je na primeru test distributivne mreže koja se sastoji od 25 čvorova.*

Abstract – *This paper presents two basic algorithms for optimal network reconfiguration of distribution network systems: Branch Exchange algorithm and Minimal branch current algorithm. Verification of their possibilities is realized on the test case of a distribution network consisting of 25 nodes.*

Ključne reči: *optimalna rekonfiguracija, distributivne mreže, metod izmene grana, metod najmanjih struja*

1. UVOD

Većina distributivnih mreža (DM) se projektuje u vidu radijalnih mreža, a sve u cilju postizanja efikasnije koordinacije zaštite pojedinih elemenata mreže [1]. U prisustvu kvara, prekidačkom opremom se može manipulisati u cilju izolovanja dela mreže sa kvarom i prebacivanja izolovane potrošnje na zdrav deo mreže. U normalnim uslovima, ovi prekidači se mogu koristiti za rekonfiguraciju, u cilju smanjenja gubitaka aktivne snage, povećanja pouzdanosti ili balansiranja opterećenja u mreži [2]. Isporuka električne energije od izvora do krajnjih potrošača je uvek praćena gubicima. Ako se u obzir uzme da su distributivna preduzeća suočena sa povećanim pritiskom za ostvarivanje veće efikasnosti, da su uvedeni podsticaji i penali koji elektrodistribucija plaća potrošačima zbog nenajavljenih prekida napajanja ili lošeg kvaliteta isporučene električne energije, i da gubici direktno utiču na finansijska pitanja i efikasnost distributivnih postrojenja, jasno je da redukcija gubitaka predstavlja cilj svakog distributivnog preduzeća [3]. Jedan od efikasnih načina za postizanje ovog cilja jeste optimalna rekonfiguracija distributivnih mreža (ORDM).

2. PRORAČUN TOKOVA SNAGA

Proračun tokova snaga predstavlja proračun promenjivih stanja (režima) mreže, gde su poznati izvor napajanja mreže i potrošnje u svim čvorovima mreže. Na osnovu vrednosti modula i faznih stavova napona proračunavaju se preostale vrednosti koje nisu unapred zadate, a to su aktivne i reaktivne snage injektiranja u čvorovima.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Goran Švenda, red.prof.

Na ovaj način definisano je stanje mreže. Dalje, vrši se proračun tokova snaga po vodovima i određivanje gubitaka po granama. S toga, metode koje se koriste u DM razlikuju se od onih koje se koriste u prenosnim mrežama. Većina poznatih iterativnih postupaka koji se koriste u prenosnim mrežama bazirani su na matričnom postupku, dok se u DM koriste specijalizovani algoritmi koji su orijentisani ka granama [1].

2.1. Modelovanje potrošača

Fazna aktivna i reaktivna snaga potrošnje potrošača priključenog u čvoru, obično se iskazuju preko tri komponente [5]:

1) konstantna snaga (nezavisna od modula napona),
2) konstantan modul struje i faktor snage (snaga srazmerna s modulom napona),
3) konstantna impedansa tj. admitansa (snaga srazmerna kvadratu modula napona).

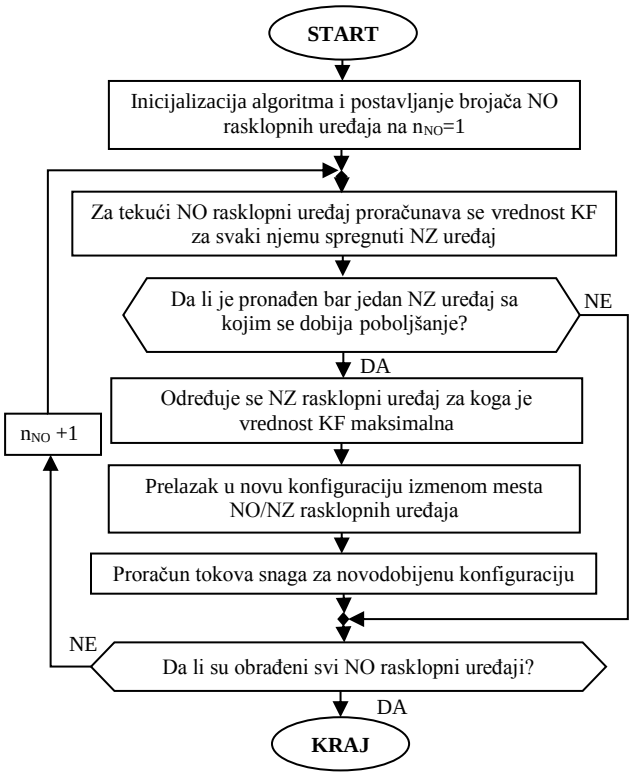
U zavisnosti od modela potrošnje, vrednosti režima EES-a se mogu značajno menjati.

3. OPTIMALNA REKONFIGURACIJA DM

Problem određivanja optimalne konfiguracije mreže je kompleksan, kombinatoran, nelinearan i diskretan optimizacioni problem. Za rešenje ovog problema koriste se: optimizacione metode [1,4], kombinatorno pretraživanje [1] i heurističke metode [1,6]. Izvorni algoritmi vezani su za metode bazirane na heuristici i optimalnim tokovima snaga. Suština ovih algoritama je da se maksimalnim poznavanjem fizike problema na najkraći način dođe do kvalitetnih radijalnih konfiguracija. Postoje dva problema vezana za algoritme bazirane na heuristici: u opštem slučaju, ne obezbeđuju globalni optimum i postizanje optimalne konfiguracije kroz višestruko puštanje tokova snaga može biti vremenski veoma zahtevno. Heuristički algoritmi kojima se rešava problem ORDM su algoritmi izmene grana i najmanjih struja [1].

3.1. Algoritam izmene grana

"Izmena grana" zapravo znači "izmena mesta" normalno otvorenog (NO) rasklopnog uređaja sa njime spregnutim normalno zatvorenim (NZ) rasklopnim uređajem [4]. Globalni blok algoritam prikazan je na slici 1. Glavna prednost ovog algoritma jeste izvršavanje tokova snaga samo u slučaju kada je postignuto poboljšanje vrednosti kriterijumske funkcije. Osnovna mana ovog algoritma jeste zavisnost konačnog od početnog rešenja. Kako je u svakom trenutku proračuna radijalnost uvek održana, mogu se primenjivati metode za proračun tokova snaga orijentisane ka granama. U ovom radu izabran je algoritam sumiranja struja -Shirmohammadi-ev algoritam [1].



Slika 1 – Blok dijagram algoritma izmene grana [1]

Kao kriterijum, odabrana je minimizacija gubitaka aktivne snage. Kriterijumska funkcija (KF) za procenu smanjenja gubitaka aktivne snage nakon izmene mesta NO/NZ prekidača, može da se predstavi kao [3]:

$$IG^{(h)} = \sum_{i=1}^{n_{izv}} \sum_{j \in \alpha_{d_i}^h} r_{ij} (J_{ij}^h)^2 \ , \qquad (1)$$

gde je:

n_{izv} – ukupan broj izvoda,

r_{ij} – rezistansa grane *j* koja pripada izvodu *i*,

$\alpha_{d_i}^h$ – skup indeksa grana izvoda *i* u konfiguraciji *h*,

J_{ij}^h – moduo struje grane *j* izvoda *i* u konfiguraciji *h*.

Ako je sa supskriptom *h* označena konfiguracija nakon izmene mesta NO/NZ rasklopnih uređaja, a sa *h*–1 konfiguracija pre te izmene, vrednost kriterijuma je [1]:

$$IG^{(h)} = IG^{(h-1)} - \Delta IG \qquad (2)$$

$$\Delta IG^{(h-1)} = 2 \cdot \mathrm{Re}\left[(J_j^{h-1})^* (\Delta V_k^{h-1} - \Delta V_m^{h-1}) \right] - \left| J_j^{h-1} \right|^2 \cdot R_{petlje} \qquad (3)$$

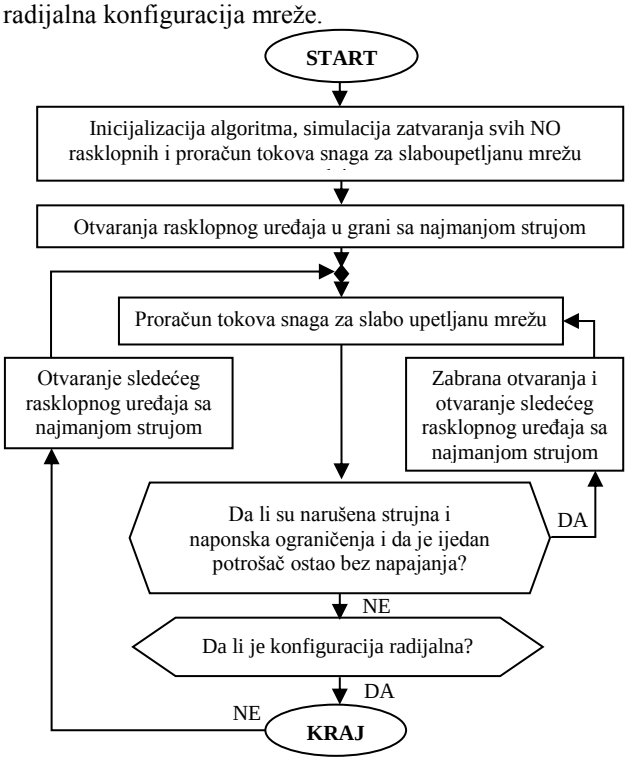
gde su sa $IG^{(h)}$, $IG^{(h-1)}$ *i* ΔIG naznačene *vrednosti gubitaka* aktivne snaga za konfiguracije *h* i *h*–1 i promena te vrednosti, respektivno; sa J_j^{h-1} struja grane *j* u radijalnoj konfiguraciji *h*–1; sa ΔV_k^{h-1} i ΔV_m^{h-1} vrednosti padova napona čvorova levo i desno od razmatranog NO rasklopnog uređaja i sa R_{petlje} rezistansa konture između dva napojna čvora koja sadrže granu sa NO rasklopnim uređajem.

Pri izvođenju algoritma izmene grana dovoljno je proračunati samo vrednost izraza (1). Ako je ta vrednost veća od nule, tada su izmenom mesta razmatranog spregnutog para NO/NZ rasklopnih uređaja gubici aktivne snage smanjeni.

3.2. Algoritam najmanjih struja

Algoritam najmanjih struja je efikasan i robustan heuristički metod koji za razliku od algoritma izmene grana pronalazi optimalno rešenje koje je u potpunosti nezavisno od inicijalnog statusa rasklopnih uređaja [7]. U algoritmu najmanjih struja korišćen je Newton/Raphsonov iterativni postupak, pogodan za upetljane i slaboupetljane mreže. Mana ovog algoritma jeste relativno veliki broj proračuna tokova snaga (jednak broju NO rasklopnih uređaja u mreži).

Globalni blok dijagram algoritma najmanjih struja dat je na slici 2. Na početku algoritma učitavaju se svi podaci o mreži. Nakon toga vrši se simulacija zatvaranja svih NO rasklopnih uređaja čime se ostvaruje maksimalno upetljan pogon. Za tako upetljanu mrežu proračunavaju se tokovi snaga da bi se dobio celokupan režim. Na osnovu rezultata pronalazi se grana sa najmanjom vrednošću struje i rasklopni uređaj u toj grani se otvara, ali samo ako nisu narušena strujna i naponska ograničenja, i ukoliko neće ostaviti potrošače bez napajanja. U suprotnom, traži se sledeća grana sa najmanjom strujom, za koju su zadovoljena navedena ograničenja. Za takvu topologiju mreže vrši se proračun tokova snaga čiji rezultati predstavljaju ulazne podatke za sledeću iteraciju proračuna. Algoritam se zaustavlja kada je postignuta radijalna konfiguracija mreže.



Slika 2 – Blok dijagram algoritma najmanjih struja [1]

4. VERIFIKACIJA ALGORITAMA

Verifikacija prikazanih algoritama optimalne rekonfiguracije DM izvršena je na primeru test DM sa 25 potrošačkih čvorova. Ti potrošači se nalaze na 5 izvoda koji se napajaju sa zajedničkih SN sabirnica Tr VN/SN. Ukupna potrošnja svih potrošača u mreži, pri nominalnom naponu na sabirnicama njihovog priključenja, je (12.00–9.00) MVA. U nastavku razmatrana su 4 primera, sa različitom početnom topologijom i raspodelom ukupnog opterećenja. Istovremeno, razmatran je i uticaj modela potrošnje na

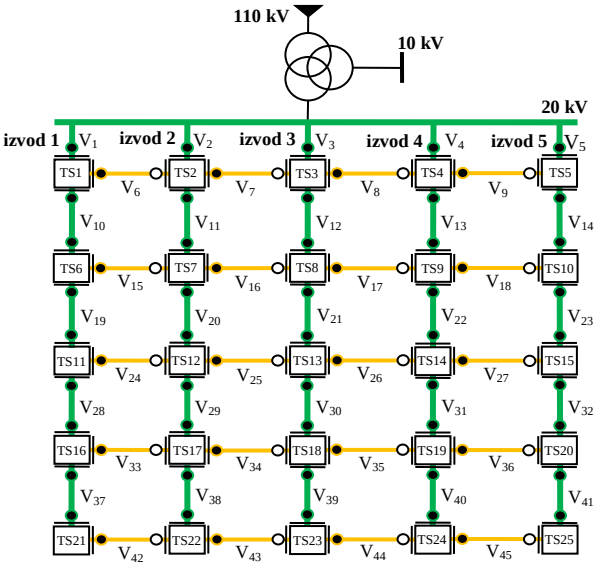
rezultate proračuna tokova snaga (tri modela potrošnje za nominalnu vrednost napona ostvaruju istu vrednost aktivne i reaktivne snage, isti faktor snage, istu struju i istu admitansu).

4.1. Primer 1

Ovo je osnovni primer i njime se vrši osnovna verifikacija proračuna. Sa svakog od pet izvoda napaja se po pet međusobno jednakih potrošača, čija je snaga 0.48 MW i 0.36 Mvar, pri $U=U_{nom}$.

4.1.1. Primena algoritma izmene grana

Nakon primene algoritma izmene grana kriterijumska funkcija u svim iteracijama ima negativnu vrednost, što znači da se promenom uklopnog stanja ne može ostvariti benefit, odnosno da konačno, optimalno rešenje odgovara početnom rešenju, topologija sa slike 3.



Slika 3 – Test DM, inicijalno stanje – Primer 1

4.1.2. Primena algoritma najmanjih struja

Nakon zatvaranja svih NO rasklopnih uređaja, u svakoj iteraciji otvarana je po jedna od grana koje su bile otvorene u početnom stanju, tako da konačno rešenje odgovara početnoj topologiji, slika 3.

4.1.3. Uticaj modela potrošača na tokove snaga

U zavisnosti od modela potrošnje: konstantna snaga, konstantni modul struje i faktor snage, ili kao konstantna admitansa (impedansa), u tako proračuna tokova snaga neke od veličina su konstantne a neke promenljive. U zavisnosti od modela u tabeli 1 su sa "+" naznačene veličine čije su vrednosti konstantna, a sa "–" veličine čije se vrednosti menjaju u toku proračuna tokova snaga.

Tabela 1 – Uticaj modela potrošača

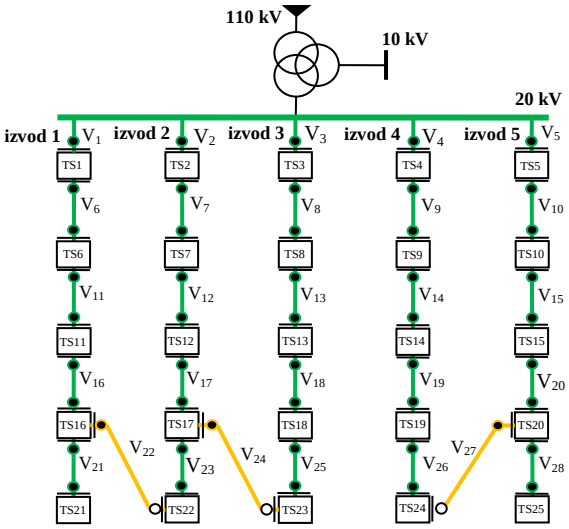
Tip potr.	P_p	Q_p	$\cos\varphi_p$	I_p	$\hat{I}_p$	$\hat{Y}_p$
1	+	+	+	–	–	–
2	–	–	+	+	–	–
3	–	–	+	–	–	+

4.2. Primer 2

Obrađuje se test DM sa 3 NO rasklopna uređaja, slika 4. Ukupno opterećenje 12.00 MW i 9.00 MVar slučajno je raspoređena po čvorovima, tabela 2.

Tabela 2 – Opterećenje: MW, MVar – Primer 2

čv.	P	Q	čv.	P	Q	čv.	P	Q
1	0.40	0.30	10	0.40	0.30	19	0.56	0.42
2	0.64	0.48	11	0.40	0.30	20	0.32	0.24
3	0.72	0.54	12	0.64	0.48	21	0.40	0.30
4	0.56	0.42	13	0.72	0.54	22	0.24	0.18
5	0.40	0.30	14	0.56	0.42	23	0.16	0.12
6	0.40	0.30	15	0.40	0.30	24	0.16	0.12
7	0.64	0.48	16	0.40	0.30	25	0.32	0.24
8	0.72	0.54	17	0.64	0.48			
9	0.56	0.42	18	0.64	0.48			



Slika 4 – Test DM, inicijalno stanje – Primer 2

4.2.1. Primena algoritma izmene grana

Razmatrana su dva pristupa obrade NO rasklopnih uređaja: 1) IG-1 – od grane sa najmanjim indeksom do grane sa najvećim indeksom i 2) IG-2 – suprotno. Za prvi pristup gubici aktivne snage su smanjeni za 2.4%, a u drugom za 1.9%. Za dva različita pristupa, ostvarena su dva različita optimuma, što ukazuje na zavisnost rešenja od redosleda obrade NO rasklopnih uređaja. Slična rešenja se dobijaju za sva tri tipa potrošača.

4.2.2. Primena algoritma najmanjih struja

Primenom algoritma najmanjih struja (NS) dobijeno je isto rešenje kao rešenje koje je dobijeno primenom prvog pristupa algoritma izmene grana. Ovo rešenje je jedinstveno. Isto rešenje dobija se za sva tri tipa potrošača.

4.3. Primer 3

Početna topologija prikazana je na slici 4. Sva opterećenja odgovaraju vrednostima koje su prikazane u tabeli 2, osim opterećenja u čvorovima TS21 i TS23. Njihove vrednosti su (0.404–j0.304) MVA i (0.234–j0.176) MVA, respektivno.

4.3.1. Primena algoritma izmene grana

Sada se posmatra samo redosled obrade NO rasklopnih uređaja od onog u grani sa najmanjim ka onom u grani sa najvećim indeksom. Nakon izvršenog proračuna za potrošače konstantne snage ostvareno je smanjenje gubitaka aktivne snage za 1.3%. Iste izmene dobijene su i kada su potrošači modelovani kao konstantna vrednost modula struje i faktora snage, ali su zapažene drugačije

vrednosti KF. U slučaju potrošača konstantne admitanse dobijeni su drugačiji rezultati uz ostvaren benefit od 1.9%. Ako se posmatra metoda „izmene grana“, varijacija vrednosti KF u zavisnosti od načina modelovanja potrošača nije velika. Stoga ukoliko KF ima veliku negativnu vrednost za jedan tip potrošnje, mala je verovatnoća da će u slučaju nekog drugog tipa posmatrana izmena biti usvojena. Ali ukoliko je vrednost KF blizu graničnoj vrednosti 0, postoji veća mogućnost da usled drugačijeg modelovanja potrošnje posmatrana izmena bude usvojena. Naravno, to utiče na raspodelu tokova snaga, i u nastavku algoritma može dovesti do usvajanja novih „izmena mesta“ ili odbacivanja nekih izmena koje su bile usvojene u slučaju drugog modela potrošača. Može se reći da model potrošača indirektno utiče na metodologiju „izmene grana“, tako što drugačije vrednosti napona i struja dobijene proračunom tokova snaga utiču na vrednosti KF.

4.3.2. Primena algoritma najmanjih struja

Primenom algoritma najmanjih struja (NS), za model potrošača konstantna snaga i konstantna admitansa podudaraju se sa rezultatima dobijenim metodom izmene grana za slučaj model potrošača konstantna admitansa. Kada se potrošnja modeluje kao konstantni modul struje i faktor snage, rezultati su nešto drugačiji, ali benefit ima približno istu vrednost. Za metodu NS uočeno je da model potrošača u svakoj iteraciji proračuna tokova snaga direktno utiče na izbor otvaranja rasklopnih uređaja, odnosno direktno utiče na krajnje rešenje metode.

4.4. Primer 4

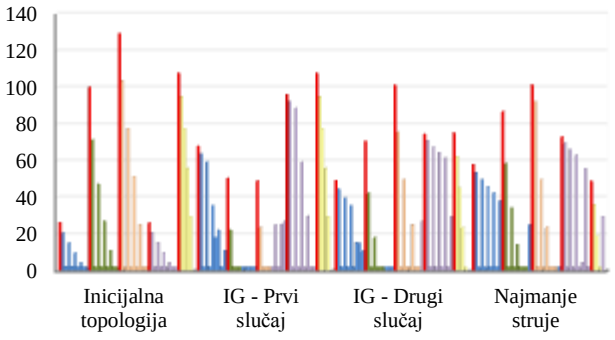
Topologija test DM odgovara onoj iz prvog primera, slika 3. Ukupno opterećenje je i dalje isto, ali je slučajno raspoređeno po čvorovima, u skladu sa tabelom 2.

4.4.1. Primena algoritma izmene grana

Kao i u prethodnim primerima, pravila obrade NO rasklopnih uređaja su ista. Za prvi slučaj IG-1 ostvaren je benefit 5.9%, a u drugom slučaju IG-2 benefit od 19.4%.

4.4.2. Primena algoritma najmanjih struja

Primenom algoritma najmanjih struja (NS) gubici aktivne snage su smanjeni za 17.3%. U poređenju sa vrednostima koji su ostvareni primenom algoritma izmene grana, sa kojim je dobijeno i loše (IG-1) i dobro (IG-2) rešenje, algoritam NS daje jedinstveno, dobro rešenje. Na slici 6 prikazane su vrednosti struja po granam za početnu topologiju i topologije dobijene primenom tri razmatrana pristupa za rešavanje ORDM problema.



Slika 7 – Raspodela opterećenja za sve mreže – primer 4

U tabeli 3 istaknute su vrednosti struja grana od interesa, na osnovu kojih se mogu potvrditi ostvareni benefiti.

Tabela 3 – Raspodela struja u granama od interesa

	Iv_1 [A]	Iv_2[A]	Iv_3 [A]	Iv_4 [A]	Iv_5 [A]
Počet. topol	27.21	100.97	130.31	27.21	108.65
IG-1	69.01	51.46	50.05	96.95	108.65
IG-2	50.13	71.69	102.25	75.58	76.08
NS	58.80	87.81	102.24	74.10	50.00

Kao rezime, u tabeli 4 prikazane su vrednosti benefita koji su ostvareni u razmatranim primerima, kao i broj proračuna tokova snaga.

Tabela 4 – Benefit i broj proračuna tokova snaga

	Primer 2		Primer 4	
	B [%]	br. TS	B [%]	br. TS
IG-1	2.38	4	5.91	7
IG-2	1.49	3	19.42	5
NS	2.38	4	17.32	21

5. ZAKLJUČAK

Algoritam najmanjih struja daje uvek isto rešenje, dok je algoritam izmene grana osetljiv na početno stanje. Broj proračuna tokova snaga kod algoritma najmanjih struja jednak je broju NO rasklopnih uređaja, dok je kod algoritma izmene grana taj broj znatno manji.

6. LITERATURA

[1] D.Popović, D.Bekut, V.Dabić: *Specijalizovani DMS Algoritmi*; Prosveta, Novi Sad, 2011.

[2] D.Shirmohammadi, H.W.Hong: Reconfiguration of Electric Distribution Networks for Resistive Line Losses Reduction, *IEEE Transactions on Power Delivery*, Vol. 4, No. 2, April 1989, pp 1492-1498

[3] Y.Al-Mahrogi, I.A.Metwally, A.Al-Hinai, A.Al-Badi: Reduction of Power Losses in Distribution Systems; *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, Vol:6, No:3, 2012.

[4] M.F.Sulaima, M.F.Mohamad, M.H.Jali, W.M.Bukhari, M.F.Baharom: A Comparative Study of Optimization Methods for 33kV Distribution Network Feeder Reconfiguration; *International Journal of Applied Engineering Research*, Vol. 6, No. 4, 2014, pp. 1169-1182

[5] V.C.Strezoski: *Osnovi Elektroenergetike*; Fakultet tehničkih nauka u Novom Sadu, Novi Sad, 2014.

[6] A.Charlangsut, N.Rugthaicharoencheep, S.Auchariyament: Heuristic Optimization Techniques for Network Reconfiguration in Distribution System; *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, Vol. 6, No. 4, 2012, pp. 365-368

[7] B.Cho, K.Ryu, J.Park, W.Moon, S.Cho, J.Kim: A Selecting Method of Optimal Load on Time Varying Distribution Systems for Network Reconfiguration considering DG; *Journal of International Council on Electrical Engineering*, Vol. 2, No. 2, pp. 166-170, 2012.

Kratka biografija:

Ivana Jokić rođena je u Šapcu 27.06.1992. god. Srednju školu, završila je 2011. god. u Šapcu. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Elektroenergetski sistemi odbranila je 2015. god. Iste godine upisala se na master studije.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/01BE03Pavlovic

DEFINISANJE SKUPA PODATAKA ZA VIZUELIZACIJU DISTRIBUTIVNE MREŽE

DEFINING THE SET OF DATA NEEDED TO VISUALIZE DISTRIBUTION NETWORK

Sara Pavlović, Fakultet tehničkih nauka, Novi Sad

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Savremenu distributivnu mrežu čini neizmeran broj entiteta, pa je njena sveobuhvatna vizuelizacija nepraktična za nadzor i upravljanje. Rješenje ovog problema je u prikazu samo važnih entiteta, tj. entiteta koji su od značaja korisniku, dok se preostali entiteti grupišu u cjeline. Da bi se postigao cilj, iskorišćena je radijalna struktura distributivne mreže. Mreža je opisana pomoću grafa i upotrebom Primovog algoritma kreirano je minimalno razapinjuće stablo koje obuhvata važne entitete mreže. Potom se stablo dijeli na odgovarajuće segmente, koji se šalju na prikaz. Rezultat ovakvog pristupa je jasna vizuelizacija distributivne mreže.*

Ključne reči: *Distributivna mreža, Vizuelizacija mreže, Minimalno razapinjuće stablo, Primov algoritam*

Abstract – *A modern distribution network contains an immense number of entities, so its overall visualization is impractical for supervision and control. Solution lies in the concept of showing only the most important entities of the network to a user: the ones that "carry" the most significant information. In order to achieve the goal, the radial structure of the distribution network is exploited. The distribution network is presented in a form of a graph, and by utilizing the Prim's Algorithm, a minimum spanning tree which includes the important entities of the grid is obtained,. Afterwards, the minimum spanning tree is partitioned into segments, which are further forwarded for displaying.*

Keywords: *Distrubution network, Network visualisation, Minimum spanning tree, Prim's Algorithm*

1. UVOD

Dostignuća u polju informacionih i komunikacionih tehnologija su omogućila razvoj pametnih elektroenergetskih sistema [1]. Distributivni menadžment sistem (DMS) [2] je softverska komponenta u okviru pametnog elektroenergetskog sistema zadužena za manipulaciju podsistemom distribucije. Sa druge strane, distribucija elek. energije zasniva se na konceptu izvoda [3], elektroenergetskih veze koje napajaju dijelove potrošačkih područja.

S obzirom da savremenu distributivnu mrežu odlikuje milionski broj entiteta, njen sveobuhvatni prikaz je opterećen suvišnim detaljima, nepraktičan i neprikladan za nadzor i upravljanje. Vizuelizacija zasnovana samo na entitetima koji su od interesa korisniku daje jasnu predstavu stanja mreže i olakšava njenu kontrolu. Cilj ovog rada je da definiše i objasni implementaciju algoritma kojim se određuje skup podataka za prikaz distributivne mreže.

2. MODEL DISTRIBUTIVNE MREŽE

Kako bi efikasno izvršavao zadatke DMS se oslanja na opis (model) mreže kojom upravlja. Takav model se najčešće zasniva na dobro provjerenom standardu koji propisuje strukturu modela. Najpopularniji standard je CIM (*Common Information Model*) [4].

Ovo istraživanja je bazirano na ideji da se oprema nebitna korisniku prikaže kao jedinstvena cjelina umjesto kao pojedinačni entiteti koji opterećuju vizuelizaciju distributivne mreže, Iz tog razloga od interesa su entiteti CIM-a koji opisuju provodnu opremu (*ConductingEquipment*), konektivnost između nje (*ConnectivityNode)*, kao i grupisanje opreme u vidu kontejnera (*EquipmentContainer*). Takođe je iskorišćena mogućnost proširenja CIM-a za modelovanje izvoda distributivne mreže (*Feeder*), indikaciju koliko je oprema ili čvor mreže važan za korisnika (atribut *importance*), kao i grupisanje opreme koja se vizuelno prikazuje kao jedinstvena cjelina nazvana segment izvoda (*FeederSegment*).

3. MODELOVANJE DISTRIBUTIVNE MREŽE PREKO GRAFA

Rješenje problema koristi radijalnu prirodu distributivne mreže [3] kako bi uprostio detaljan model zasnovan na CIM-u. Entiteti i njihova konektivnost se predstavljaju grafom (tačnije strukturom tipa stablo) [5]. Nad novom organizacijom podataka primjenjuju se algoritmi za manipulaciju grafovima modifikovani u skladu sa potrebama ovog rada.

Prvi korak je modelovanje čvorova grafa. Čvorovi elektrodistributivne mreže, objekti tipa *Connectivity Node* u CIM-u, su ujedno i čvorovi grafa. Oprema u distributivnoj mreži (dvokrajnici, jednokrajnici i transformatori) služi za modelovanje grana grafa. Dodatno, svakoj grani i svakom čvoru je dodijeljena indikacija važnosti na osnovu *importance* atributa iz originalnog modela. Ovako predstavljena distributivna mreža omogućava jednostavnu identifikaciju važne opreme i važnih čvorova kao i putanja između njih.

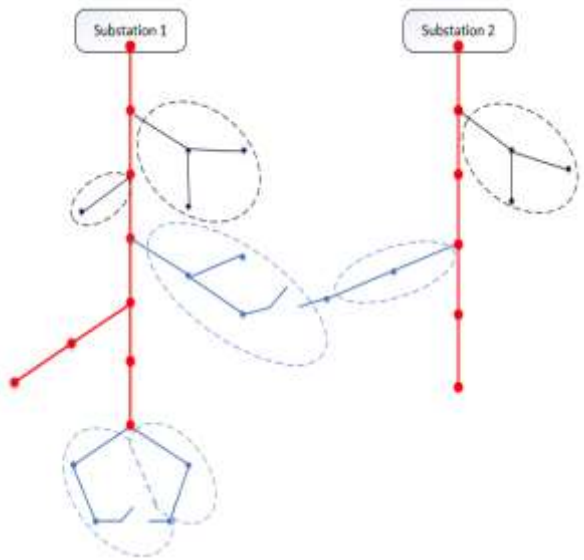
2. INICIJALNO RJEŠENJE

Segment izvoda definiše grupu opreme koja treba da se prikaže kao jedna cjelina. Podjela izvoda na segmente omogućuje prikaz samo onih elemenata koji su od značaja korisniku. Drugim riječima, ukoliko se korisnik odluči na

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Nemanja Nedić, docent.

prikaz preko segmenata dobija pojednostavljeni prikaz, što mu omogućuje fokusiranje na najvažnije entitete mreže.

Osnovni tipovi segmenta u ovom radu su definisani kao: *trunk*, *lateral* i *loop*. *Trunk* je glavna trofazna grana izvoda. Pored toga, segmentima tipa *trunk* su obuhvaćeni vodovi koji dosežu do entiteta (opreme ili *ConnectivityNode*-ova) proglašenih za važne od strane korisnika. *Lateral* je tip segmenta koji se nadovezuje na *trunk* i sadrži opremu bez petlji koja nije od važnosti za korisnika. U pojedinim slučajevima, distributivna mreža ima upetljani dio, radi obezbjeđivanja sigurnosti u napajanju potrošača iz više izvora ukoliko dođe do ispada dijela mreže usljed kvara. Takvi segmenti mreže čine segmente izvoda tipa *loop*. *Loop* segmenti mogu biti između dva povezana izvoda ili unutar jednog. Na slici 2.1 su prikazana dva susjedna izvoda. Crvenom bojom su označene glavne grane izvoda, kao i bočni vodovi koji su od interesa operateru. Crnom bojom su predstavljeni nevažni entiteti koji nisu uključeni u petlje, dok su plavom bojom označeni entiteti koji su dio petlje.



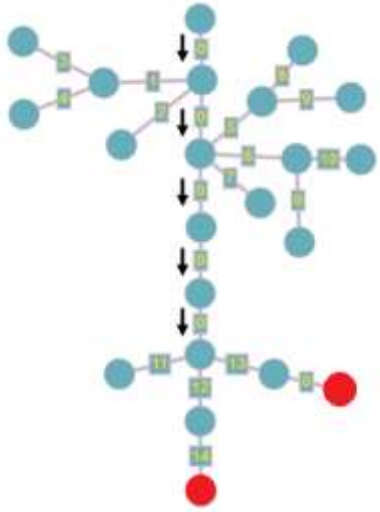
Slika 2.1. *Grafički prikaz segmenata izvoda*

2.1. Detekcija povezanosti važnih entiteta

Svi važni entiteti, uz izuzetak električnih vodova, treba da se prikažu individualno. Važni električni vodovi koji se nadovezuju jedan na drugi mogu da se prikažu kao jedan segment. Iz tog razloga, prvenstveno je neophodno detektovati važne entitete i putanje između njih. U tu svrhu je prilagođen koncept minmalnog razapinjujećeg stabla [5] koje će obuhvatiti sve važne grane i važne čvorove stabla.

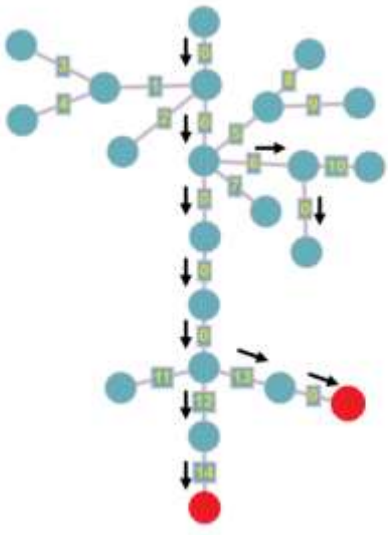
S obzirom da graf kojim se modeluje izvod nije težinski, a Primov algoritam [6] za određivanje razapinjujećeg stabla radi s težinskim grafovima, potrebno je dodijeliti težine granama grafa. Svim važnim granama se dodjeljuje težina 0. Za dodjeljivanje težina nevažnim granama je korišten *Breadth First Search* (BFS) algoritam [6].

Nakon što je svakoj grani dodijeljena težina, slijedi formiranje "baze" minimalnog razapinjućeg stabla, koju čine samo važne grane, do kojih se može doprijeti krećući se Primovim algoritmom od početnog čvora. Na slici 2.2 su strelicama označene grane grafa koje čine bazu minimalnog razapinjućeg stabla.



Slika 2.2. *Baza minimalnog razapinjućeg stabla*

Važni čvorovi i važne grane koji nisu obuhvaćeni bazom minimalnog razapinjućeg stabla se nazivaju viseći. Neophodno je sve viseće čvorove i grane povezati sa bazom, kako bi se u cjelosti formiralo minimalno razapinjuće stablo. Za svaki viseći element pojedinačno se poziva Primov algoritam, gdje se kao početni čvor postavi baš taj viseći element. Obzirom da grane bliže početnom čvoru imaju manju težinu, Primov algoritam će se uvijek kretati prema bazi minimalnog razapinjućeg stabla. Algoritam se zaustavlja kada se dođe do čvora koji pripada minimalnom razapinjućem stablu. Na slici 2.3 je prikazan primjer grafa izvoda i minimalnog razapinjućeg stabla.



Slika 2.2. *Graf izvoda i minimalno razapinjuće stablo*

2.2. Region povezanih izvoda

Svaka promjena u izgrađenosti ili normalnom uklopnom stanju mreže utiče na jedan ili više izvoda, a time i na njihove segmente. Skup izvoda koji su direktno ili posredno povezani sa izvodima i koji su pod uticajem promjena, čine region povezanih izvoda.

S obzirom da se kreiraju segmenti izvoda čitavog regiona, neophodno je odrediti njegovo minimalno razapinjujuće stablo. Prvenstveno se kreiraju razapinjujuća stabla svakog izvoda posebno, a nakon toga se granični čvorovi koji pripadaju razapinjujućem stablu izvoda, preko grana minimalne težine povezuju sa njegovim susjedom.

2.3. Kreiranje segmenata izvoda

Generisano minimalno razapinjuće stablo regiona predstavlja polaznu tačku za formiranje segmenata izvoda.

Segmenti izvoda tipa *trunk* se kreiraju od opreme koja pripada minimalnom razapinjućem stablu. Pretraga stabla ide u dubinu i obilaskom grafa se kreira jedan po jedan segment. Ispunjenjem jednog od sljedećih uslova, trenutni segment fidera se "prekida" i kreiraju se novi segmenti:

- Algoritam naišao na važan čvor.
- Algoritam naišao na granu koja modeluje važnu opremu koja nije sekcija.
- Algoritam naišao na čvor sa više od dvije susjedne grane, koje pripadaju minimalnom razapinjućem stablu.

Skup opreme, koja pripada segmentima tipa *trunk* se može prikazati kao jedna grana. Od ostatka opreme se kreiraju segmenti izvoda tipa *lateral* i *loop*.

U nastavku će biti opisan algoritam za kreiranje segmenata izvoda koji sadrže isključivo nevažnu opremu. *Loop*-ovi se javljaju u paru i grade upetljanu strukturu - krećući se granama petlje, od čvora preko kog je petlja zakačena na minimalno stablo, moguće je doći do istog čvora ili nekog drugog koji, takođe, pripada minimalnom stablu, istog ili različitog izvoda. Petlja unutar jednog izvoda povezuje čvorove jednog minimalnog stabla. Ako ovakva petlja sadrži normalno otvoren prekidač, onda se kreiraju dva segmenta tipa *loop*, razdvojena tim prekidačem. Petlja između dva izvoda povezuje čvorove minimalnih stabala tih izvoda.

Svi ostali segmenti, koji su izgrađeni samo od nevažne opreme su tipa *lateral*. Za svaki čvor minimalnog stabla, algoritam traži susjedne grane, koje ne pripadaju minimalnom stablu. Za svaku takvu početnu granu kreira novi segment fidera, čiji tip zavisi od uslova opisanih u prethodnim pasusima. U polje *equipmentOnThePath* se dodaje oprema koju modeluje početna grana i grane do kojih se može stići iz početne grane, a koje ne pripadaju minimalnom stablu.

2.3. Analiza inicijalnog rješenja

Inicijalno rešenje se zasniva na akcijama kreiranja segmenta izvoda nad čitavim reginom, koji može imati veliki broj izvoda i izvršavanje algoritma može biti nezadovoljavajućeg trajanja. Stoga se nameće pitanje na koji način da se izbjegne kreiranje segmenata cijelog regiona, ukoliko dođe do promjene modela, naročito ako su promjene nerelevantne za segmente izvoda.

3. UNAPREĐENJE ALGORITMA

Analiza inicijalnog rješenja je uvela ideju detekcije situacija kada promjena modela distributivne mreže ne zahtijeva generisanje segmenata izvoda čitavog regiona. Potencijalno unapređenje kojim bi se izbjeglo nepotrebno kreiranje segmenata izvoda bi umanjilo trajanje izvršavanja algoritma, a time i čitavu aktivnost izmjene modela distributivne mreže. Da bi se ovakav napredak postigao neophodno je prepoznati promjene koje nisu od značaja za kreiranje segmenata, kao i promjene koje neće uticati na čitav region, već samo na jedan izvod ili jedan segment. Promjene su definisane kolekcijama operacija za dodavanje, ažuriranje i brisanje entiteta.

3.1. Uvođenje graničnih segmenata izvoda

U potrazi za boljim performansama DMS-a i realizaciji naprednog rješenja prvi korak je uvođenje novih tipova segmenata izvoda – graničnih segmenata (*boundary trunk, boundary lateral i boundary loop*). Segment je granični ako sadrži granični čvor, tj. čvor koji je povezan sa opremom nekog drugog izvoda.

Promjene negraničnih segmenata uglavnom ne utiču na susjedne izvode i ostatak regiona. S druge strane, promjene graničnih segmenata mogu izazvati ”lančane” promjene segmenata u ostatku regiona, pa je u tom slučaju jednostavnije i brže kreirati segmente izvoda u regionu ispočetka.

Stoga je pogodno prilikom kreiranja segmenata prepoznati one koji su granični, kako bi se izvršio odgovarajući postupak u slučaju promjene segmenata.

3.2. Filtriranje zahtjeva za promjenom modela

Dalje unapređenje rješenja podrazumijeva filtriranje zahtjeva za promjenom modela. Svi dodati i obrisani entiteti direktno utiču na konektivnost mreže, a samim tim i na segmente izvoda. Stoga se operacije koji opisuju ove akcije automatski proglase za relevatne. Međutim, samo podskup operacija koje definišu ažuriranje entiteta imaju uticaj na segmente izvoda:

- Promjena čvorova za koje je zakačena oprema.
- Promjena važnosti opreme.
- Promjena debljine električnog voda koji predstavlja glavnu trofaznu granu (*trunk*).
- Promjena debljine bilo kog električnog voda koja uzrokuje promjenu važnosti tog voda.

Nakon filtriranja relevantnih promjena slijedi definisanje segmenata na koje one utiču.

3.3. Obrada promjena relevantnih za segmente izvoda

U daljem procesu kreiranje segmenta izvoda (tokom inkrementalne promjene modela distributivne mreže) analiziraju se relevante promjene modela. Rezultat analize predstavljaju sljedeće kolekcije:

- izvodi nad čijim regionima je neophodno kreirati segmente,
- izvodi za koje je pojedinačno potrebno kreirati segmente,
- oprema koja se dodaje u postojeći segment, bez potrebe za izvršavanjem algoritma.

Za sve relevantne operacije dodavanja, brisanja ili ažuriranja entiteta se provjerava da li je segment izvoda, kome entitet pripada, granični ili ne. Ukoliko segment nije granični, njemu odgovarajući izvod se markira za obradu, u suprotnom čitav region je markiran.

Ako se dodaje nevažna oprema, a segment je tipa *lateral* ili *loop* (bili oni granični ili ne) oprema se markira za ubacivanje u postojeći segment.

3.4. Kreiranje segmenata izvoda

Nakon pripreme podataka izvršava se kreiranje segmenta izvoda nad željenim dijelom mreže. Prvenstveno se formiraju segmenti za izvode u regionima markiranim za obradu. Za izvode markirane za pojedinačnu obradu se segmenti kreiraju nezavisno, bez formiranja regiona. S obzirom na lančani uticaj promjene segmenata izvoda neophodno je tekući (novonastali) skup graničnih segmenta uporediti sa pređašnjim. Ukoliko je do promjene došlo (npr. pojavio se novi granični trunk) postoji mogućnost da je takva promjena izazvala promjene i u segmentima susjednih izvoda. U tom slučaju, da bi se očuvao korektan poredak segmenata, potrebno je pokrenuti kreiranje segmenata nad čitavim regionom.

S obzorom na karakteristike *lateral* i *loop* segmenata (graničnih ili ne), oprema, pod pretpostavkom da je nevažna, se može direktno uključiti. Na taj način je izbjegnuto pokretanje algoritma za kreiranje segmenata cijelog izvoda ili regiona izvoda. Kada se oprema direktno uključuje u postojeći segment neophodno je proširiti kolekciju koja definiše opremu datog segmenta identifikatorom opreme.

4. EKSPERIMENTALNA STUDIJA

Implementirana rješenja su primjenjena na regionu, koji sadrži 76 izvoda realne distributivne mreže. U Tabeli 4.1 prikazana su prosječna vremena izvršavanja algoritama kada se dodaje oprema u mrežu (scenario 1) i kada se briše oprema iz mreže (scenario 2).

Tabela 4.1. *Vremena izvršavanja algoritama prilikom dodavanja i brisanja opreme u negraničnim segmentima*

	Osnovni algoritam [ms]	Unaprijeđeni algoritam [ms]
Scenario 1	12246	33
Scenario 2	12569	205

U Tabeli 4.2 su prikazana prosječna vremena potrebna za dodavanje opreme koja povezuje 2 izvoda. Ovaj scenario uzrokuje kreiranje segmenata nad cijelim regionom izvoda kada je u pitanju osnovni i unaprijeđeni algoritam.

Tabela 4.2. *Vremena izvršavanja algoritama prilikom dodavanja opreme u granične segmente*

	Osnovni algoritam	Unaprijeđeni algoritam
Vrijeme [ms]	11896	11964

Na osnovu eksperimentalne studije zaključeno je sljedeće:

- Unaprijeđeno rješenje donosi značajna poboljšanja po pitanju performansi u odnosu na inicijalno rešenje u pojedinim slučajevima. Brzina izvršavanja je znatno veća kada se ažurira postojeći segment ili kreiraju segmenti jednog izvoda. Unapređenje naročito dolazi do izražaja kada region obuhvata veliki broj izvoda.
- U preostalim slučajevima, unaprijeđeno rješenje takođe podrazumijeva kreiranje segmenata nad svim izvodima u regionu, odnosno svodi se na inicijalno rješenje. Pri tome, dodatna obrada informacija u okviru unapređenog algoritma ne donosi primjetno usporenje.

5. ZAKLJUČAK

Eksperimentalna studija je potvrdila efikasnost i korisnost implementiranih algoritama. Oba pristupa, inicijalni i unaprijeđeni, obezbjeđuju smanjenje nivoa detaljnosti prikaza distributivne mreže, tako da prikaz postaje jasniji i omogućuje fokus na najvažnije entitete mreže. Drugim riječima, rješenje zasnovano na segmentima izvoda je omogućilo prikaz samo onih entiteta koji su od značaja korisniku.

Dodatno analizom performansi, unaprijeđeni pristup je postigao više nego zadovoljavajuće rezultate. U pojedinim situacijama izmjene modela distributivne mreže, a samim tim i segmenata izvoda, čija frekvencija nije zanemarljiva, vrijeme izvršavanja unaprijeđenog algoritma je znatno kraće u odnosu na vrijeme izvršavanja inicijalnog rješenja. Međutim, očigledno je da postoje scenariji koji imaju uticaj na segmente čitavog regiona kada su performanse oba pristupa identične.

6. LITERATURA

[1] R.DeBlasio, T.Cherry, "*Standards for the Smart Grid*", In Proceedings of International Conference IEEE Energy 2030, Atlanta, USA, pp. 1-7, 2008

[2] D.Popović, "*Power applications: A cherry on the top of the DMS cake*", DA/DSM DistribuTECH Europe, Vienna, Austria, 2000.

[3] V.Strezoski, "Osnovi elektroenergetike", Fakultet tehničkih nauka, Novi Sad, 2014.

[4] *"IEC 61970–301 (2009-04) Ed. 2.0 Energy management system application program interface (EMS-API)"*, 2009, ISBN 978-2-88910-593-9

[5] R. Diestel, "*Graph Theory*", 4th Edition, Springer, ISBN 978-3-642-14278-9, 2012.

[6] B. Stojanovic, *"Algoritamske strategije - Materijali za predavanja"*, Prirodno matematički fakultet, Kragujevac

Kratka biografija:

Sara Pavlović rođena je u Vogošći 1995. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva, smjer Primjenjeno softversko inženjerstvo odbranila je 2018.god.
Kontakt: srpvlvc@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4:621.395
DOI: https://doi.org/10.24867/01BE04Stevanovic

XAMARIN.FORMS ОКВИР ЗА РАЗВОЈ МОБИЛНИХ АПЛИКАЦИЈА

XAMARIN.FORMS CROSS-PLATFORM FRAMEWORK

Милош Стевановић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Тема мастер рада спада у дисциплину програмирања мобилних апликација. У мастер раду је приказан увод у развој вишеплатформских апликација и употребу Xamarin.Forms алата за њихов развој. Приказан је развој мобилне апликације коришћењем Xamarin.Forms алата као и опис развијене апликације.*

Abstract – *The subject of the master thesis belongs to the discipline of mobile application development. The thesis contains introduction to cross-platform development of mobile applications and usage of Xamarin.Forms tool. It is shown process of mobile application development using Xamarin.Forms tool.*

1. УВОД

Да би мобилна апликација била доступна што већем броју корисника потребно је да буде развијена за више платформи. То углавном није једноставан задатак зато што свака од популарних платформи (*Android*, *iOS* и *Windows*) подржава само апликације које су развијене за њу. *Cross platform* оквири за израду мобилних апликација омогућавају да једном развијена мобилна апликација може да се покрене на свим популарним платформама. Постоји велики број *cross platform* оквира, а један од најпопуларнијих је *Xamarin*.

2. *XAMARIN.FORMS*

Xamarin.Forms је вишеплатформско окружење за развој мобилних апликација за *Android* [1], *iOS* [2] и *Windows* [3] платформе користећи програмски језик *C# [4]*. Представља сет алата који омогућавају развој заједничког корисничког интерфејса који се дефинише на једном месту али функционише на све три платформе. Свака платформа дефинише своје матичне компоненте корисничког интерфејса због чега оне изгледају и понашају се исто као и матичне компоненте на конкретним платформама. Апликације развијене коришћењем *Xamarin.Forms* алата могу да имају све матичне карактеристике и да истовремено деле заједнички програмски код, логику апликације.

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је био др Стеван Гостојић, ванр. проф.

2.1 Како Xamarin.Forms функционише

Све *Xamarin* библиотеке, *Xamarin.Android*, *Xamarin.iOS* и *Xamarin.Forms*, развијене су помоћу *Mono* платформе. *Mono* платформа представља отворену (енг. *open-source*) верзију *.NET [5]* окружења које може да се покреће на различитим оперативним системима (*Linux*, *Unix* и *macOS*).

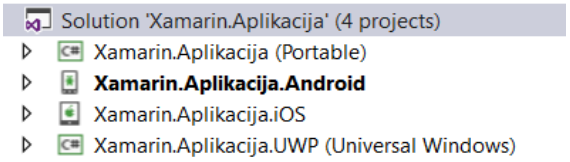
Xamarin.Forms је развијен на основу *Xamarin.Android* и *Xamarin.iOS* библиотека. Састоји се од *Xamarin.Forms.Core* библиотеке у којој се налазе класе у којима је дефинисан јединствени *API* за рад са више платформи. Садржи велики број функционалности које су мапиране на одговарајуће матичне функционалности појединачних платформи.

2.2 Развој *Xamarin.Forms* апликација

За развој *Xamarin* апликација могу да се користе *Visual Studio [6]* и *Xamarin Studio [7]* развојна окружења доступна за *Windows* оперативни систем, док се за *OS X* [8] оперативни систем користе *Xamarin Studio* и *XCode [9]*. *Xamarin* сет алата може да се преузме помоћу *Visual Studio* развојног окружења.

2.2.1 Структура Xamarin.Forms пројекта

Могућност развоја матичних апликација за више платформи захтева специфичну структуру пројеката која се састоји од пројеката везаних за сваку платформу и пројекта у коме се налази заједнички код за све платформе (енг. *shared code*). На слици 2.1 је приказан *Solution* са пројектима потребним да се апликација развија за све три платформе.



Слика 2.1. *Xamarin.Forms solution*

Пројекат означен на слици 2.1 као Portable је пројекат у коме се налази заједнички код за све три платформе, док су остали пројекти везани за специфичну платформу.

2.3 Елементи *Xamarin.Forms* апликације

2.3.1 XAML

Кориснички интерфејс (*UI*) у *Xamarin.Forms* апликацији може да се дефинише процедурално у класама и декларативно користећи *XAML* (*eXtensible Application Markup Language*). *XAML* није неопходан за дефинисање корисничког интерфејса, али је у већини случајева мање обиман и прегледнији од

еквивалентног C# кода. Користи хијерархију родитељ - дете што доприноси бољем представљању структуре корисничког интерфејса. *XAML* је посебно погодан уколико се користи *MVVM* (*Mode-View-ViewModel*) архитектура јер у *XAML* језику могу да се дефинишу погледи (*View*) који се повезују са *ViewModel*-ом.

2.3.2 *Стране*

Стране у *Xamarin.Forms* апликацији представљају визуелне елементе који се приказују на екрану мобилног уређаја. Стране су дефинисане *Page* класом и садрже погледе (тј. компоненте корисничког интерфејса помоћу којих корисници врше интеракцију са мобилном апликацијом). Постоји више унапред дефинисаних типова страница: *ContentPage*, *NavigationPage*, *MasterDetailPage* и *TabbedPage*.

2.3.3 *Погледи (views)*

Погледи представљају визуелне компоненте које се приказују на странама. *Xamarin.Forms* садржи више унапред дефинисаних погледа који се на мобилном уређају приказују као матичне компоненте.

2.3.4 *Организација садржаја на екрану*

За организацију садржаја који се приказује на екрану користе се распореди (енг. *layout*). Распореди су визуелни елементи који могу да садрже погледе или друге распореде.

2.3.5 Application класа

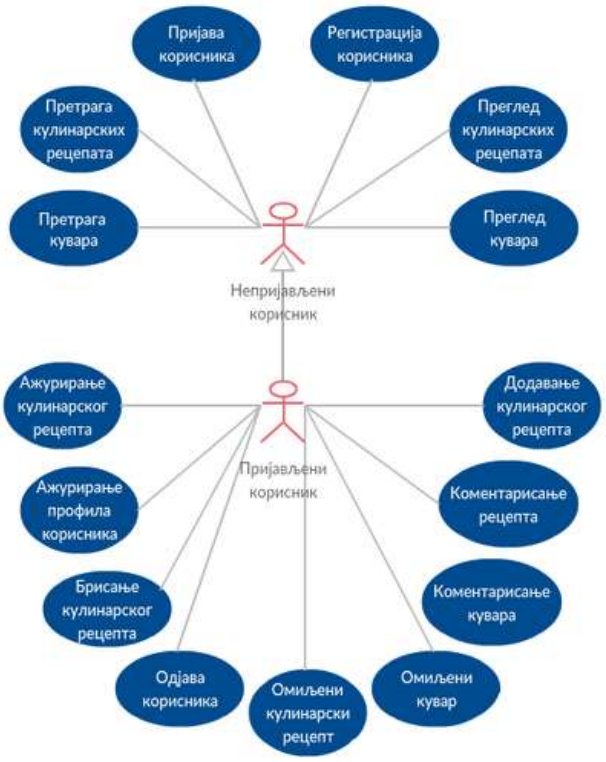
Application класа је основна класа *Xamarin.Forms* апликације. Представља страну која се прва отвара по покретању апликације, може да чува променљиве примитивног типа у пољу *Properties* које представља кључ-вредност структуру и садржи догађаје одговорне за приказивање и уклањање модалних погледа. У оквиру апликације је изложена *App* поткласом чији атрибут *Current* садржи референцу на тренутни објекат класе *Application*. Такође, садржи референцу на страну која се тренутно приказује (*MainPage*).

У *Application* класи су изложене методе животног циклуса апликације: *OnStart* (позива се током покретања апликације), *OnSleep* (позива се када апликација није више у првом плану) и *OnResume* (позива се када апликација поново дође у први план).

3. СПЕЦИФИКАЦИЈА ЗАХТЕВА

FoodBook мобилна апликација је имплементирана као пример употребе *Xamarin.Forms* алата. Апликација кориснику омогућава преглед, оцењивање и коментарисање кулинарских рецепата, преглед, оцењивање и коментарисање кувара, као и додавање сопствених кулинарских рецепата.

Све функционалности које корисник може да изврши на апликацији специфициране су коришћењем дијаграма случајева коришћења (енг. *use case*). На слици 3.1 приказан је дијаргам случајева коршћења *FoodBook* мобилне апликације.



Слика 3.1. *Дијаграм случајева коришћења*

4. СПЕЦИФИКАЦИЈА ДИЗАЈНА

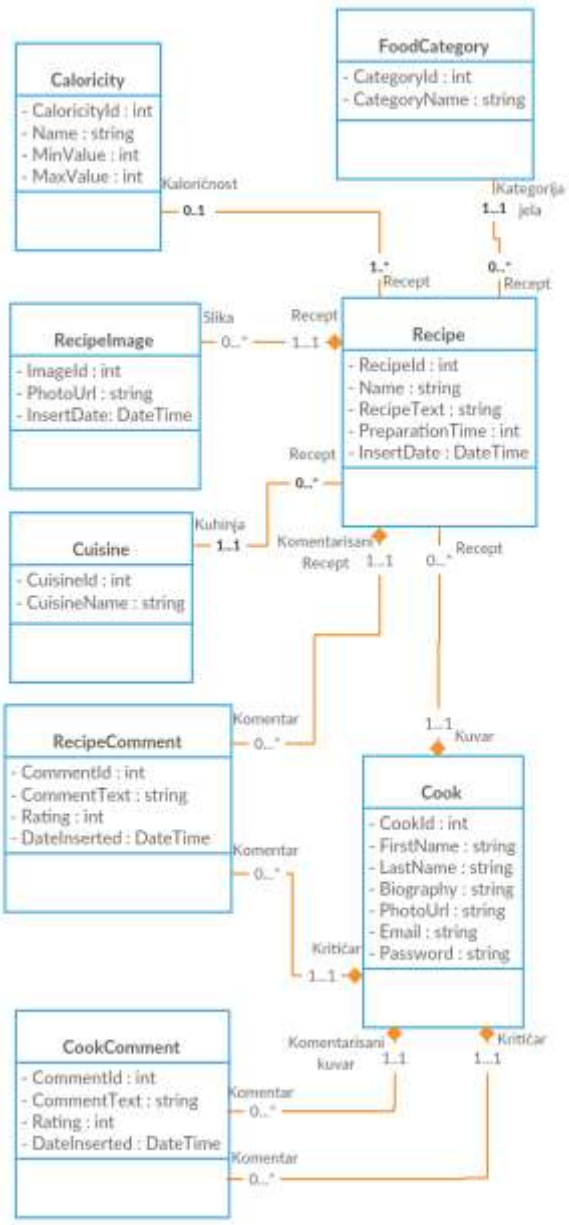
За опис модела података је употребљен дијаграм класа на ком су приказане класе и њихов међусобни однос.

На слици 4.1 приказане су класе и њихове међусобне везе. Класа *Recipe* представља кулинарски рецепт који може бити креиран од стране кувара који је представљен класом *Cook*.

Кулинарски рецепт припада одређеној категорији јела (класа *FoodCategory*), одређеној кухињи (класа *Cuisine*) и може садржати више слика (класа *RecipeImage*). Кулинарски рецепт може бити коментарисан од стране корисника мобилне апликације, коментар рецепта је представљен класом *RecipeComment*. Као и кулинарски рецепти, кувари такође могу бити коментарисани. Коментари кувара представљени су класом *CookComment*.

За приказ распореда компоненти апликације употребљен је дијаграм распореда. Дијаграм распореда *FoodBook* апликације приказан је на слици 4.2.

На дијаграму је приказано више компоненти од којих је главна компонента мобилна апликација имплементирана у *Xamarin.Forms* платформи. Мобилна апликација је састављена од компоненти у којима се налази заједничка пословна логика и компоненти везаних за конкретну платформу. Мобилна апликација комуницира са веб сервисом и на тај начин добавља и шаље податке.



Слика 4.1 - Дијаграм класа *FoodBook* апликације.

База података је компонента која складишти податке потребне мобилној апликацији. Поред базе података, веб сервис компонента комуницира са *Azure Cloud [10]* компонентом на којој се налазе компонента за складиштење слика и компонента за слање нотификација мобилној апликацији.

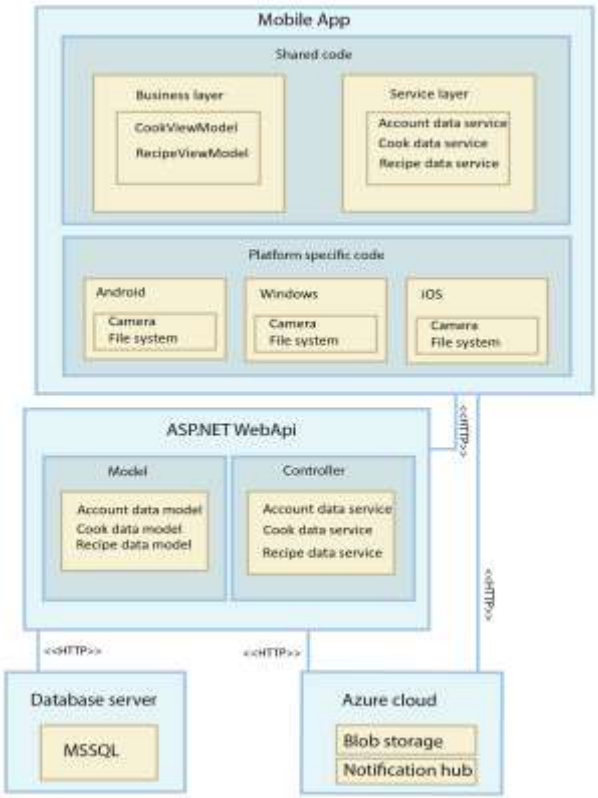
5. ИМПЛЕМЕНТАЦИЈА

Као пример употребе *Xamarin.Forms* платформе имплементирана је *FoodBook* мобилна апликација.

5.1 Почетак рада са платформом

За развој *FoodBook* мобилне апликације коришћено је *Visual Studio* развојно окружење на *Windows* рачунару што је довољна конфигурација за развој апликације за *Android* и *Windows* платформу.

Развој апликације за *iOS* платформу додатно захтева рачунар са *OS X* оперативним системом који се користи за извршавање кода *iOS* апликације.



Слика 4.2 - Дијаграм распореда *FoodBook* мобилне апликације

5.2 Имплементација екрана

За имплементацију страница апликације коришћен је *MVVM* архитектонски шаблон. Свака имплементирана страница садржи *Model*, *View* и *ViewModel* компоненте. Екрани *FoodBook* мобилне апликације приказују податке сачуване у бази података који се добављају позивајући веб сервисе. Након добављања података потребно је да се они припреме и проследе *View* компоненти која податке приказује на екран. Манипулација са подацима се обавља у *ViewModel* компоненти која садржи поља и команде које се везују за елементе корисничког интерфејса. Кориснички интерфејс, односно *View* компонента, садржи погледе помоћу којих се кориснику приказују одређени подаци.

Погледи помоћу којих корисник може да интереагује са апликацијом повазани су са пољима *ViewModel* компоненте техником везивања података (*data binding*).

5.3 *Комуникација између компоненти*

За комуникацију између независних компоненти апликације *Xamarin.Forms* нуди класу *MessagingCenter* која омогућава компонентама да се претплате на добијање информација које шаље нека друга компонента.

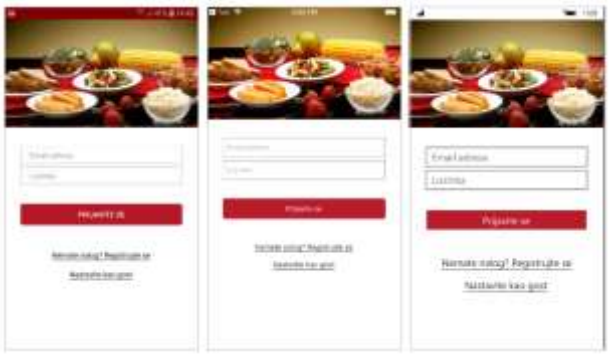
6. ДЕМОНСТРАЦИЈА

Након покретања апликације отвара се почетни екран за пријаву корисника (ако корисник није пријављен). Екран за пријаву корисника омогућава кориснику да се пријави на систем, а уколико није регистрован постоји опција за регистрацију корисничког налога.

Избор опције за регистрацију налога отвара нови екран на коме корисник уноси потребне податке за креирање новог налога. Након успешне регистрације корисник може да се пријави на апликацију уз помоћ креираног налога.

Ако корисник не жели да креира налог може приступити апликацији као непријављени корисник избором опције "Наставите као гост". Непријављеном кориснику су доступни екрани за претраживање и преглед информација о постојећим кулинарским рецептима и куварима.

Пријављеном кориснику су такође доступни исти екрани али он има могућност креирања нових кулинарских рецепата, преглед детаља постојећих рецепата уз могућност избора кулинарских рецепата које су креирали остали корисници апликације у листу омиљених кулинарских рецепата. На екрану за преглед информација о рецепту пријављеном кориснику је доступна опција коментарисања тог рецепта, такође на екрану за преглед информација о одређеном кувару постоји могућност остављања коментара за тог кувара.



Слика 6.1. *Екран за пријаву корисника приказан на Android, iOS и Windows платформама тим редоследом*

7. ЗАКЉУЧАК

Тема овог рада је Xamarin.Forms оквир за имплементацију вишеплатформских мобилних апликација.

Xamarin.Forms је оквир вредан пажње зато што омогућава развој апликација за више платформи уз велики удео заједничког кода. Апликације развијене употребом ове платформе не заостају у великој мери у односу на апликације развијене коришћењем матичних технологија, а платформа омогућава коришћење једног програмског језика за имплементацију апликација на више платформи.

Поред свих добрих страна *Xamarin.Forms* платформе, током развоја и коришћења апликације приметне су и одређене мане. Не постоји стабилан прегледач корисничког интерфејса креираног у *XAML* документу, па креирани кориснички интерфејс може да се види тек након покретања апликације. Апликације креиране помоћу ове платформе заузимају доста више простора на мобилном уређају у односу на матичне апликације.

Иако је највећи део планираних функционалности имплментиран, FoodBook мобилна апликација може да се развија даље. Планиран је развој вишејезичности, као и додавање видео записа за кулинарски рецепт. Такође је планирана имплементација приватних профила корисника и приватних рецепата, којима могу да приступе само регистровани корисници. Рад апликације у режиму без интернет конекције би свакако био пожељан што би допринело да апликација буде доступна у сваком тренутку.

8. ЛИТЕРАТУРА

[1] Android, https://www.android.com/

[2] iOS, http://www.apple.com/ios

[3] Windows, https://www.microsoft.com/en-us/windows

[4] C#, https://docs.microsoft.com/en-us/dotnet/csharp/

[5] .NET, https://www.microsoft.com/net/

[6] Visual Studio, https://visualstudio.microsoft.com/

[7] Xamarin Studio, https://developer.xamarin.com/releases

[8] OS X, https://support.apple.com/macos/

[9] XCode, https://developer.apple.com/xcode/

[10] Azure Cloud, https://azure.microsoft.com/

Кратка биографија:

Милош Стевановић рођен је 26.05.1991. године у Приштини, Република Србија. У Смедереву је 2010. године завршио Средњу техничку школу, основне академске студије завршио је на Факултету техничких наука у Новом Саду 2014. године, а мастер академске студије на истом факултету уписао је 2014. године. Положио је све испите предвиђене студијским програмом.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE05Rasajski

PRIMENA METODA MAŠINSKOG UČENJA ZA AUTOMATSKU KLASIFIKACIJU MUZIKE PO ŽANRU

AUTOMATIC MUSIC GENRE CLASSIFICATION USING MACHINE LEARNING

Nemanja Rašajski, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Muzički žanrovi su konvencionalne kategorije koje se koriste za opisivanje muzike. Danas se najčešće koriste za klasifikaciju rastućeg broja muzičkih numera, koja bi dalje trebalo da omogući precizniju preporuku i jednostavniju pretragu muzike. U radu je analizirano nekoliko metoda i strategija za automatsku klasifikaciju muzike uključujući konvolucione neuronske mreže (Convolutional neural network – CNN), rekurente neuronske mreže (Reccurent neural network – RNN), mašine potpornih vektora (Support vecotor machines – SVM), random forrest (RF), AdaBoost kao i One vs. Rest (OVR) i klasifikaciju glasanjem. Muzičke numere klasifikovane su na osnovu mel-frequency cepstrum coefficients (MFCC) predstave audio zapisa, a za potrebe CNN-a korišćen je spektrogram. Ostvareni rezultati (~60%) se mogu porediti sa tačnošću (~70%) sa kojom su ljudi u stanju da ispravno procene muzički žanr kao i sa rezultatima ostvarenim u radovima koji su se bavili sličnom temom na istom skupu podataka. Obzirom da preciznost ostvarena u radu nije daleko od procene ljudi, metode bi mogle naći primenu u automatskoj klasifikaciji muzike za potrebe radio stanica ili web sajtova koji se bave distribuiranjem i preporukom muzičkih numera.*

Ključne reči: *klasifikacija muzike po žanrovima, mašinsko učenje, GTZAN skup podataka.*

Abstract – *Music genres are conventional categories that are used for describing music. Today they are most often used for classifying growing music collections, for easier access and recommendation. This paper has analyzed a number of methods for automatic music classification, which include convolutional neural networks (CNN), recurrent neural networks (RNN), support vector machines (SVM), random forests (RF), AdaBoost, voting classifier and one versus rest (OVR). Features used for audio classification were created using mel-frequency cepstrum coefficients (MFCC), and spectrograms which were used in combination with CNNs. The accuracy achieved (~60%) was not at the human level (~70%), but it was not too far off, and it is in line with other similar approaches. Thus, methods presented in this paper can be used for automatic classification of music, by radio stations or web portals that distribute and recommend music.*

Keywords: *Automatic music genre classification, machine learning, GTZAN dataset*

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kovačević, vanr. prof.

1. UVOD

Razvoj interneta i prenosnih medija omogućio je korisnicima pristup velikim količinama multimedijalnog sadržaja. Da bi se organizovale i pretražile rastuće muzičke kolekcije, potrebni su automatizovani alati za filtriranje i procesiranje audio fajlova. Neke od ovih funkcionalnosti mogu biti olakšane pomoću metapodataka, koji pružaju dodatne informacije o sadržaju. Često ti metapodaci nisu dostupni, pa je potrebno analizirati podatke bez njih. Tada se neophodne informacije o pesmama uzimaju direktno iz audio fajlova. Takve informacije mogle bi obuhvatati žanr, raspoloženje, stil, izvođača. U ovom radu akcenat je na indentifikovanje žanra pesme iz grupe od 10. Većina sistema za klasifikaciju muzike imaju dva koraka, izvlačenje osobina na osnovu kojih se vrši klasifikacija i sama klasifikacija. Za svrhe klasifikacije koriste su razne osobine signala, poput zero-crossing, bandwidth, spetral centroid. Skup osobina korišćen za klasifikaciju u ovom radu je MFCC [2], a korišćeni su i spektrogrami (spectrogram) [4] koji su grafička predstava spektruma kao i hromagram [6] (chromagram).

Isprobane su metode predložene u prethodnim radovima koji su se bavili ovom temom, poput SVM (Support vector machines) [8]. Takođe su istražene varijacije gore napomenutih metoda, poput SVM ansambla, kao i u ovoj oblasti slabije istraženi algoritmi poput RNN, CNN.

Skup podataka korišćen za izradu rada je GTZAN Dataset – kolekcija 1000 audio fajlova.

U poglavlju II, biće dat pregled postojećih rešenja. Poglavlje III detaljnije objašnjava skup podataka, kao i sve transformacije nad njim vršene za potrebe ovog rada. U poglavlju IV će biti reči o metodama korišćenim u procesu klasifikacije. Poglavlje V daje pregled i diskusiju rezultata. Poglavlje VI nudi zaključke koje je autor stekao.

2. PREGLED POSTOJEĆE RELEVANTNE LITERATURE

Algoritmi za automatsko prepoznavanje muzičkog žanra opisani su u [3]. U radu je takođe predložen skup svojstava za predstavu muzike, a performanse skupa svojstava evaluirane su treniranjem klasifikatora baziranih na statističkom prepoznavanju šablona. Za klasifikaciju je kreiran GTZAN skup podataka koji se koristi i u ovom radu i dobijena je tačnost od 61% korišćenjem Expectation-maximization (EM) i k-nearest neighbours (KNN) algoritma.

Analiza GTZAN skupa podataka izložena je u [5]. U radu je navedeno da 7.2% audio fajlova potiču iz istog

muzičkog zapisa, uključujući 5% audio fajlova koji su potpuno identični, te mogu dovesti do pogrešne predstave rezultata (npr. ukoliko se isti zapis koristi i za treniranje i za testiranje). Dalje, autori tvrde da je 10.6% audio fajlova pogrešno označeno vodeći se načinom na koji su pesme i autori označeni na sajtu last.fm, kao i da pojedini izvođači dominiraju određenim žanrom (npr. 34% audio fajlova iz rege žanra izvodi Bob Marley) pa se postavlja pitanje iskoristivosti treniranih klasifikatora za klasifikaciju muzike ostalih izvođača.

Radovi predstavljaju dobru osnovu za dalji rad i daju uvid u to kakve rezultate treba očekivati, kao i koja su ograničenja GTZAN skupa podataka.

3. SKUP PODATAKA

Za izradu rada korišćen je GTZAN skup podataka [1] koji sadrži 1000 audio fajlova gde je trajanje svakog fajla 30 sekundi. Skup podataka je podeljen na 10 kategorija (bluz, klasična, metal, kantri, džez, pop, rok, rege, hiphop i dens muzika), a za svaku kategoriju postoji 100 muzičkih numera. Sve numere su 22050 Hz monofoni 16-bit audio fajlovi u AU formatu. Podaci su konvertovani u WAV format. Na osnovu liste date u [5] uklonjeni su svi istovetni audio fajlovi iz skupa podataka, a klase izbalansirane nasumičnim izostavljanjem primera. Tako modifikovani skup podataka sadrži 850 audio fajlova - po 85 audio fajlova za svaki od žanrova, odnosno nešto više od 7 sati ukupnog materijala.

Za treniranje klasifikatora korišćene su tri reprezentacije audio signala - MFCC, hromagram (chromagram) i spektrogram (spectrogram).

3.1 MFCC

Mel-frequency cepstrum (MFCC) reprezentacija je bazirana na načinu na koji ljudi doživljavaju muziku. MFCC sadrži dva tipa filtera koji su raspoređeni linearno na niskim frekvencijama ispod 1000 Hz i logaritamski na frekvencijama iznad 1000 Hz.

Proces pretvaranja audio signala u MFCC reprezentaciju obuhvata šest koraka [2]:

1. *Isticanje (Pre-emphasis)*

Obuhvata proces prosleđivanja signala kroz filter koji naglašava visoke frekvencije

2. *Kadriranje (Framing)*

Signal se deli na frejmove od N uzoraka. Susedni frejmovi se dele na M podfrejmova gde je M < N. Tipične vrednosti koje se koriste su M=100 i N=256.

3. *Okno Heminga (Hamming window)*

Jednačina glasi:

$$Y(n) = X(n) \times W(n) \quad (1)$$

gde je Y(n) izlazni signal, X(n) ulazni signal, a W(n) se definiše kao:

$$W(n) = 0.54 - 0.46cos\left(\frac{2\pi n}{N} - 1\right)\ 0 \le n \le N - 1 \quad (2)$$

gde je N širina izražena u semplovima.

4. *Brza Furijeova transformacija (Fast Fourier Transformation)*

Svaki frejm od N semplova se konvertuje iz vremenskog domena u frekventni domen.

5. *Procesiranje skupa mel filtera (Mel Filter Bank processing)*

Faza obuhvata izračunavanje težinske sume spektralnih komponenti filtera tako da se izlaz može aproksimirati Mel skalom. Sledeća jednačina se koristi za izračunavanje opsega (mel) za svaku frekvenciju f u Hz:

$$F(Mel) = [2595 * log10[1 + f]700] \quad (3)$$

6. *Diskretna kosinusna transformacija*

Proces podrazumeva konvertovanje logaritamskog Mel spektra u vremenski domen korišćenjem diskretne kosinusne transformacije.Rezultat konverovatnja je MFCC. Skup koeficijenata predstavlja akustični vektor.

3.2 HROMAGRAM

Hroma (chroma) je jedna od dve komponente koje se mogu dobiti iz visine zvuka. U poređenju sa tonovima u muzici, hroma predstavlja elemente u skupu {C, $C^{\#}$, D,...,B}, gde oznake $C^{\#}$ i D^{b} odgovaraju istoj hromi. Klasa visine zvuka je definisana kao skup svih tonova koji dele istu hromu.

Reprezentacija hrome se dobija tako što se svakoj klasi dodeljuju tonovi kojima oni pripadaju. Tačnije, audio signal se prvo rastavlja na 88 podgrupa koristeci filtere, izračunava se STMSP (Short-time mean-square power) za svaku podgrupu, i potom se sabiraju svi STMPS koji pripadaju istoj klasi. Ovo rezultuje u realnom 12-dimenzionalnom vektoru, gde v(1) odgovara hromi C, v(2) hromi $C^{\#}$ i tako dalje. Primena ove metode na različitim vremenskim intervalima se naziva hromagram [6].

3.3 SPEKTROGRAM

Spektorgram je grafička reprezentacija spektra frekvencija u zvučnom ili drugom signalu, po vremenu ili nekoj drugoj promenljivoj. Često se koriste u oblasti analiziranja muzike.

Spektrogrami se predstavljaju u formi grafika, gde horizontalna osa predstavlja vreme, a vertikalna osa frekvenciju, treća dimenzija koja predstavlja amplitudu određene frekvencije u određeno vreme, predstavljena je intenzitetom boje na svakoj tački slike.

Spektrogrami se obično kreiraju na dva načina. Aproksimacija pomoću filterbanka (niz band-pass filtera), ili pomoću Fourierove transformacije [7]. U ovom radu spektrogrami su generisani pomoću Fourierove transoformacije. Proces kreiranja spektrograma odgovara računanju, kvadrat apsolutne vrednosti STFT (Short-time Fourier transform) signala *s(t)* za interval *ω* [4]*:*

$$\text{spectrogram}(t,\omega) = |\text{STFT}(t,\omega)|^2 \quad (4)$$

4. METOD

S ciljem klasifikacije muzike korišćeno je nekoliko algoritama i strategija. Evaluacija algoritama nad istim skupom podataka pružila je uvid u iskoristivost datih algoritama za rešavanje problema klasifikacije muzike. Razlozi za njihovo dati su u nastavku.

4.1 CNN

Konvolutivne neuronske mreže su našle veliku primenu u oblasti klasifikacije slika, gde daju izuzetne rezultate (iznad 95% na ImageNet takmičenju, koje zahteva

klasifikaciju na 200 klasa). Tema ovog rada je klasifikacija muzike na osnovu audio signala, gledajući da se i on može grafički predstaviti pomoću spektrograma. CNN je izabrana u pokušaju da se problem klasifikacije pesama, pretvori u problem klasifikacije slika i primeni algoritam koji se pokazao kao veoma uspešan u tom polju.

4.2 RNN

Pretpostavka je da bi rekurentna neuronska mreža bila u stanju da izmodeluje temporalne zavisnosti među semplovima.

Za treniranje rekurentne neuronske mreže korišćena je NEAT metoda. Metod se zasniva na evoluciji neuronske mreže kroz promenu kako topologije mreže tako i težina. [9]. Kroz proces koji obuhvata enkodiranje, mutaciju, ukrštanje, selekciju, praćenje najboljih gena i očuvanje inovativnih rešenja, algoritam je u stanju da od inicijalne neuronske mreže kreira mrežu koja je u stanju da reši problem za koji je evoluirana.

4.3 AdaBoost

Dve karakteristike AdaBoost algoritma su otpornost na overfitovanje i osetljivost na šumove.. Sa jedne strane, otpornost na overfitovanje će omogućiti bolju generalizaciju što je esencijalno za uspšeno klasifikovanje muzike, dok sa druge strane, osetljivost na šumove može predstavljati prepreku (što će se kasnije i primetiti u rezultatima) sa obzirom na raznolikost u numerama. Za slabe klasifikatore odabrani su stabla odlučivanja zato što se smatra da oni daju najbolje rezultate.

4.4 Random Forest

Razlog za korišćenje Random forest algoritma leži u njegovoj robustnosti na autlajere i šum, brzini izvršavanja obzirom da ga je jednostavno paralelizovati i tačnosti koja je jednako dobra ili bolja od Adaboost algoritma [10].

4.5 SVM

SVM je algoritam koji se najčešće spominje u literaturi kada se rešava problem klasifikovanja muzike, rezultati dobijeni njegovim korišćenjem su među najboljim i zbog toga je procenjeno da je potrebno i njega primeniti.

4.6 One vs. Rest

One vs. Rest strategija korišćena je s ciljem postizanja boljih rezultata kod Random Forest, SVM i AdaBoost klasifikatora.

Metoda obuhvata formiranje deset birnarnih klasifikatora, balansiranje klasa, treniranje svih klasifikatora, njihovo evaluiranje i preuzimanje rešenja onog klasifikatora sa najvećom tačnošću.

Formianje klasifikatora podrazumeva kreiranje klasifikatora za svaki od žanrova (npr. klasifikator koji klasifikuje podatke na džez i nije džez ili metal i nije metal). Za treniranje je korišćeno 136 fajlova, a za testiranje 34 fajlova gde je odnos broja pozitivnih i negativnih primera 1:1 kako u skupu podataka za treniranje tako i u skupu podataka za testiranje.

Sva tri klasifikatora (RF, AdaBoost, SVM) su trenirana i testirana nad istim podacima.

4.7 Većinsko glasanje (Voting classifier)

Razlog za korišćenje većinskog glasanja leži u kombinovanju konceptualno različih klasifikatora za predviđanje klasa glasanjem. Kada se radi o klasifikatorima koji rezultuju sličnim performansama, većinsko glasanje može biti dobar izbor jer omogućava balansiranje nedostataka ovih klasifikatora. U ovom slučaju, za većinsko glasanje korišćeni su AdaBoost, SVM i Random Forest klasifikatori jer su pojedinačno dali slične rezultate.

5. REZULTATI I DISKUSIJA

Kako je istraživanje [11] pokazalo da su ljudi u stanju da sa 70% tačnosti procene žanr muzike, težilo se pronalaženju metode koja bi dostigla približnu tačnost ne bi li se takva metoda mogla koristiti za automatsku klasifikaciju muzike.

Sve metode testirane su na modifikovanom GTZAN skupu podataka (duplikati su uklonjeni, a klase izbalansirane nakon uklanjanja) korišćenjem 5-struke unakrsne validacije (5-fold cross validation). Svi audio fajlovi su iz AU formata konvertovani u WAV format. Nakon toga, izvršeno je konvertovanje WAV fajlova u CSV fajlove koji sadrže MFCC predstavu audio signala odnosno hromagram. Eksperimenti su pokazali da je MFCC dao dvostruko bolje rezultate od hromagrama pa su rezultati isloženi u nastavku nastali klasifikacijom audio fajlova na osnovu MFCC (s izuzetkom CNN-a gde je korišćen spektrogram). Za treniranje i testiranje korišćeno je prvih 12900 semplova što odgovara maksimalnom broju semplova pojedinih fajlova. Izuzetak je rekurentna neuronska mreža koja je zbog hardverskih ograničenja morala biti trenirana korišćenjem 2500 semplova obzirom da je trebalo voditi računa i o sekvencama.

Kao mera evaluacije korišćena je F mera, konkretno implementacija F mere iz biblioteke sklearn [12], micro i macro varijacije. U Tabeli 1 prikazani su rezultati svih testiranih metoda

Kao što se može videti, svi klasifikatori su dali približno iste rezultate, s izuzetkom RNN-a gde je skup podataka morao biti pet puta smanjen zbog hardverskih ograničenja. One vs. Rest strategija doprinela je povećanju tačnosti SVM metode, ali je negativno uticala na AdaBoost metodu. Razlog za to mogu je lošija tačnost binarnih AdaBoost klasifikatora u odnosu na Random Forest i SVM. Na samom kraju, glasanje klasifikatora dalo je najveću tačnost od 62% što odgovara rezultatu ostvarenom u [3] (61%) korišćenjem EM i KNN metode.
Ono što treba imati u vidu je da za potrebe treniranja i testiranja klasifikatora nije korišćen ceo skup podataka već 85% njega. Razlog za to su ponovljeni audio fajlovi - problem GTZAN skupa podataka [5]. U istom radu navedena su i neslaganja sa načinom na koji su audio fajlovi klasifikovani, ali je taj iskaz zanemaren vodeći se mišljenjem da razlog za pripisivanje muzičnje numere žanru može biti subjektivan.

Radi ispitivanja uticaj trajanja trening skupa na rezultate, klasifikatori su trenirani koristeći audio fajlove trajanja od 15s. Sa dvostruko manje semplova, rezultati su bili 47%

lošiji. Takođe, korišćenje hromagrama umesto MFCC-a rezultovalo je tačnošću klasifikatora koja se kretala oko 30%. Iz gore predstavljenog može se zaključiti da bi drugačija predstava audio signala kao i veći broj numera ili duže trajanje numere pozitivno uticalo na tačnost klasifikatora.

Tabela 1 – *Rezultati klasifikacije*

	macro F-score	micro F-score
CNN	0.58	0.59
RNN	0.21	0.23
Random Forest	0.57	0.59
AdaBoost	0.54	0.56
SVM	0.50	0.53
One vs. Rest		
SVM	0.55	0.59
Random Forest	0.51	0.60
AdaBoost	0.32	0.35
Voting Classifier		
SVM+RF+AB	0.57	0.62

6. ZAKLJUČAK

Tema ovog rada je bio problem automatske klasifikacije muzike, radi olakšavanja obrade i labeliranja rastućih muzičkih kolekcija. U radu je korišćeno nekoliko različitih klasifikatora, kao i kombinacije klasifikatora, koji su obučavani korišćenjem GTZAN skupa podataka. Prilikom obučavanja pokazalo se da ovaj skup podataka ima nekoliko nedostataka. Osobine audio fajlova korišćene za klasifikaciju su izvedene pomoću MFCC-a i spektrograma. Svi uspešno obučeni klasifikatori su imali približno istu tačnost od oko 60%.

Iako klasifikatori nisu postigli ljudsku tačnost, razlika između ljudi i klasifikatora nije toliko velika i postoji mogućnost da se može nadoknaditi korišćenjem deskriptivnijeg skupa osobina, većih skupova podataka i kompleksniji arhitektura klasifikatora.

Dalji pravci razvoja bi se prvenstveno odnosili predstavu signala, odnosno izvođenje osobina za klasifikaciju iz signala, gde bi se mogle koristiti i neke primitivnije karakteristike poput, boje zvuka (timbre), ritmičkog sadržaja, visine tona (pitch), samostalno ili u kombinaciji sa MFCC i sličnim skupovima koeficijenata. Druga mogućnost je veći i potencijalno preciznije labeliran (mišljenje većine) skup podataka. Na kraju mogle bi se istražiti i upotrebiti složenije arhitekture trenutno korišćenih klasifikatora, kao i njihove kombinacije.

7. LITERATURA

[1] http://marsyasweb.appspot.com/download/data_sets/, Datum pristupa: 25.3.2017.

[2] Muda, Lindasalwa, Mumtaj Begam, and Irraivan Elamvazuthi. "Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques." arXiv preprint arXiv:1003.4083 (2010).

[3] Tzanetakis, George, and Perry Cook. "Musical genre classification of audio signals." IEEE Transactions on speech and audio processing 10.5 (2002): 293-302.

[4] http://zone.ni.com/reference/en-XX/help/371361E-01/lvanls/stft_spectrogram_core/#details, Datum pristupa: 15.8.2018.

[5] Sturm, Bob L. "An analysis of the GTZAN music genre dataset." Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies. ACM, 2012.

[6] Müller, Meinard. *Information retrieval for music and motion*. Vol. 2. Heidelberg: Springer, 2007.

[7] https://ccrma.stanford.edu/~jos/sasp/, Datum pristupa: 15.8.2018.

[8] Mandel, Michael I., and Dan Ellis. "Song-Level Features and Support Vector Machines for Music Classification." ISMIR. Vol. 2005. 2005.

[9] Stanley, Kenneth O., and Risto Miikkulainen. "Evolving neural networks through augmenting topologies." Evolutionary computation 10.2 (2002): 99-127.

[10] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[11] D. Perrot and R. Gjerdigen, “Scanning the dial: An exploration of factors in identification of musical style,” in Proc. Soc. Music Perception Cognition, 1999

[12] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score, Datum pristupa: 15.8.2018.

8. BIOGRAFIJA

Nemanja Rašajski rođen je 1993. godine u Novom Sadu. Osnovnu školu ,,Vasa Stajić“ u Novom Sadu završio je 2008. godine, Gimnaziju ,,Isidora Sekulić“ u Novom Sadu, prirodno-matematički smer završio je 2012. godine. Iste godine upisao je osnovne akademske studije na smeru Računarstvo i automatiku, Fakulteta tehničkih nauka u Novom Sadu. Zvanje diplomirani inženjer elektrotehnike i računarstva stekao je 2016. godine, sa prosečnom ocenom 9.70. Iste godine upisao je master akademske studije na smeru Softversko inženjerstvo i informacione tehnologije Fakulteta tehničkih nauka u Novom Sadu. Uža specijalnost na master studijama bila je inteligntni sistemi.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
Doi: https://doi.org/10.24867/01BE06Mrksic

POREĐENJE ANGULARJS I ANGULAR 5 TEHNOLOGIJE I UPOTREBA BOOTSTRAP BIBLIOTEKE

COMPARISON OF ANGULARJS AND ANGULAR 5 TECHNOLOGY AND THE USE OF THE BOOTSTRAP LIBRARY

Tamara Mrkšić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Objektno-orijentisano programiranje predstavlja jednu od osnovnih programskih paradigmi u računarstvu. Zato tehnologije pokušavaju da poštuju glavne principe objektno-orijentisanog programiranja. U radu je istražena AngularJS i Angular 5 tehnologija. Zatim je izvršeno upoređivanje Angular 5 i AngularJS tehnologije. Objašnjena je upotreba Bootstrap biblioteke.*

Ključne reči: *Objektno-orijentisano programiranje, AngularJS, Angular 5, Bootstrap, AngularJS vs Angular5*

Abstract – *Object-oriented programming is one of the basic programming paradigms in computing. Therefore, technologies try to follow the main principles of object-oriented programming. The paper deals with AngularJS and Angular 5 technology. Then Angular 5 and AngularJS Technology were compared. The use of the Bootstrap library is explained.*

Keywords: *Object-oriented programming, AngularJS, Angular 5, Bootstrap, AngularJS vs Angular5*

1. UVOD

Rad je podeljen u šest poglavlja. Prvih pet poglavlja opisuje problematiku rada iz drugog ugla, kako bi se njihovim spajanjem dobila sveobuhvatna slika teme.

Kroz drugo poglavlje će biti date teorijske osnove, neophodne za problematiku rada. U trećem poglavlju je dato poređenje tehnologija kroz prikaz uvedenih izmena u Angular 5 tehnologiju. Kroz četvro poglavlje će biti opisana Bootstrap biblioteka. U petogom poglavlju je dat zaključak, odnosno sumiranje samog rada. Šestim poglavljem je predstavljena literatura, neophodna za pisanje rada.

Na kraju rada se nalazi kratka biografija autora.

2. TEORIJSKE OSNOVE

U ovom poglavlju će biti opisane teorijske osnove, koje su vezane pre svega za oblast objektno-orijentisanog programiranja. Zatim biće pojašnjeno čemu su namenjani JavaScript, AngularJS, TypeScript, kao i Angular 5.

2.1 Objektno-orijentisano programiranje

U računarstvu, objektno-orijentisano programiranje (OOP) je jedna od programskih paradigmi. Od ranih 1980-tih pa do danas ova paradigma je postala najuticajnija paradigma u komercijalnom razvoju softvera. Programski jezici poput C^{++}-a i Jave programskog jezika su svojom popularnošću utvrdili vodeći status OOP-a kao *de facto* obaveznog pristupa pri razvoju softvera danas, te i jezici koji tradicionalno nisu bili objektno-orijentirani su naknadno dodali OOP koncepte [1].

Osnovni koncepti objektno-orijentisanog programiranja su:

a. Apstrakcija
Osnovni nivo apstrakcije se svodi na uočavanje šablona za objekte sa više zajedničkih osobina. Na osnovu tih šablona se grade klase sa kontrolisanim uključivanjem i isključivanjem nekih detalja o objektima, što je još jedan nivo apstrakcije.

b. Enkapsulacija
Ona predstavlja postupak objedinjavanja stanja i ponašanja objekta u jednu celinu. Tako organizovane objekte je lakše kontrolisati i onemogućiti neovlašćeni pristup. Korisnici sa objektom komuniciraju samo kontrolisano, pomoću javnih metoda i ne mogu neovlašćeno menjati unutrašnja stanja objekta. Samo internim metodama objekta je omogućen pristup stanjima.

c. Nasleđivanje
To je posledica generalizacije kao metoda za modelovanje objekata. Obezbeđuje redukciju i lakše održavanje koda. Tako se zajedničke osobine i funkcionalnosti pišu samo jednom, a njihove eventualne izmene se vrše u osnovnoj klasi.

d. Polimorfizam
To je osobina da se ista metoda osnovne klase izvršava na različite načine u zavisnosti od toga kojoj izvedenoj klasi objekat koji ga poziva pripada. Tako se u izvedenim klasama mogu predefinisati neke od nasleđenih funkcionalnosti u skladu sa specifičnostima izvedene klase. Na taj način je omogućeno da objekti različitih izvedenih klasa reaguju različito izvršavajući istu funkcionalnost osnovne klase.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac, vanr. prof.

2.2 Angular

Bez obzira na vrstu Angular tehnologije, bitno je pojasniti čemu je ona uopšte namenjena. Angular je objavljen kao projekat još 2009. pod nazivom <angular/> , i njegova osnovna namena je bila pomoć web programerima, kao i web dizajnerima da lakše izrade web stranice, koristeći html oznake. Sam naziv Angular potiče upravo od zagrada odnosno <>, koje obuhvataju sve HTML oznake, kao na primer , i druge.

Angular je ustvari JavaScript okvir, koji pomaže programerima da razvijaju svoje aplikacije. Ova biblioteka nudi niz funkcija, koje ga čine trivijalnim za implementaciju kompleksnih zahteva modernih aplikacija, kao što su vezivanje podataka, rutiranje i animacije.

On takođe nudi niz konverzija kako pristupati razvoju aplikacije, što je veoma korisno za velike timove, koji moraju raditi zajedno. Angular predstavlja jedinu JavaScript biblioteku, koja sadrži sveobuhvatni vodič za stil pisanja koda. Dobro radi u form-based aplikacijama, a pogodan je i za velike i složene aplikacije, kao i one koje treba da rade u više razvojnih okruženja [2].

2.3 JavaScript

JavaScript ili skraćeno JS predstavlja objektno zasnovani skriptni jezik. Pored HTML-a i CSS-a, on predstavlja jedan od osnovna tri jezika WWW-a. Omogućava interaktivne web stranice, i zato predstavlja osnovni deo web aplikacija. Velika većina web stranica ga koristi, a web pretraživači imaju i JavaScript mehanizam za njegovo izvršavanja. Uključuje se u web stranice kako bi postale dinamičnije. HTML se koristi samo za oblikovanje i uređivanje elemenata stranice (tekst, forme, linkovi, tabele), ali ne postoji način kako da im se nametne ponašanje. Mogućnost uključivanja JavaScript skripte daje mnogo veću kontrolu kako da se ponaša web stranica.

2.4 AngularJS

AngularJS ili poznatiji pod nazivom Angular.js je JavaScript okvir, baziran na „front-end“ web aplikacijama. Cilj ove platforme jeste pojednostavljenje razvoja i testiranje takvih aplikacija pružanjem okvira za arhitekturu klijent-model-pogled-kontroler (MVC) i model-pogled-pogled-model (MVVM), zajedno sa komponentama, koje se obično koriste u bogatim internet aplikacijama.

Bitno je napomenuti da se pod AngularJS-om podrazumeva ustvari prva verzija i sve njene podverzije Angulara.

Ovaj okvir radi po principu tako što prvo čita HTML stranicu, koja ima dodatne prilagođene tag atribute. Zatim tumači te atribute kao uputstva za povezivanje ulaznih i izlaznih delova stranica sa modelom, koji predstavlja standardne JavaScript promenljive. Vrednosti ovih promenljivih se mogu ručno postaviti u kodu, ili preuzeti iz statičkih ili dinamičkih JSON resursa.

AngularJS je zasnovan na uverenju da se deklarativno programiranje treba koristiti za kreiranje korisničkog interfejsa i povezivanje softverskih komponenti. Sa druge strane, neophodno je imperativno programiranje za definisanje same poslovne logike aplikacije. Samim tim, ovaj okvir produžava i proširuje tradicionalni HTML da bi predstavio dinamički sadržaj putem dvosmernog povezivanja podataka, koji omogućava automatsku sinhronizaciju modela i prikaza. Kao rezultat, AngularJS naglašava eksplicitnu manipulaciju DOM stabla sa ciljem povećanja performansi i testiranja.

Ciljevi dizajna AngularJS uključuju:

- razdvajanje DOM manipulacije od logike aplikacije (Na težinu dosta utiče sama struktura koda.)
- odvajanje klijentske strane od servera (Ovo omogućava ravojnim timovima da paralelno napreduju i omogućavaju ponovnu upotrebu obe strane.)
- obezbeđivanje strukture za putanju izgradnje aplikacije (Ovo podrazumeva dizajniranje UI-ja, pisanje poslovne logike i testiranje.) [3]

On pojednostavljuje razvoj aplikacija predstavljanjem višeg stepena apstrakcije programeru. Kao i svaka apstrakcija, ona dolazi po cenu fleksibilnosti. Drugim rečima, nije baš svaka aplikacija najbolja za AngularJS, jer je on napravljen uz primenu CRUD aplikacije. Međutim, na sreću većina web aplikacija predstavlja upravo CRUD aplikacije. Da bi se bolje razumelo za šta je dobar, najlakše je dati primere za šta nije. Igre i GUI uređaji su primeri aplikacija sa intenzivnom i neobičnom manipulacijom DOM staba. U takvim slučajevima je bolje koristiti neku biblioteku sa nižim stepenom apstrakcije, kao što je JQuery.

2.5 TypeScript

TypeScript predstavlja programski jezik, namenjen za izgradnju web aplikacija, kojeg razvija i održava Micro-Soft. Njegova sintaksa je veoma slična sintaksi Java-Script-a, s tim da su dodate razne napredne mogućnosti. Jedna od njih je opciono statično pisanje, koje olakšava pisanje velikih web aplikacija. Statično pisanje zahteva dodeljivanje tipa promenljivima, koje se kreiraju. JavaScript je sam po sebi dinamički pisan jezik, u kojem se ne mora unapred dodeliti tip promenljivoj. Sa statičkim pisanjem, kompajler može detaljnije da prikaže i da uvid u grešku, ukoliko do nje dođe. Nastao je zbog nedostataka JavaScript-a za razvoj velikih aplikacija od strane Micro-Soft-a i njegovih klijenata. Izazovi sa kompleksnim Java-Script kodom doveli su do potražnje za prilagođenim alatima da bi se olakšao razvoj komponenti u jeziku.

2.6 Angular 5

Angular 5 predstavlja novu verziju AngularJS okvira, koju je razvio Google. Predstavljen je kao kompletno prepisan AngularJS, ali sa različitim poboljšanjima. Ta poboljšanja se ogledaju u optimizovanijoj građi i bržem kompajliranju. On je JavaScript okvir, koji koriste

programeri širom sveta za izgradnju web, desktop i mobilnih aplikacija. Takođe, Angular 5 kombinuje deklarativne „template“-ove, „dependecy injection“ i integrisane najbolje prakse za rešavanje raznih razvojnih problema.

Mnogobrojne web platforme, kao što su Google Adworbs, Google fiber, Adsense i druge koriste Angular za izgradnju korisničkog interfejsa.

3. POREĐENJE ANGULARJS I ANGULAR 5 TEHNOLOGIJE

Novine, koje je uneo Angular 5, je najbolje shvatiti kroz njegovo poređenje sa AngularJS-om. Samim tim, u ovom poglavlju će biti opisane razlike između AngularJS i Angular 5 tehnologije, prolazeći kroz poglavlja koja predstavljaju njihove razlike. Angular 5 predstavlja noviju verziju Angulara, i samim tim je doneo dosta različitih promena u pogledu funkcionalnosti, kao i određenih novina. Date promene će detaljnije biti opisane u nastavku.

3.1 TypeScript

TypeScript ustvari predstavlja jezik, kao i set alatki za generisanje JavaScripta. On predstavlja projekat otvorenog koda, koji pomaže programerima da pišu velike JavaScript projekte.

On ustvari generiše JavaScript. Umesto da zahteva potpuno novo radno okruženje, JavaScript generisan preko TypeScripta, može ponovo da upotrebi sve postojeće alatke JavaScripta, radne okvire i biblioteke, koje su dostupne za JavaScript.

Međutim, TypeScript jezik i kompajler približavaju razvoj JavaScripta tradicionalnijem objektno-orijentisanom programiranju.

3.2 Progresivne web aplikacije

Trenutni razvoj progresivnih web aplikacija je razrađen i složen proces. Potrebno je voditi računa kako tokom razvoja, tako i u upotrebi da ne dođe do greške ili da nijedna starija verzija ne bude oštećena. Međutim ovde je došlo do promene u Angularu 5.

Razvoj progresivnih web aplikacija treba toliko da se pojednostavi da mogu podrazumevano da se kreiraju. U ovom slučaju korisnici, kao i programeri bi trebali imati jednake koristi od ovog. Uz Angular CLI, Angular 5 ima mogućnost samostalnog kreiranja konfiguracije i koda.

3.3 Komponente

Komponenta ustvari predstavlja dekorator, koji se mapira na klasu kao Angular komponentu. Ona pruža metapodatke konfiguracije kako se komponenta treba obrađivati, instancirati, i koristiti u toku izvršavanja.

Komponente čine najosnovniji UI Angular aplikacije. Aplikacija ustvari predstavlja stablo komponenti. One su podgrupa direktiva, koje su uvek povezane za neki šablon. Za razliku od drugih direktiva, samo jedna komponenta se može instancirati po elementu u šablonu.

3.4 Podrška za multi alijase

U Angularu 5 je moguće davati više imena komponentama i direktivama prilikom eksporta. Eksport komponente sa više imena može pomoći korisnicima da migriraju bez prekidanja postojećeg koda.

3.5 Rad sa događajima

Ukoliko je potreban rad sa događajima, to je omogućeno kroz korišćenje interpolacije, vezivanje osobina, vezivanje podataka, kao i vezivanje događaja. Na drugačiji način je implementirano u Angularu 5, u odnosu na AngularJS tehnologiju.

3.6 Forme

U Angularu 5, kada su u pitanju forme, poboljšane su performanse i moguće je specificirati kada je potrebno izvršiti validator u formi. Svaki put kada se promeni vrednost u „FormControl“, validacija će se izvršiti shodno tome, i to potencijalno prilikom svakog pritiska na tastaturu. U slučaju složenih validacija, to može dovesti do veoma loših performansi. Sa novom opcijom „UpdateOn“, Angular 5 omogućava preciznije određivanje kada se validacija treba izvršiti. Tu se može izabrati promena, koja određuje prethodno ponašanje, potvrđivanje forme i „blurovanje“.

3.7 HttpClient API

Pre verzije Angulara 5 za HTTP zahteve je korišćen „@angular/http“ u Angular aplikacijama. Međutim, sa verzijom 5, ovakvo korišćenje je postalo zastarelo. Umesto njega se koristi HttpClient API iz „@angular/common/api“, i on je preporučen za upotrebu u aplikacijama.

3.8 RxJS

Reaktivno programiranje je paradigma asinhronog programiranja, koja se odnosi na tokove podataka i širenje promena. RxJS je biblioteka za reaktivno programiranje pomoću „posmatrača“, što olakšava sastavljanje asinhronog poziva ili poziva zasnovanog na odgovoru. RxJS obezbeđuje implementaciju tipa „Obeservable“, koji je potreban sve dok tip ne postane deo jezika i dok ga pretraživači ne podrže.

3.9 Rutiranje i moduli

Ruter predstavlja opcioni servis, koji predstavlja određeni prikaz komponente za datu URL adresu. Rutirana Angular aplikacija ima samo jedan primerak instanciranog Router servisa. Kada se URL adresa pregledača promeni, taj ruter traži odgovarajuću rutu, iz koje može odrediti komponentu, koja bi trebalo da se prikaže. U okviru rutera ne postoje neke podrazumevane rute. Sve rute je potrebno konfigurisati.

3.10 Animacije

Paket animacija u Angularu 5 je proširen sa nekoliko sintaksnih elemenata. Sada je moguće reagovati na numeričke promene vrednosti primenom inkrementa i dekrementa, kako bi animirali u skladu sa tranzicijom.

Takođe, u Angularu 5 se animacije mogu aktivirati i deaktivirati korišćenjem vrednosti, koje su povezane sa vezivanjem podataka. Tako na primer kada je u pitanju događaj „disabled“ ukoliko je on „okinut“, samim tim se može pozvati odgovarajuća animacija.

4. UPOTREBA BOOTSTRAP BIBLIOTEKE

Bootstrap predstavlja najpopularniji HTML, CSS i JavaScript okvir za web „front-end“ razvoj aplikacija. Takođe čini odlično rešenje za razvoj odzivnih, „mobile-first“ web stranica. Bootstrap okvir se može koristiti zajedno sa JavaScript web okruženjem, kao i mobilnim okvirima kao što je na primer Android. On uključuje HTML i CSS zasnovane dizajnerske šablone za tipografiju, dugmiće, forme, tabele, navigaciju, modele i mnoge druge, kao i JavaScript dodatke. Ti dodaci omogućavaju lako kreiranje željenog dizajna. Ova biblioteka takođe omogućava kreiranje prilagodljivog dizajna. Takva vrsta dizajna dozvoljava kreiranje web aplikacija, koje se automatski prilagođavaju kako bi dobro izgledale na svim uređajima, od malih telefona do velikih desktop računara.

Kada je Bootstrap u pitanju, izdvajaju se njegove glavne prednosti, a to su:

- jednostavan za upotrebu – potrebno je samo osnovno znanje HTML-a i CSS-a,
- prilagodljive funkcije – odziv CSS-a se prilagođava prema vrsti uređaja, na kom se koristi,
- „mobile-first“ pristup – od verzije 3, „mobile-first“ stilovi predstavljaju deo osnovnog okvira,
- kompatibilnost ka pretraživačima – u te pretraživače spadaju: Chrome, Firefox, Internet Explorer, Edge, Safari i Opera [4].

Postoji više načina kako „doći“ do Bootstrapa prilikom razvoja web aplikacije, a to su njegovim preuzimanjem ili uključivanjem od CDN-a. Ukoliko se koristi način uključivanja, kao rezultat će se učitavati iz keš memorije kada se pokrene aplikacija, što dovodi do bržeg vremena učitavanja. Takođe, većina CDN-ova će se pobrinuti da kada se od njega zatraži datoteka, omogućiće pristup ka najbržem serveru, što opet vodi do bržeg vremena učitavanja. Bootstrap koristi JQuery za JavaScript dodatke kao što su modalni dijalozi i slično. Međutim, ako se koristi Bootstrapov CSS, onda nije potreban JQuery.

5. ZAKLJUČAK

U ovom radu vršeno je upoređivanje AngularJS i Angular 5 tehnologije. Takođe je opisana u najbitnijim crtama Bootstrap biblioteka.

Promena, koja je obeležila Angular 5 pre svega predstavlja uvođenje TypeScripta, i na njemu se on temelji. On predstavlja jezik, koji generiše JavaScript, i zajedno sa kompajlerom približava razvoj JavaScripta tradicionalnijem objektno-orijentisanom programiranju. Zatim omogućava razvoj progresivnih web aplikacija i pruža podršku za multi alijase. Izgradnja projekta pomoću komponenata znatno olakšava pisanje koda, i njegovo ponovno korišćenje. Takođe, dosta promena je vezano za rad sa događajima i u formama. Angular 5 omogućava razvoj aplikacija korišćenjem HttpClient API-a i reaktivnim programiranjem. Osim toga, promene su nastale i u samom načinu rutiranja i animacijama.

Bez obzira koja je verzija Angulara u pitanju, jedna od najpoznatijih biblioteka, koja se koristi za uređivanje HTML stranica uz njega, je Bootstrap. Ono što se moglo zaključiti jeste da se primenom ove biblioteke dobija savremeni dizajn, „user-friendly“ izgled aplikacije, kao i na određeni način očekivano ponašanje. To je posebno bitno korisnicima, koji su do tada koristili neku sličnu aplikaciju, i samim tim će se lakše prilagoditi novoj, ako se poštuju određeni šabloni prilikom izrade. Na taj način se dobija aplikacija, koja je „lepa na oko“, a sa druge strane prilagođena i lakša za korišćenje.

Rad je propraćen sa dosta teorije, koja je bila neophodna za ulazak dublje u samu problematiku teme rada. Kada je u pitanju generalno Angular tehnologija, moglo se zaključiti da predstavlja odličan okvir i pruža dosta mogućnosti za izradu „front-end“ aplikacija. Kroz razlike u verzijama se uvidelo koliko je ustvari bilo „šupljina“ u samom AngularJS-u, koje je popunio Angular 5. Uz povećavanje mogućnosti, koje do tada nisu bile, uvedena su i poboljšanja postojećih funkcionalnosti. Takvim radom je postignuta mogućnost lakše izrade velikih aplikacija, bolje organizacije koda, bržeg pokretanja aplikacije i eventualnih otklanjanja nastalih grešaka tokom izrade.

6. LITERATURA

[1] http://www.stroustrup.com/whatis.pdf (pristupljeno u avgustu 2018.)

[2] https://developer.telerik.com/topics/webdevelopment/-what-is-angular/ (pristupljeno u avgustu 2018.)

[3] https://docs.angularjs.org/ (pristupljeno u avgustu 2018.)

[4] https://www.w3schools.com/bootstrap/bootstrap_get_-started.asp (pristupljeno u avgustu 2018.)

Kratka biografija:

Tamara Mrkšić rođena je u Novom Sadu 1994. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Soft computing odbranila je 2017.god.
kontakt: tamarica94mrksic@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE07Odalovic

APLIKACIJA ZA PRAVLJENJE MOCKUP-A I NJENA PRIMENA

MOCKUP CREATION APPLICATION AND ITS USAGE

Mirko Odalović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj članak treba da demonstrira probleme koji se javljaju tokom implementacije aplikacije za pravljenje* ***mockup-a*** *kao i način na koji su spomenuti problemi rešeni. Takođe pojasniće se i tehnologije u kojima je implementirana aplikacija kao i pojam* ***mockup****.*

Ključne reči: ***Mockup-a****, WPF, MVVM, Dizajn sabloni*

Abstract – *This article should demonstrate problems that appear during implementation of application for creating mockups and a way in which the following problems are solved. Technologies used for application implementation and term mockup will also be explained.*

Keywords: ***Mockup****, WPF, MVVM,* ***Design patterns***

1. UVOD

U članku se opisuje način na koji je napravljena aplikacija za pravljenje ***mockup-a*** kao i primena aplikacije. U prvoj celini pojašnjeni su pojmovi čije razumevanje je neophodno kao bi se znalo za šta aplikacija služi. Pojam koji se objašnjava u prvoj celini je ***mockup*** jer je ključni pojam čije razumevanje je neophodno za uspešno praćenje i razumevanje ovog članka.

U drugoj celini opisan je način komunikacije pomoću ***mockup-a*** izmedju dizajnera i programera, to su dve uloge koje imaju najviše dodirnih tačaka sa ***mockup-om***.

U trećoj celini opisana je sama ***mockup*** aplikacija odnosno funkcionalnosti koje podržava, kako bi svojom bogatom ponudom funkcionalnosti privukla što više korisnika a isto tako ih i zadžala.

U četvrtoj celini opisani su dizajn šabloni, koji se koriste u implementaciji aplikacije, da bi se mogla pratiti demonstracija implementacije programskog rešenja.

U petoj celini demonstrirane su tehnologije u kojima je razvijena aplikacija za pravljenje ***mockup-a***. A to je *WPF* aplikacija gde se prilikom implementacije vodi računa da bude podržan *MVVM* šablon.

U poslednjoj šestoj celini opisani su problemi koji se javljaju tokom imaplementacije kao i način na koji su isti rešeni. Isto tako u ovoj celini je demonstrirana primena dizajn šablona na spomenute probleme.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac, vanr. prof.

2. MOCKUP

U jednoj reči ***mockup*** se predstavlja kao skica, pomoću koje se opisuje izgled korisničkog interfejsa aplikacije. ***Mockup*** veoma često predstavlja nacrt dizajna ili čak stvarni dizajn aplikacije. Dobro kreiran ***mockup*** dobro reprezentuje strukturu informacija, vizualizaciju sadržaja i demonstraciju osnovnih funkcija na statičan način. Izgradnja ***mockup-a*** pomaže ostvarivanju idealne ravnoteže izmedju krajnjeg korisničkog interfejsa i jednostavnoj modifikaciji istog. Danas se ***mockup*** može brzo napraviti u poređenju sa vremenom potrebnim da se napravi aplikacija. Koristan je u fazi kada je potrebno osmisliti korisnički interfejs aplikacije, pomaže da osobe zadužene za kreiranje korisničkog interfejsa budu skoncentrisane samo na svoj deo posla, samim tim poboljšava kreativnost. ***Mockup*** je naročito koristan kada je potrebno na brz način privući potecijalne klijente, jer kada se klijentu ponudi nešto vizualno(konkretno), na njega će to ostaviti bolji utisak nego da se ponudi sama ideja. Na početku razvoja aplikacije potroši se više vremena kada se prave ***mockup-i***, ali se zato puno vremena uštedi u budućnosti jer nakon uspešno napravljenih ***mockup-a*** zna se tačno šta se radi i zna se da je to predstavljeno klijentu i da se klijent složi sa tim rešenjem. Uobičajena mera koja se koristi kako bi pokazala koliko su napravljeni ***mockup-i*** precizni ili tačni naziva se *vernost* (eng. ***fidelity***). Vernost je mera koliko je ***mockup*** sličan krajnjem proizvodu. Nisku vernost imaju ***mockup-i*** koji predstavljaju jednostavne skice. Visoku vernosti imaju ***mockup-i*** koji savršeno predstavljaju korisnički interfejs aplikacije nazivaju se još i piksel savršeni. Kada se prave ***mockup-i*** sa visokom vernosti treba imati na umu da je za takve ***mockup-e*** potrebno dosta više vremena kako bi se napravili, nego što je potrebno da se naprave ***mockup-i*** sa niskom vernosti. Zato se često prave ***mockup-i*** sa srednjom vernosti, jer su dovoljno verni da se na osnovu njih može napraviti korisnički interfejs aplkacije, a opet su dovoljno jednostavni da se mogu brzo napraviti.

3. MOCKUP KAO SREDSTVO KOMNIKACIJE

Da bi faze u životnom ciklusu razvoja softvera kroz koje je potrebno proći kako bi se stiglo do finalnog proizvoda odvijale što lakše i bez velikih zastoja potrebno je obezbediti dobru komunikaciju između ljudi koji učestvuju u tim fazama, a isto tako je važno da bude dobra komunikacija izmđu ljudi koji pripadaju raličitim fazama. U nastavku spominju se samo dve faze životnog ciklusa razvoja softvera jer u tim fazama se javalju uloge koje su od ineteresa odnosno uloge koje imaju najviše dodirnih tačaka sa ***mockup-om*** a to su uloga dizajner i uloga programer.

U fazi dizajna potrebno je osmisliti korisnički interfejs buduće aplikacije. Pravljenje korisničkog interfejsa aplikacije je posao dizajnera, znači da se uloga dizajnera javalja u fazi dizajna životnog ciklusa razvoja softvera. U fazi razvoja potrebno je razviti samu aplikaciju. Posao programera je da razvije aplikaciju, znači da se uloga programera pojavljuje u fazi razvoja životnog ciklusa razvoja softvera.

Grafiči dizajneri su obučeni da koriste računarske programe za stvaranje slike, kao što su logo ili sam dizajn aplikacije. Dizajneri za cilj imaju stvaranje efikasnog i atraktivnog korisničkog interfejsa, kako bi korisnici koji koriste njihov korisnički interfejs bili zadovoljni. Prvenstveno dizajneri dizajniraju raspored grafičkih komponenti na korisničkom interfejsu(tj. poziciju grafičkih komponenti) pored toga odlučuju o boji, fontu, stilu i veličini grafičkih komponenti, kako bi dobili željen korisnički interfejs. Posao dizajnera je: zna da koristi aplikacije za dizajniranje rasporeda elemenata i uređivanje slika, predstavlja svoje projekte klijentima ili poslodavcima, uvaži promene koje su mu tražili klijenti i poslodavci i bude u toku sa najnovijim tehnologijama i softverom.

Programeri koriste programske jezike pomoću kojih pišu kod za aplikacije kako bi obavili svoj deo posla. Obično poznaju jedan ili više programksih jezika kao što su *Java*, *C#*, *C++*, itd. Njihov posao nije samo da razvijaju programe(aplikacije) od nule, već treba da znaju na postojećim aplikacijama koje nisu oni razvijali da rešavaju probleme ili dodavaju nove funkcionalnosti ako klijent to zahteva, tj. da se lako i brzo snalze u drugim programskim rešenjima koje oni nisu razvijali. Programeri oživaljvaju projekat, od dokumentacije i ***mockup-a*** koji su napravljeni u ranijim fazama, programeri kodiranjem prave konkretan proizvod koji korisnicima omogućava da ga primene na svoje problem zbog kojih je i sam proizvod napravljen. Posao programera je: održavanje i ažuriranje programa koji već postoje, poznavanje velikog broja programskih biblioteka kako bi brže i lakše mogao da radi, testranje razvojnog koda i dizajniranje i potvrda funkcionalnosti koda.

Nakon objašnjenja uloga programera i dizajnera vraćamo se na samu temu ***mockup*** kao sredstvo komunikacije između dizajnera i programera. Tradicionalno se posao programera i dizajnra gleda skroz odvojeno ali da bi razvili dobru aplikaciju neophodna je saradnja. Na kraju svakog dana i programer i dizajner rade da bi ostvarili isti cilj, da su korisnici krajnje aplikacije zadovoljni, a ako su oni zadovoljni onda je i klijent koji je od njih tražio da napravi aplikaciju takođe zadovoljan. Da bi ostvarili svoj cilj aplikacija mora da radi, a to je u nadležnosti programera, isto tako mora da bude prilagođen korisnički interfejs kao i jednostavna interakcija korisnika sa aplikacijom što je u nadležnosti dizajnera tj. mora dobro da dizajnira aplikaciju. Komunikacija između programera i dizajnera upotrebom ***mockup-a*** se pokazala jako dobro, kao što je poznato slika govori više od hiljadu reči. Dizajner ima ideju ali koliko bi njemu trebalo vremena da tu ideju prenese programeru, može se desiti da programer u glavi stvori sasvim drugu sliku. Opšte je poznato programeri razmišljaju na različite načine, programer se već sretao sa nekim problemima i on će nastojati da ako je to moguće ponudi rešenja koja je već imao, da bi mu bilo lakše da uradi svoj deo posla. Dok dizajneri ima svoj pogled na svet i on nije svestan koliko je nešto teško za programera, dizajner nema ta ograničenja. Dizajner je zadužen za ***mockup***, i taj ***mockup*** se prosleđuje u vidu slike programeru, programer na osnovu slike zna koja je ideja dizajnera i može ideju da implementira, na ovaj način se smanjuje šansa za nesporazum u komunikaciji.

4. MOCKUP APLIKACIJA

Aplikacija za pravljenje ***mockup-a*** dizajneru treba da pomogne kao što samo ime kaže u pravljenju ***mockup-a***, na kraju tako napravljen ***mockup*** može da izveze u sliku, i da u formatu slike može da pošalje klijentu, programeru. Takođe aplikacija treba da omogući korisniku snimanje projekta koji je započeo u samoj aplikaciji kao bi posle mogao da nastavi sa radom, kako ne bi izgubio sve prethodno što je napravio. A isto tako kada šalje drugom dizajneru bolje da pošalje sam fajl od projekta kako bi dizajner kome je poslat fajl mogao da učita postojeći projekat i ukoliko želi da doda neke svoje ideje, može da doradi sam ***mockup*** bez da mora sve ispočetka već samo da nastavi na već postojećem projektu.

Korisnicima aplikacija treba da bude omogućeno dodavanje slika na nešto što bi trebalo da predstavlja ***mockup***. Pored slika korisnici mogu da ubacuju i tekstualni sadržaj na svoje ***mockup-e***, kako bi mogli da prave željene ***mockup-e***. Još jedna bitna stvar koja je potrebna korisnicima je rad sa slikama i tekstovaim, korisnik u svakom trenutku može da promeni sliku ili tekstualni sadržaj gde god želi upotrebom miša. Korisniku treba da budu prikazane predefinisane slike koje može da koristi za pravljenje svojih ***mockup-a***, pored predefinisanih slika korisnik trebalo da ima mogućnost dodavanja svoje slike koja se ne nalazi u predefinisanim slikama.

Što se tiče tekstualnog sadržaja korisnik može da podešava sam tekstualni sadržaj kako on želi, aplikacija treba da podržava promenu boje, veličine i fonta tekstualnog sadržaja kao i promenu samog sadržaja. Kako bi korisnik moga preciznije da pozicionira slike i tekstualni sadržaj na ***mockup-u***, korisnik ima mogućnost povećavanja i smanjivanja samog ***mockup-a***. Takođe korisnik u svakom trenutku može da poništi akcije koje je napravio, preko funkicije ***unod*** koja se nalazi u aplikaciji, a ima i funkciju ***redo*** kako bi korisnik mogao da opozove akciju.

Da bi korisnik mogao lakše da radi sa slikama i tekstualnim sadržajem obezbeđene su funkcije: ***cut***, ***copy***, ***paste***. Ukoliko je korisniku potrebna slika ili tekst koji je već koristio u projektu, ubačene su funkcije koje korisnik može da iskoristi za kopiranje istih slika ili tekstova, i ne mora da radi isti posao više puta. Korisnik ima mogućnost da postavi boju pozadine kao i mogućnost postavke željene slike za pozadinu ***mockup-a***.

Korisnik može da menja dimenzije samog ***mockup-a*** u zavisnosti od okruženja i zahteva samog klijenta. Ako korisnik želi da obriše neki tekstualni sadržaj ili sliku koju je prethodno dodao na ***mockupa***, aplikacija to obezbeđuje korisniku. Korisnik prilikom dodavanja nove slike u ***mockup*** ima mogućnost pregleda slike u prozoru koji je namenjen za pregled slike. U aplikaciji korisnik imam mogućnost da napravi novi projekat.

5. STATE I KOMAND DIZAJN ŠABLON

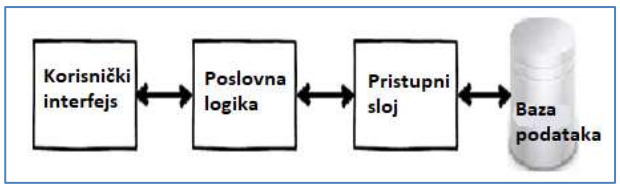
State je dizajn šablon ponašanja koji imeplementira mašinu stanja na objektno orjentisan način. Sa ***state*** šablonom, mašina stanja se implementira primenom svakog pojedinačnog stanja koje implementira interfejs koji predstavlja stanja i imeplementira prelazak u sledeće stanje. ***State*** šablon može se tumačiti kao strategiski obrasac koji je u stanju da promeni trenutnu strategiju kroz pozive metoda koje su definisane u interfejsu stanja. ***State*** šablon se najčešće koristi u programiranju kako bi se enkapsuliralo različito ponašanje na osnovu njegovog unutrašnjeg stanja. Predstavlja čist način da objekta promeni svoje ponašanje bez pribegavanja uslovnim izrazima i na taj način se olakšava kasnije održavanje.

U objektno orjentisanom programiranju, komandni šablon predstavlja dizajn šablon ponašanja u kojem se objekat koristi za enkapsuliranje svih informacija potrebnih za izvođenje akcije ili pokretanje dogadja kasnije. Ove informacije uključuju imena metoda, objekata koji poseduju metode kao i vrednosti parametara koje treba proslediti metodama.

6. WPF i MVVM

Windows Presentation Foundation (WPF) je sistem nove generacije za prezentaciju i izgled ***Windows*** klijentskih aplikacija sa vizuelnim boljim korisničkim iskustvom. Pomoću *WPF-a* može se kreirati širok opseg samostalnih aplikacija i aplikacija koje se izvršavaju u internet pretraživaču. Samo jezgro *WPF-a* je nezavisno od rezolucije i vektorkog načina crtanja, napravljeno da iskoristi prednosti modernih grafičkih procesora. *WPF* proširuje jezgro sa sveobuhvatnim skupom funkcija razvoja aplikacije koje uključuju ***Extensible Application Markup Language*** (XAML), kontrole, povezivanje podataka, *2D* i *3D* grafici, animacije, stilove, šablone, dokumente, medije i teks. *WPF* je uključen u ***Microsoft*** *.NET* platformu, tako da se mogu napraviti aplikacije koje sadrže u sebi elemente iz *.NET* platforme.

Život je razvoj koji počinje kada se čovek rodi, on vremenom uči i razvija se kako bi od deteta postao odrasla osoba. Isto važi i za arhitekturu softvera, u početku se radi sa osnovnim strukturama a zatim se razvija i napreduje prema traženim zahtevima i potrebama. U ***Model View View Model*** (*MVVM*) arhitekturi projekat je podeljen u tri logičke celijne: korisnički interfejs (*UI*), poslovni sloj (eng. ***Bisiness Logic***) i pristupni sloj (eng. ***data access layer***). Svaki od ovih slojeva odgovoran je za svoj deo posla. Na slici 1 prikazana je *MVVM* arhitektura.

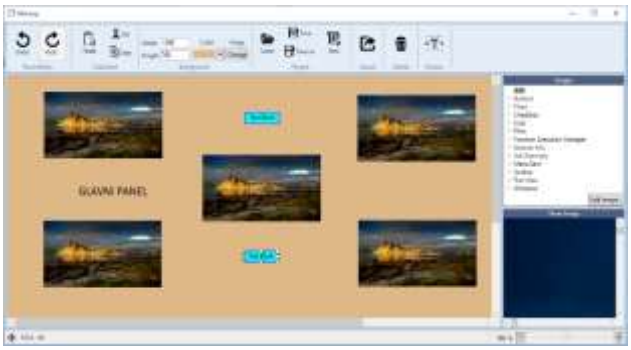


Slika 1. *MVVM arhitektura*

Korisnički interfejs zadužen je za prezentaciju podataka, poslovni sloj vodi računa o validaciji podataka i sloj za pristup podacima zadužen je za rukovanje sa bazom podataka. Prednosti troslojne arhitekture: zadržava promene (promene u jednom sloju ne utiču na druge slojeve), ponovna iskoristivost (povećana je ponovna iskoristivost jer su svi slojevi odvojeni, samostalni i pojedinačni delovi).

7. IMPLEMENTACIJA APLIKACIJE

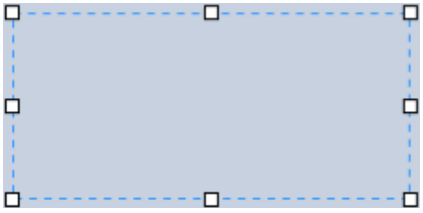
Aplikacja je pisana u programsko jeziku *C#*, pomoću *WPF-a* i primenom *MVVM* šablona. Na slici 2 prikazan je korisnički interfejs ***mockup*** aplikacije.



Slika 2. *Korisnički izgle mockup aplikacije*

Glavni panel, samo ime kaže, predstavlja glavni deo ove aplikacije i on prekriva veći deo korisničkog interfejsa. Pomoću glavnog panela se prikazujue korisniku ***mockup***, isto tako se omogućuje da pravi ili menja već postojeće ***mockup-e***. Jedna od glavnih funkcionalnosti koje korisnik može da radi je ubacivanje druge komponente u glavni panel. Glavni panel je trebalo da podrži rad sa više komponenti različitog tipa npr. može da primi liniju, pravougaonik, neku tekstualnu komponentu i neku grafičku komponentu unutar sebe. Da bi se podržao rad sa više različitih komponenti napravljena je klasa koja predstavlja baznu komponentu koja sadrži sve osobine, metode i komande koje se neophodne glavnom panelu da bi mogao da podrži bazne funkcije za komponente kao što je, na određenim pozicijama da iscrta komponentu, da zna koliko su komponente visoke i široke. I sada kada postoji ova bazna komponenta lako mogu da se prave nove komponente. Da bi se napravila nova komponenta potrebno je samo naslediti baznu komponentu i dodati osobine i metode koje su karakteristične za komponentu koja se pravi. I naravno za svaku komponentu potrebno je definisati ***View***, da bi glavni panel znao da nacrta komponentu. Ovakvom implementacijom obezbeđeno je lako dodavanje novih komponenti.

Selekcija komponenti, korisnik može da selektuje željenu komponentu koja se nalazi na glavnom panelu. Da bi selekcija bila podržana napravljena je grafička komponenta koja se iscrtava oko komponente koja treba da bude selektovana i predstavlja selekciju. Komponenta koja predstavlja selekciju se sastoji od više drugih grafičkih komponenti, tako da je napravljena grafička komponenta koja predstavlja četvorougao. Komponenta za selekcijiju sadrži devet komponenti četvorougla, kao što se vidi na slici 3.



Slika 3. *Grafička komponenta za selekciju*

Grafička komponenta se sastoji od jednog glavnog pravougaonika i osam malih kvadrata koji se nalaze na uglovima i polovinama stranica glavnog pravougaonika.

Pomeranje i promena dimenzije selektovane komponente, pored selekcija korisnik može i da pomera i menja dimenziju selektovane komponente. Pošto ove dve funkcionalnosti trebaju da se pozivaju na isti događaj, tj. kada korisnik pritisne levi klik i krene da pomera miš a i dalje nije pustio klik, onda bi trebalo u zavisnosti gde je korisnik pritisnuo klik na nekoj selektovanoj komponenti da se dogodi pomeranje komponente ili promena dimenzije. Ova funkcionalnost je implementirana upotrebom ***state*** šablona. Za svaki četvorougao koji se nalazi u komponenti za selekciju postoji jedno stanje, jer kada se klikne na bilo koji od tih četvorouglova različite operacije treba se izvrše nad komponentom koja je selektovana. Ako se klikne na glavni pravougaonik onda treba da se pomera selektovana komponenta kada se pomera miš ako klik nije pušten. A ako korisnik pritisne levi klikne na neko od kvadrata onda u zavisnosti na koji je pritisnu menja se dimenzija ili pozicija selektovane komponente. Npr. ako korisnik klikne na gornji levi kvadrat otiće se u ***LeftTopResizeState*** stanje, u tom stanje je definisano šta treba da se desi sa komponentom kada korisnik pomer miš. U ovom slučaju komponente treba da menja dimenzije a isto tako treba i da joj se menja pozicija jer se koridinate računaju u četvrtom kvandrantu.

Slikovna komponenta, korisnik na glavni panel može da dodaje slkovnu komponentu. Ova komponenta ko i sve do sad komponente da bi se napravila mora da nasledi baznu komponentu, dodatna osobina koju poseduje ova komponenta je putanja do slike koja treba da se iscrta unutar ove komponente. I kao što je već rečeno treba napraviti i izgled same komponente i sada na brz i jednostavan način je napravljena nova komponenta.

Tekstualna komponenta, korinik na glavni panel može da dodaje i tekstualnu komponentu. Isti je postupak pravljenja bio kao kod slikovne kompnente stim da ova komponenta ima dve dodatne osobine a to su sam tekst i još jednu osobinu u kojoj se nalaze sve neophodne informacije za sam izgled teksta kao što su font, boja, veličina, da li treba boldvati tekst.

Undo, ***Redo*** funkcije, omogućuju korisniku da poništava sve akcije koje je stvarao dok je pravio ***mockup*** na glavnom panelu. Isto tako ako je već poništio neke akcije može da ih opozove. Prilikom implementacije ovih funkcionalnosti korišten je komandni šablon. Sve akcije koje korisnik može da uradi na glavnom panelu su implementirane preko komandi. Komande koje postoje su: komanda za promenu pozicije i dimenzije komponente, komanda za dodavanje, komanda za brisanje, komanda za isecanje, komanda za nalepljivanje, komanda za promenu fonta tekstualnoj komponenti.

Mehanizam zumiranja, korisnik ima mogućnost da uveća glavni panel ako mu je sadržaj koji se nalazi u glavnom panelu sitan, isto tako ima mogućnost da uveća glavni panel ako to želi. Ovaj mehanizam je implementiran upotrebom ***state*** šablona. Svako stanje u sebi definiše koje je sledeće a koje prethodno stanje i određuje parametre koji definišu trenutno stanje. Korisnik ima mogućnost da vidi glavni panel u sledećim razmerama: 25, 33, 50, 67, 75, 80, 90, 100, 125, 150, 175, 200, 250, 300, 400 i 500. I za svaku ovu razmeru postoji posebno stanje, stanje 100 predstavlja redovan prikaz glavnog panela.

Čuvanje i otvaranje projekta, korisnik ima mogućnost da sačuva projekat koji trenutno radi, kasnije kada želi može da ga otvori i da nastavi gde je stao. Kako bi se ovo implementiralo bilo je potrebno da sve klase koje se snimaju budu serijabilne odnostno anotirane anoticajom „***Serializable***“, kao i sve osobine koje se u njima nalaze. Problem koji se ovde javalje jeste to što klase koje su u sebi imale osobine koje nisu serijabilne nisu mogle da se snime u fajl, a ne postoji mogućnost da te osobine postanu serijabilne. Način na koji je ovo rešen je da su dodate nove osobine koje su serijabilne i u koje se prilikom snimanja u fajl kopiraju informacije koje sa nalaze u osobinama koje nisu serijabilne, da bi se posle kada se čita iz fajla postojala informacije da bi se kreirali ti objekti koji nisu bili serijabilni.

8. ZAKLJUČAK

U ovom članku demonstrirana je implementacija aplikacije za pravljenje ***mockup-a***. Pored same aplikacije prikazana je i primena aplikacije, kojim ulogama u životnom razvoju softvera ova aplikacija pomaže. I na koji način te uloge mogu da iskoriste aplikaciju kako bi rešili svoje probleme. Isto tako opisani su glavni pojmovi zbog kojih je i sama aplikacija napravljena kako bi se razumeo rad aplikacije kao i sam primena. Odluka da se pravi *WPF* aplikacija prateći *MVVM* šablon u programskom jeziku *C#* se pokazala kao dobra jer su uspešno i kvalitetno rešeni svi problem na koje se nailazilo u toku imlementacije. Kao i većina današnjih aplikacija i ova aplikacija se može poboljšati, jedan od predloga za poboljšanje je da se korisniku omogući da pretražuje slike pomoću meta podataka, kako bi brže pronašao željenu sliku koju želi da stavi na glavni panel, a ne kako je sada slučaj da korisnik mora da prođe kroz sve slike dok ne nađe sliku koju želi.

9. LITERATURA

[1] https://www.vikingcodeschool.com/web-design-basics/what-are-mockups
[2] https://study.com/articles/graphic_designer_vs_-programmer.html
[3] https://sourcemaking.com/design_patterns
[4] https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-4.0/aa970268(v=vs.100)
[5] https://www.codeproject.com/Articles/819294/WPF-MVVM-step-by-step-Basics-to-Advance-Level, 2014

Kratka biografija:

Mirko Odalović rođen je 20.01.1994. godine u Bačkoj Topoli. Završio je srednju tehničku školu Ivan Sarić u Subotici 2013. godine. Fakultet tehničkih nauka u Novom Sadu je upisao 2013. godine. Ispunio je sve obaveze i položio je sve ispite predviđene studiskim programom.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.31
DOI: https://doi.org/10.24867/01BE08Zukorlic

VERIFIKACIJA REZULTATA EMS PRORAČUNA KROZ IZMENU MODELA

VERIFICATION OF EMS FUNCTIONS' RESULTS THROUGH MODEL MODIFICATION

Nemanja Zukorlić, Savo Đukić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je izvršena verifikacija rezultata nekoliko funkcija softvera za upravljanje prenosnim mrežama kroz izmenu modela. Ukratko su opisane najvažnije funkcije softvera za upravljanje prenosnim mrežama. Nakon toga, kroz nekoliko primera izmena modela prenosne mreže, izvršena je analiza rezultata funkcija i dati su osnovni zaključci rada.*

Abstract – *The paper provides verification of EMS functions' results through network model modifications. The most important EMS functions are briefly described. Through several examples of network model modifications, function results are analyzed and main conclusions are given.*

Ključne reči: *softver za upravljanje prenosnom mrežom, model prenosne mreže.*

1. UVOD

Elektroenergetski sistem se sastoji od skupa generatora, transformatora i vodova koji deluju kao jedinstvena celina a čiji je osnovni zadatak obezbeđenje pouzdanog, kvalitetnog i racionalnog napajanja električnom energijom različitih vrsta potrošača [1]. Elektroenergetski sistem čine četiri podsistema: proizvodnja, prenos, distribucija i neposredna potrošnja. Cilj ova četiri podsistema je da se ostvari najveći mogući promet i profit, što je dovelo do potrebe korišćenja sofisticiranih alata za njihovo upravljanje.

Podsistem prenosa čini prenosna mreža, koja se sastoji pre svega od vodova i transformatora. Prenosnom mrežom se električna energija, proizvedena u elektranama, prenosi do distributivnih mreža [2]. Softver za upravljanje prenosnim mreža (EMS softver) pomaže preduzeću za prenos električne mreže da što kvalitetnije prenese električnu energiju do potrošačkih područja, uz što manje troškova. Funkcije EMS softvera se koriste u cilju smanjenja gubitaka i operativnih troškova uz dodatno poboljšanje kvaliteta prenosa energije. Osnova svih funkcija je mrežni model koji sadrži podatke o mreži, uključujući i odgovarajuće matematičke prezentacije elemenata i topologiju mreže.

Cilj ovog rada je da se izvrši verifikacija rezultata EMS funkcija kroz izmene mrežnog modela. U drugoj glavi rada opisane su osnovne karakteristike EMS softvera, osnovne funkcije EMS softvera, način njihovog korišćenja i pogodnosti koje se njima dobijaju.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Savo Đukić, doc.

Treća glava sadrži konkretne postavke primera koji će biti razmatrani u radu u svrhu verifikacije rezultata EMS funkcija. U istoj glavi prikazani su rezultati dobijeni korišćenjem EMS softvera i dati odgovarajući komentari. U četvrtoj i petoj glavi rada dati su osnovni zaključci i literatura korišćena prilikom izrade rada, respektivno.

2. EMS SOFTVER

U okviru ove glave opisana je uloga softvera za upravljanje prenosnom mrežom u funkcionisanju modernih prenosnih mreža. Objašnjen je značaj korišćenja EMS softvera, kao i osnovne karakteristike i funkcije EMS softvera.

2.1 Značaj EMS softvera

EMS softver obezbeđuje, operatorima u preduzećima za prenos električne energije, alate za dinamičku vizuelizaciju, monitoring i kontrolu prenosne mreže, zajedno sa širokim setom aplikacija za operativnu analizu, planiranje i optimizaciju. EMS softver može da poveća sigurnost sistema tako što omogućava operatorima da predvide da li će neki nepredviđeni događaj dovesti do problema u prenosnoj mreži, poput preopterećenja opreme, novih ispada, pada frekvencije, niskih napona ili do nestabilnosti sistema. Bilo koja od navedenih situacija može isključiti iz pogona značajan broj elemenata i ostaviti bez napajanja veliki broj potrošača, pri čemu troškovi u vidu izgubljenih prihoda, odštete potrošača i oštećenja opreme mogu biti ekstremno veliki.

EMS softver, preduzećima za prenos električne energije, obezbeđuje pogodnosti kao što su nadzor i kontrola prenosne mreže, pristup izveštajima sa podacima za svaki elemenat i za celu mrežu, pomoć u održavanju prenosne mreže, smanjenje gubitaka električne energije, eliminisanje preopterećenja na vodovima, efikasno iskorišćenje postojeće opreme i odlaganje novih investicija, kao i bolju obuka inženjera i dispečera [3].

Na kraju treba naglasiti da je EMS softver visoko profitabilna i relativno mala investicija u poređenju sa troškovima izgradnje, rekonstrukcije i održavanja prenosne mreže.

2.2 Karakteristike EMS softvera

EMS softver se razvija u kombinaciji sa SCADA sistemom koji se koristi za nadgledanje i kontrolu rasklopne i druge opreme u prenosnim transformatorskim stanicama. Softver koristi kolekcije podataka koji su podeljene u dve kategorije:

- statički, i
- dinamički.

Statičkim podacima opisana je topologija elektroenergetskog sistema. Tipični statički podaci koji opisuju prenosnu mrežu su podaci o vodovima, transformatorima i transformatorskim stanicama. Dinamički podaci su vrednosti koje vraćaju merni uređaji, podaci o položaju prekidača, podaci o položaju regulacione sklopke regulacionih transformatora i slično. Ova kategorija podataka predstavlja stanje elemenata elektroenergetskog sistema u datom trenutku.

EMS softver je izgrađen na konceptu *Smart Grid* rešenja i odgovara na mnoge savremene zahteve potrošača kao što su: integracija i dvosmerna komunikacija sa SCADA-om, praćenje i kontrola mreže u realnom vremenu, automatska restauracija napajanja u slučaju kvarova, smanjenje prosečnog trajanja prekida napajanja sistema, validacija promene topologije mreže, smanjenje gubitaka i troškova rada, fleksibilan pristup bazi podataka i slično.

2.3 Funkcije EMS softvera

Funkcije EMS softvera se koriste u cilju smanjenja gubitaka i operativnih troškova rada prenosne mreže, kao i poboljšanja kvaliteta i kvantiteta snabdevanja potrošača električnom energijom. Neke od najvažnijih funkcija EMS softvera su:

1. Estimacija stanja - funkcija koja na osnovu telemetrisanih podataka u realnom vremenu određuje operativno stanje mreže. Estimacija stanja je jedna od najvažnijih funkcija jer se njeni rezultati koriste kao početna tačka za druge EMS funkcije [4].
2. Tokovi snaga - funkcija koja za datu topologiju i specifiranu proizvodnju i potrošnju određuje stacionarno stanje prenosne mreže. Ova funkcija se koristi u okviru proračuna velikog broja drugih EMS funkcija, ali i za različite vrste analiza i simulacija.
3. Naponska stabilnost - funkcija koja se koristi za procenu naponske sigurnosti prenosne mreže.
4. Optimalni tokovi snaga - funkcija koja proračunava stacionarno stanje prenosne mreže uz optimizaciju tokova snaga na prenosnim vodovima.
5. Analiza ispada - funkcija koja proverava da li je stanje prenosne mreže ostalo u normalnim granicama funkcionisanja prilikom ispada jednog ili više elemenata [5].
6. Optimalna promena topologije - funkcija koja analizira koje izmene u topologiji prenosne mreže mogu dovesti do eliminisanja opasnih pogonskih stanja u mreži i/ili smanjenja gubitaka aktivne snage.
7. Proračun kratkih spojeva - funkcija koja proračunava stacionarno stanje pri kratkom spoju. Rezultati proračuna kratkih spojeva se koriste pri izboru zaštitne, prekidačke i druge opreme u prenosnoj mreži.
8. Provera kapaciteta rasklopne opreme - funkcija koja proverava prekidačka svojstva rasklopnih uređaja za zadatu topologiju i razmatrano stanje mreže sa kratkim spojem.
9. Provera podešenja relejne zaštite - funkcija koja služi za analizu efikasnosti (osetljivosti i selektivnosti) relejne zaštite.
10. Nadgledanje rezervi reaktivnih snaga - funkcija koja nadgleda indvidualne i ukupne rezerve reaktivne snage u prenosnoj mreži.

3. PRIMERI UTICAJA PROMENE MREŽNOG MODELA NA EMS FUNKCIJE

Zajednički ulazni parametar svih EMS funkcija je mrežni model. Za svaku funkciju, određene izmene u mreži mogu da dovedu do promene njihovih rezultata. U ovom radu, na prenosnoj mreži čije su karakteristike date u tabeli 1, proverene su određene funkcionalnosti kroz izmenu mrežnog modela, korišćenjem EMS softvera.

Tabela 1- *Karakteristike prenosne test mreže*

Karakteristika	Vrednost
Broj transformatorskih stanica	30
Broj generatora	17
Broj potrošača	10
Broj vodova	68
Ukupna dužina vodova [m]	870 016
Broj čvorova	119
Broj kondenzatora	11
Broj statičkih VAR kompenzatora	2
Broj transformatora	34

Za funkciju tokova snaga obrađen je primer dve paralelne sekcije, istih dužina i kataloških parametara. Pre izmena u modelu, tokovi snaga obe sekcije su približno istih vrednosti, što je prikazano u tabeli 2. Aktivne snage su date u [MW], reaktivne u [MVAr], a vrednosti struje u [A].

Tabela 2 - *Inicijalne vrednosti tokova snaga dve paralelne sekcije*

Sekcija	Relativno opterećenje	Čvor 1			Čvor 2			Pg	Qg
		P	Q	I	P	Q	I		
1	85,6	90,9	1,4	392,2	88,4	3,8	392,1	2,5	5,2
2	85,7	91,0	1,4	392,5	88,5	3,8	392,4	2,5	5,2

Vrednosti tokova snaga nakon smanjenja vrednosti dužine druge sekcije sa 37780 m na 32000 m, nalaze se u tabeli 3.

Tabela 3 - *Vrednosti tokova snaga dve paralelne sekcije nakon smanjenja dužine druge sekcije*

Sekcija	Relativno opterećenje	Čvor 1			Čvor 2			Pg	Qg
		P	Q	I	P	Q	I		
1	78,3	83,2	0,6	359,5	81,1	3,3	359,4	2,1	4,0
2	92,5	98,7	1,1	425,1	95,8	4,3	425,0	2,9	5,4

Kao što se i očekivalo, kako bi se održale ukupne injektirane aktivne i reaktivne snage u čvorovima, opterećenje, aktivna snaga i struja na krajevima druge sekcije se povećavaju, dok se vrednosti istih veličina na krajevima prve sekcije smanjuju.

Kao što se očekivalo, kada dve sekcije rade u paraleli, promena dužine jedne sekcije značajno utiče na raspodelu tokova snaga duž istih.

Za funkciju naponske stabilnosti obrađen je primer uticaja dodavanja sekcije na veličinu rezerve aktivne snage (Prez). Sekcija se dodaje paralelno sekciji koja predstavlja usko grlo mreže sa aspekta naponske stabilnosti (rezerve aktivne snage). Kritičan limit ispod kojeg rezerva aktivne snage u razmatranoj test mreži ne sme da padne je 100 MW. Pre dodavanja paralelne sekcije, rezerva aktivne snage je ispod limita u prevojnoj tački, vidi tabelu 4. Aktivne snage su date u [MW], a vrednosti napona u [%].

Tabela 4 – Inicijalni rezultati funkcije naponske stabilnosti u prevojnoj i kritičnoj tački

Prevojna tačka sistema			Kritična tačka sistema		
Napon	Ptok	Prez	Napon	Ptok	Prez
90,5	247,8	58,5	85,9	311,1	121,8

Nakon dodavanja paralelne sekcije sa istim parametrima, opterećenje se raspoređuje na obe sekcije. Rezultati funkcije naponske stabilnosti, nakon dodavanja sekcije dati su u tabeli 5.

Tabela 5 – Rezultati funkcije naponske stabilnosti u prevojnoj i kritičnoj tački nakon dodavanja paralelne sekcije

Prevojna tačka sistema			Kritična tačka sistema		
Napon	Ptok	Prez	Napon	Ptok	Prez
88,8	486,1	302,8	87,1	525,4	342,2

Očekivano, tok (Ptok) i rezerva aktivne snage su se povećali i za prevojnu i za kritičnu tačku. Rezerva aktivne snage više nije ispod limita.

Za funkciju optimalnih tokova snaga obrađen je primer izmene krive troškova proizvodnje aktivne snage generatora.

U primeru se razmatra kako smanjenje koeficijenata krive troškova proizvodnje generatora utiče na proračunatu optimalnu vrednost proizvodnje aktivne snage tog generatora. Optimalne vrednosti proizvodnje aktivne snage i odgovarajućih troškova generatora, pre smanjenja koeficijenata, prikazane su u tabeli 6.

Tabela 6 - Optimalne vrednosti proizvodnje i troškova generatora pre izmene koeficijenata krive troškova

P [MW]	Q [MVAr]	Troškovi [$]
1,7	-1,3	113,12

Nakon smanjenja koeficijenata krive troškova generatora, promenile su se i optimalne vrednosti proizvodnje aktivne snage i troškova generatora, vidi sliku 7.

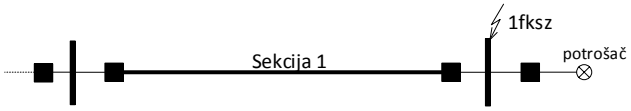
Tabela 7 - Optimalne vrednosti proizvodnje i troškova generatora nakon smanjenja koeficijenata krive troškova

P [MW]	Q [MVAr]	Troškovi [$]
4,5	-1,4	107,47

Kao što je i očekivano, nakon smanjenja koeficijenata odgovarajuće krive troškova, povećava se optimalna vrednost proizvodnje aktivne snage generatora, naravno pod uslovom da prethodno nije dostignuta maksimalna vrednost proizvodnje.

Za funkciju proračuna kratkih spojeva obrađen je primer uticaja promene parametara sekcije na struju kratkog spoja.

Simuliran je jednofazni kratak spoj sa zemljom u fazi L1 na radijalno napajanom čvoru prenosne stanice 132, prikazanom na slici 1.



Slika 1 - Deo mreže na kojem se vrši analiza proračuna kratkih spojeva

Vrednosti struje i napona kratkog spoja u razmatranom čvoru, pre izmene parametara sekcije koja napaja predmetnu prenosnu stanicu, prikazane su u tabeli 8.

Tabela 8 - Inicijalne vrednosti struje i napona jednofaznog kratkog spoja sa zemljom

I [A]			V [kV]		
L1	L2	L3	L1	L2	L3
16 944,8	0,0	0,0	0,0	92,0	91,1

Vrednosti istih veličina, nakon povećanja podužne otpornosti gore pomenute sekcije sa 0,118 Ω/km na 0,3 Ω/km, nalaze se u tabeli 9.

Tabela 9 - Vrednosti struje i napona jednofaznog kratkog spoja sa zemljom nakon povećanja podužne otpornosti sekcije

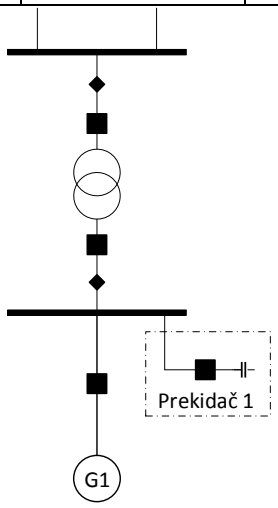
I [A]			V [kV]		
L1	L2	L3	L1	L2	L3
16 814,2	0,0	0,0	0,0	92,8	89,7

Kao što je i očekivano, povećanjem podužne otpornosti sekcije, raste vrednost impedanse kratkog spoja, pa samim tim struja kratkog spoja opada.

Za funkciju nadgledanja rezerve reaktivnih snaga obrađen je primer dodavanja kondenzatora reaktivne snage 1 MVAr u prenosnoj transformatorskoj stanici (slika 2). U tabeli 10 prikazane su vrednosti rezervi po tipu i njihov procentualni udeo u ukupnoj rezervi reaktivne snage u toj transformatorskoj stanici pre gore pomenute izmene mrežnog modela.

Tabela 10 - Vrednosti rezervi reaktivnih snaga pre dodavanja kondenzatora

Tip rezerve	Rezerva [MVAr]	Rezerva [%]
Rotirajuća	203,90	100,00
Brzo startna	0,00	0,00
Operativna	203,90	100,00



Slika 2 - Deo mreže sa novododatim kondenzatorom

Kada je Prekidač 1 zatvoren, kondenzator je povezan na mrežu, čime se oslobađa 1 MVAr reaktivne snage na generatoru (1 MVAr rotirajuće rezerve). Vrednosti rezervi po tipu i njihov procentualni udeo u ukupnoj rezervi reaktivne snage, za ovaj slučaj, prikazani su u tabeli 11.

Tabela 11 – Vrednosti rezerve reaktivne snage nakon dodavanja kondenzatora (Prekidač 1 zatvoren)

Tip rezerve	Rezerva [MVAr]	Rezerva [%]
Rotirajuća	205,00	100,00
Brzo startna	0,00	0,00
Operativna	205,00	100,00

U slučaju kada je Prekidač 1 otvoren, kondenzator može brzo da se priključi na mrežu, pa predstavlja brzo startnu rezervu. Vrednosti rezervi po tipu i njihov procentualni udeo u ukupnoj rezervi reaktivne snage, za ovaj slučaj, prikazani su u tabeli 12.

Tabela 12 – Vrednosti rezerve reaktivne snage nakon dodavanja kondenzatora (Prekidač 1 otvoren)

Tip rezerve	Rezerva [MVAr]	Rezerva [%]
Rotirajuća	203,90	99,51
Brzo startna	1,00	0,49
Operativna	204,90	100,00

Kao što je očekivano, dodavanjem kondenzatora reaktivne snage 1 MVAr u mrežu, rezerva (rotirajuća u slučaju kada je kondenzator povezan na mrežu, odnosno brzo startna u slučaju kada kondenzator nije povezan na mrežu) se povećala za tu vrednost.

4. ZAKLJUČAK

Prenosna mreža, kao jedan od najvažnijih delova elektroenergetskog sistema, mora da obezbedi kontinualan i pouzdan prenos električne energije od proizvodnih jedinica do potrošača. Usled težnje za što kvalitetnijim prenosom električne energije, razvila se potreba za korišćenjem EMS softvera. EMS funkcije imaju za cilj smanjenje gubitaka i operativnih troškova u prenosnoj mreži, uz poboljšanje kvaliteta i kvantiteta snabdevanja potrošača.

Radom je dat predlog verifikacije rezultata EMS funkcija kroz promenu modela prenosne mreže. Kroz nekoliko primera obrađeni su različiti aspekti uticaja izmena mrežnog modela na rezultate funkcija. Na osnovu razmatranih primera može se zaključiti da izmene modela mreže mogu biti efikasan način za proveru kvaliteta rezultata EMS funkcija.

5. LITERATURA

[1] Duško Bekut, *Relejna zaštita,* Fakultet tehničkih nauka, Novi Sad, 2009.

[2] Vladimir Strezoski, *Analiza elektroenergetskih sistema*, Fakultet tehničkih nauka, Novi Sad, 2011.

[3] https://web.stanford.edu/class/archive/ee/ee392n/ee392n.1116/Lectures/EE392n_Lecture5GE.pdf

[4] A. Monticelli, *State estimation in electric power systems: A generalized approach*, University of Campinas, Unicamp, 1999.

[5] V. Mishra, M. Khardevnis, *Contingency analisys of power systems*, Electrical, Electronics and Computer Science Conference, Bhopal, 2012.

Kratka biografija:

Nemanja Zukorlić rođen je 1991. godine u Zrenjaninu. Fakultet tehničkih nauka u Novom Sadu upisao je školske 2010/2011, smer energetika, elektronika i telekomunikacije. Na osnovnim studijama diplomirao je 2015. godine, smer elektroenergetski sistemi. Diplomski – master rad iz oblasti Elektrotehnike i računarstva odbranio je 2018. godine.

Savo Đukić rođen je u Novom Sadu 1983. godine. Diplomirao je i doktorirao na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva, smer Elektroenergetski sistemi 2007. i 2014. godine, respektivno.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK 004.3
DOI: https://doi.org/10.24867/01BE09Popovic

VERIFIKACIJA REKONFIGURABILNE ARHITEKTURE ZA HARDVERSKU AKCELERACIJU PREDIKTIVNIH MODELA MAŠINSKOG UČENJA

VERIFICATION OF A RECONFIGURABLE ARCHITECTURE FOR HARDWARE ACCELERATION OF MACHINE LEARNING PREDICTIVE MODELS

Jasna Popović, Rastislav Struharik, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu je predstavljena funkcionalna verifikacija rekonfigurabilne arhitekture za hardversku akceleraciju prediktivnih modela mašinskog učenja - Reconfigurable Machine Learning Classifier (RMLC).*

Ključne reči: *funkcionalna verifikacija, mašinsko učenje, hardverska akceleracija, univerzalna verifikaciona metodologija (UVM), SystemVerilog*

Abstract –*In this paper we present functional verification of a reconfigurable architecture for hardware acceleration of machine learning predictive models - Reconfigurable Machine Learning Classifier (RMLC).*

Keywords: *functional verification, machine learninig, hardware acceleration, universal verification methodology (UVM), SystemVerilog*

1. UVOD

Ovaj rad predstavlja digitalnu verifikaciju rekonfigurabilne arhitekture za hardversku akceleraciju prediktivnih modela mašinskog učenja - *Reconfigurable Machine Learning Classifier* (RMLC). Verifikacija je realizovana u Univerzalnoj Verifikacionoj Metodologiji (*eng. Universal Verification Methodology - UVM*) u kombinaciji sa *SystemVerilog* programskim jezikom.

RMLC se sastoji od većeg broja istovetnih blokova i može se konfigurisati da implementira ortogonalna stabla odluke (eng. *Decision Trees - DT*), neortogonalna stabla odluke, nelinearna stabla odluke, funkcionalna stabla odluke, regresiona stabla odluke, linearne i nelinearne *Support Vector Machine* (SVM) prediktivne modele sa polinomijalnim i radijalnim kernelima, veštačke neuronske mreže zasnovane na radijalnim funkcijama (eng. *Artificial Neural Network* - ANN) i višeslojne perceptrone (eng. *Multilayer Perceptron - MLP*) [1]. Verifikaciono okruženje je projektovano na način koji omogućava jednostavnu konfiguraciju i testiranje navedene arhitekture.

1.1. Motivacija

Skorašnji napredak u informacionoj i komunikacionoj tehnologiji doveo je do eksponencijalnog rasta količine podataka skladištenih u bazama podataka.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Rastislav Struharik, red. prof.

Od početka informacionog doba, skupljanje podataka je postalo jednostavnije, a čuvanje informacija jeftinije. Kako se povećava količina mašinski čitljivih podataka, sposobnost razumevanja i primene ovih informacija ne prati korak sa tim rastom. *Data mining* se javlja kao metoda suočavanja sa ovim eksponencijalnim rastom podataka [2].

Mašinsko učenje predstavlja ključnu stavku prilikom izvlačenja značajnih podataka iz prostora dostupnih podataka koji se skupljaju svaki dan. Jedan od izvora ovih podataka su podaci koji su očitani sa senzora. Više podataka je nastalo u poslednje dve godine nego u čitavoj istoriji ljudske rase [3]. Da bi se ovi podaci analizirali, neophodan je hardver koji omogućava brz protok velike količine informacija [3].

Sposobnost automatskog učenja iz ogromne količine podataka je dovela do neverovatnih dostignuća u naučnim poljima kao što su računarski vid (eng. *comuputer vision*), prepoznavanje govora (eng. *speech recognition*) i procesiranje prirodnog jezika (eng. *natural language processing*). Ovi modeli zahtevaju procesiranje velike količine podataka, i veliku računarsku moć za treniranje. Njihov napredak je ograničen postojećim računarima [4]. Trenutno stanje hardverskog ubrzanja se svodi na klastere grafičkih procesora koji služe kao procesori opšte namene. Snažan protivnik grafičkih procesora su FPGA platforme (eng. *Field Programmable Gate Array*), koji za razliku od grafičkih procesora, imaju fleksibilnu hardversku konfiguraciju, i često pružaju bolje performanse u pogledu energetske efikasnosti.

U okviru procesa dizajna hardvera, postoji više verifikacionih procesa. Ti procesi uključuju funkcionalnu verifikaciju, vremensku (eng. *timing*) verifikaciju, test verifikaciju i proveru ekvivalentnosti. Vremenski najzahtevnija je funkcionalna verifikacija. Funkcionalna verifikacija garantuje da logika u čipu i sistemu radi korektno u svim okolnostima, onako kako je definisano u specifikaciji sistema [5].

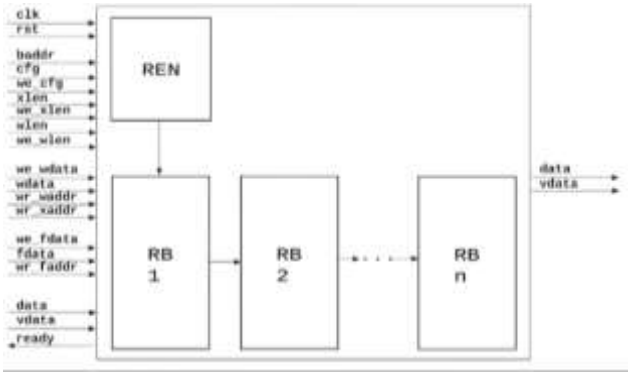
2. OPIS REŠAVANOG PROBLEMA

U ovom poglavlju je dat kratak opis RMLC arhitekture koja se verifikuje. Sadržaj ovog poglavlja se zasniva na radu V. Vranjković [1].

2.1. Detalji arhitekture

Na Slici 1 je prikazana RMLC arhitektura. Arhitekturu čini nekoliko istovetnih blokova koji su međusobno povezani u niz. Ovi blokovi se zovu Reconfigurable

Block (prev. rekonfigurabilan blok - RB) moduli. Broj ovih modula u arhitekturi je tokom dizajna postavljen na 5, i predstavlja parametar arhitekture koji se može podesiti tokom dizajna sistema. Pored RB modula, u arhitekturi se nalazi i Reconfigurable Enable – RE modul, čija je odgovornost da bira koji RB modul se konfiguriše tokom programiranja sistema.

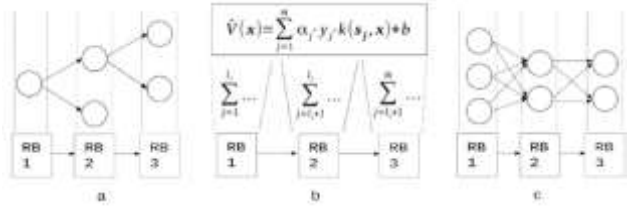


Slika 1. *Blok šema RMLC arhitekture*

RMLC arhitektura implementira tri različita klasifikatora: stabla odluke, SVM, i veštačke neuronske mreže. Kada je RMLC konfigurisan da radi kao stablo odluke, tada će na izlazu arhitekture biti sračunata klasa ulazne instance. Prilikom mapiranja na RB module, svi čvorovi i listovi u jednom nivou stabla su preslikani u jedan RB modul. Kada je RMLC konfigurisan da radi kao SVM, na izlazu klasifikatora će se naći izračunata suma funkcije za predikciju:

$$v(s) = \sum_{i=1}^{m} y_i \alpha_i K(x_i, s) + b \,, \qquad (1)$$

gde je s ulazni podatak. Znak $v(s)$ je klasa ulaznog primera. Prilikom mapiranja, ova suma se deli na delimične sume. Svaka od dobijenih delimičnih suma se potom računa u pojedinačnom RB modulu. I na kraju, kada je RMLC konfigurisan da radi kao veštačka neuronska mreža, na *data* portu će biti izlazne vrednosti za svaki neuron izlaznog sloja, Prilikom mapiranja, svi neuroni jednog skrivenog sloja se preslikavaju u jedan RB modul. Metode mapiranja su prikazane na Slici 2.



Slika 2*: a) mapiranje stabla odluke na RMLC sa tri RB modula, b) mapiranje SVM na RMLC sa tri RB modula, c) mapiranje veštačke neuronske mreže na RMLC sa tri RB modula*

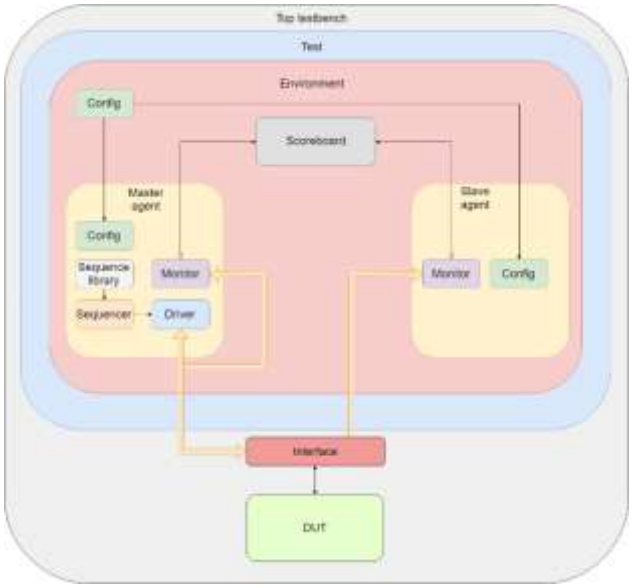
3. IMPLEMENTACIJA

Ovo poglavlje opisuje verifikaciono okruženje koje je implementirano. Opisan je način na koji je realizovan UVM testbench i generator stabala odluke.

3.1. UVM Testbench

Verifikacija je realizovana korišćenjem UVM metodologije i *SystemVerilog* programskog jezika.

Testbench predstavlja *SystemVerilog* modul u kome se instancira DUT, podešava se virtuelni interfejs putem konfiguracione baze podataka, i povezuje se sa instanciranim DUT-om. U ovom modulu se takođe instancira test klasa, i pokreće test pozivom taska run_test(), u okviru initial begin … end bloka, gde su definisani i reset i sinhronizacioni signal (Blok šema testbench-a se nalazi na Slici 3).



Slika 3. *Blok šema realizovanog testbench-a*

Test je *top level* UVM komponenta u okviru UVM t*estbench-a.* Test ima tri glavne funkcije: da instancira verifikaciono okruženje najvišeg nivoa (*eng. top level verification environment*), da konfiguriše verifikaciono okruženje (putem *factory override*-a ili konfiguracione baze podataka), i da primeni stimulus pozivom UVM sekvence kroz okruženje do DUT-a [6]. U ovom testbench-u postoji osnovni test test_base.sv koji instancira verifikaciono okruženje i konfiguraciju najvišeg nivoa. Ovaj osnovni test nasleđuju svi drugi testovi, koji potom drugačije konfigurišu okruženje, ili pokreću neke druge sekvence.

Verifikaciono okruženje se sastoji iz dva agenta, od kojih je jedan aktivan a drugi pasivan agent, konfiguracione komponente i skorborda.

Master agent se sastoji iz tri UVM komponente: drajver, sekvencer i monitor, pored kojih postoje i biblioteka sekvenci, koje se pokreću na sekvenceru, i konfiguracija. Drajver komponenta je aktivna komponenta u agentu, koja je zadužena za pokretanje aktivnosti na interfejsu ka DUT-u. Od sekvencera dobija sekvence transakcija sa podacima koje šalje ka interfejsu. Komunikacija između drajvera i DUT-a se odvija preko virtuelnog interfejsa, što je samo pokazivač na interfejs DUT-a. Sekvence koje sekvencer komponenta prosleđuje drajveru se nalaze u biblioteci sekvenci. Sekvenca koja će se proslediti drajveru se bira na nivou testa. Uloga monitor komponente je da nadgleda interfejs. Ukoliko se pojavi neka aktivnost na interfejsu, monitor komponenta će pokupiti podatke koristeći virtuelni interfejs, i sakupiti ih u jednu transakciju. Tu transakciju potom prosleđuje skorbordu.

Slejv agent (eng. slave) verifikacionog okruženja je definisan kao pasivan agent. Čine ga samo monitor komponenta i konfiguracija. Monitor komponenta nadgleda izlazne signale DUT-a, i ukoliko se validni

podaci pojave na interfejsu, skuplja ih u transakciju i šalje skorbordu.

Skorbord komponenta je osnovni element verifikacionog okruženja posvećen proveri rada DUT-a na funkcionalnom nivou [6]. Skorbord prima transakcije od strane monitor komponente agenata u verifikacionom okruženju preko UVM analysis_port portova. Ovi portovi su specijalizovani TLM portovi čiji se interfejs sastoji iz jedne write() funkcije.

U realizovanom verifikacionom okruženju, skorbord implementira funkcionalnost DUT-a na visokom nivou apstrakcije. Konkretno, implementirane su funkcije za svaku pojedinačnu podržanu konfiguraciju.

Transakcije koje monitor master agenta šalje skorbordu se koriste kao ulaz za implementirane funkcije. Rezultat funkcije se zatim poredi sa transakcijom koju je monitor slejv agenta poslao skorbordu. Ukoliko se rezultati ne poklapaju, šalje se obaveštenje o greški putem uvm_error metode.

3.2. Generator konfiguracije RMLC arhitekture

Kako bi se RMLC arhitektura verifikovala, potrebno je testirati što veći broj slučajeva u kojima bi mogla da se nađe. Ukoliko radi kao DT klasifikator, potrebno je dovesti na ulaze RMLC arhitekture sve moguće konfiguracije stabla odluke. Ukoliko radi kao SVM klasifikator, potrebno je dovesti dovoljno veliki i mali broj SVM vektora, sa manjim ili većim brojem ulaznih instanci. Ukoliko radi kao ANN mreža, potrebno je na ulaze arhitekture dovesti mreže sa manje ili više neurona u skrivenom sloju, sa različitim brojem ulaznih instanci.

Za potrebe verifikacije RMLC arhitekture kada radi kao stablo odluke, razvijen je model stabla odluke na visokom nivou apstrakcije u *SystemVerilog* programskom jeziku. Generisano stablo odluke se potom analizira, i generiše se odgovarajuća konfiguracija za RMLC. Stablo se dalje prosleđuje do scoreboard-a, gde se proverava rad RMLC arhitekture u ovom modu rada.

Za verifikovanje RMLC arhitekture kada radi kao veštačka neuronska mreža nije bilo potrebno razviti poseban model za generisanje konfiguracije RMLC arhitekture. Bilo je neophodno razviti model u okviru scoreboarda, koji će predviđati izlazni rezultat RMLC arhitekture. U tu svrhu, razvijen je jednostavan model koji računa rezultat rada neuroske mreže na nivou sloja neurona. Rezultat ulaznog sloja se prosleđuje do funkcije koja računa rezultat rada neurona iz skrivenog sloja. Rezultat skrivenog sloja se prosleđuje izlaznom sloju, i taj rezultat se poredi sa izlazom DUT-a.

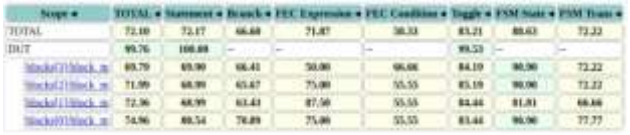
Kada arhitektura radi kao SVM klasifikator nije bilo potrebe za posebnim modelom za generisanje konfiguracije, niti za proračun rezultata. Rad SVM klasifikatora je jednostavno implementiran kao metoda u okviru scoreboard komponente.

4. REZULTATI VERIFIKACIJE

Za potrebe verifikacije razvijen je detaljan verifikacioni plan i test plan. Implementirano je ukupno 10 testova. Testovi se mogu podeliti u tri grupe: testovi gde je DUT konfigurisan kao stablo odluke, testovi gde je DUT konfigurisan kao SVM klasifikator, i testovi gde je DUT konfigurisan kao neuronska mreža sa radijalnim kernelom. Svaka od ove tri grupe ima po jedan ili više direktnih testova koji gađaju specifične slučajeve, i po jedan potpuno nasumičan test.

Kao merilo uspešnosti verifikacije koristili smo metode funkcionalne pokrivenosti i pokrivenosti koda.

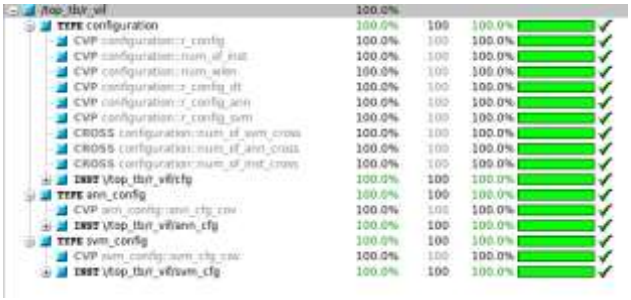
Pokrivenost se definiše kao procenat verifikacionih ciljeva koji su ispunjeni, i koristi se kao mera uspešnosti verifikacije [7]. U širem smislu postoje dva tipa pokrivenosti: pokrivenost koda i funkcionalna pokrivenost. Pokrivenost koda je mera koja označava do kog stepena je kod digitalnog dizajna testiran u datom verifikacionom okruženju. Izveštaj o pokrivenosti koda se automatski generiše od strane alata koji se koristi [7]. Rezultati pokrivenosti koda po instancama dizajna su prikazani na Slici 4.



Slika 4*: Rezultat pokrivenosti koda RMLC arhitekture po instancama dizajna*

Kao što se može zaključiti iz Slike 4, pokrivenost koda nije 100%. Ovakav rezultat je očekivan, s obzirom da su testirane tri konfiguracije, a postoje još tri konfiguracije koje nisu testirane. Samim tim, pokrivenost koda je manja, s obzirom da postoje stanja u koja RMLC neće ulaziti.

Funkcionalna pokrivenost predstavlja metriku koju definiše korisnik, i koja pruža informacije o tome u kojoj meri je funkcionalnost dizajna predviđena specifikacijom upotrebljena prilikom testiranja [7]. Može pružiti informacije o tome da li su se neki interesantni scenariji, krajnji slučajevi ili neki drugi uslovi dizajna desili, i da li su bili provereni od strane verifikacionog okruženja. Na Slici 5 su prikazani rezultati funkcionalne pokrivenosti, koja je 100%.



Slika 5*: Rezultat funkcionalne pokrivenosti*

5. ZAKLJUČAK

U ovom radu je predstavljen proces funkcionalne verifikacije rekonfigurabilne arhitekture za hardversku akceleraciju prediktivnih modela mašinskog učenja. Postupak realizacije verifikacije je započet pravljenjem verifikacionog plana i test plana, zatim je izgrađeno verifikaciono okruženje korišćenjem univerzalne verifikacione metodologije i *SystemVerilog* programskog jezika, i implementirani su testovi koji proveravaju funkcionalnost dizajna. Uspešnost verifikacije je izmerena pokrivenošću koda i funkcionalnom pokrivenošću.

6. LITERATURA

[1] V. Vranjković, "Rekonfigurabilne arhitekture za hardversku akceleraciju prediktivnih modela mašinskog učenja", *Autorski reprint*, 139., 2015.

[2] L. Rokach, O. Maimon, "Series in Machine Perception and Artificial Intelligence – Vol. 69, Data Mining with Decision Trees Theory and Applications", *Worlds Scientific Publishing Co. Pte. Ltd*, 2008.

[3] V. Sze, Y. Chen, J. Emer, A. Suleiman, Z. Zhang, "Hardware for Machine Learning: Challenges and Opportunities*", Massachusetts Institute of Technology Cambridge, MA 02139*, Oktobar 2017.

[4] G. Lacey, G. Taylor, S. Areibi, "Deep Learning on FPGAs: Past, Present, and Future", *University of Guelph 50 Stone Rd E Guelph, Ontario*, Februar 2016.

[5] B. Wile, John C. Goss, W. Roesner, "Comprehensive Functional Verification: The Complete Industry Cycle", *Library of Congress Cataloging-in-Publications Data, ISBN: 0-12-78183-7*, 2005.

[6] "Universal Verification Methodology (UVM) 1.2 User's Guide", *Accellera Systems Initiative (Accellera). Accellera Systems Initiative, 8698 Elk Grove Bldv Suite 1, #114, Elk Grove, CA 95624, USA*, 2011 – 2015.

[7] Mozhikunnath, R., Garg, R. "Cracking Digital VLSI Verification Interview Interview Success", 2016.

Kratka biografija:

Jasna Popović je rođena u Sremskoj Mitrovici 1993. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Embeded sistemi i algoritmi odbranila je 2016.god.

Rastislav Struharik je vanredni profesor na Fakultetu tehničkih nauka. Doktorsku disertaciju pod naslovom "Digitalna elektronska kola za realizaciju stabala odluka" odbranio je dana 18. decembra 2009. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.3
DOI: https://doi.org/10.24867/01BE10Lasu

PROJEKTOVANJE VERIFIKACIONOG OKRUŽENJA ZA APB2SPI NA UVM METODOLOGIJI

DESIGN VERIFICATION ENVIRONMENT FOR APB2SPI BASED ON UVM METHODOLOGY

Čongor Lašu, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Zadatak master rada jeste projektovanje verifikacionog okruženja za modul koji je APB2SPI, zasnovano na UVM metodologiji, programski jezik je SystemVerilog. Zadatak je pisanje dokumenata koji su verifikacioni plan i verifikaciona arhitektura, razvijanje UVC-eva, i razvijanje testova koji proveravaju osobine dizajna.*

Ključne reči: *Funkcionalna verifikacija, APB2SPI, UVM, pokrivenost*

Abstract – *The task of master thesis is developing verification environment for the module which is APB2SPI, based on UVM methodology and SystemVerilog programming language is used. To make documents which are verification plan and verification architecture, developing UVCs and developing tests which will check the features of design.*

Keywords: *Functional verification, APB2SPI, UVM, coverage*

1. UVOD

Verifikacija kao zadatak ima da potvrdi da logika dizajna radi usklađeno sa dizajniranom specifikacijom. Prevedeno, verifikacija pokušava da da odgovor na pitanje: "Da li ovaj predloženi dizajn radi ono što ste mislili?" Ovaj zadatak je složen, i oduzima dosta vremena i napora u većini projekata tokom projektovanja elektronskih sistema.

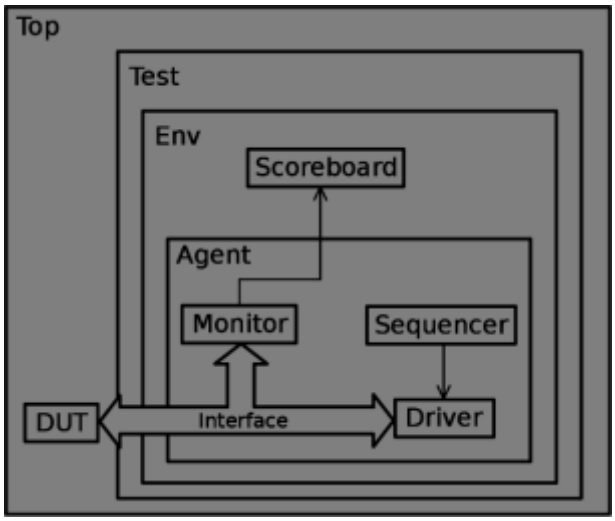
Funkcionalna verifikacija je definisana kao proces provere da li RTL dizajn (Verilog, VHDL, SystemVerilog) ispunjava svoje specifikacije iz funkcionalne perspektive. RTL verifikacija se obično deli na dve diskretne oblasti, funkcionalna verifikacija i fizička verifikacija. Funkcionalna verifikacija utvrđuje da *DUT* (Design Under Test) pravilno implementira funkcionalnost specifikacije. Fizička verifikacija proverava da sinteza, implementacija i protok protokola održavaju istu funkcionalnost u svakom nivou apstrakcije. Funkcionalna verifikacija obično predstavlja jednu od najizazovnijih oblasti u dizajnu čipa.

Kako se veličina dizajna povećava, tako se povećava i složenost. Zbog ogromnog broja potencijalnih stanja dizajna, u velikim dizajnima, funkcionalna verifikacija obično nije u mogućnosti da detaljno proveri dizajn. Iz pomenutog razloga uvedena je nova metrika, koja se zove pokrivenost (eng. Coverage).

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Rastislav Struharik, vanredni profesor.

2. UVM (Universal Verification Methodology)

Universal Verification Methodology (UVM) je standardizovana metodologija za verifikaciju dizajna integrisanih kola. UVM je izvedena uglavnom iz OVM-a (Open Verification Methodology). UVM je jedna SystemVerilog biblioteka, koja omogućava lako kreiranje fleksibilnih, i ponovo upotrebljivih verifikacionih komponenti. Verifikaciono okruženje se sastoji od velikog broja klasa, gde se svaka klasa koristi sa specifičnom svrhom, koje su grupisane tako da se povećava modularnost i portabilnost celog okruženja. Ove SystemVerilog klase su implementirane unutar UVM biblioteke i predstavljaju verifikacione komponente kao što su agenti, monitori, sekvence itd. Više informacija se može naći u knjigama [1,2]. Jednostavan primer verifikacionog okruženja, koje poštuju UVM pravila, je moguće videti na *Slika 1*. Kao što se može videti, sve komponente se nalaze na nižem nivou, dok se test klasa, koja obuhvata sve komponente, nalazi na najvišem nivou.



Slika 1 : *UVM verifikaciono okruženje*

U nastavku će biti opisane glavne komponente verifikacionog okruženja:

- Sequencer, sequences - Sequencer predstavlja jedan napredni stimulus generator, koji obezbeđuje sekvence (sequence item) driver-u na izvršenje.

Sequencer vrši kontrolu generisanja slučajnog stimulusa izvršavajući sekvence. Jednostavna sekvenca će generisati jedan ili više slučajnih data item-a. Kompleksne sekvence mogu sadržati informacije o tajmingu, parametrima i dodatnim ograničenjima.

- Driver - Driver je jedna aktivna komponenta koja oponaša logiku koja pokreće DUT. Tipični driver uzima podatke generisanih od strane sequencer-a, i prosleđuje ih DUT-u i scoreboard-u na dalju obradu.

- Monitor - Monitor je jedna pasivna komponenta koja sakuplja vrednosti sa izlaza DUT-a. Ova komponenta je zadužena za sakupljanje informacija o pokrivenosti, i proveru protokola. Nakon obrade primljenih podataka, monitor vrši pakovanje ovih podataka, i preko analysis port-ova prosleđuje scoreboard komponenti.
- Agent - Agent vrši enkapsulaciju sequencer-a, driver-a i monitor-a u smislenu celine. Verifikaciono okruženje može sadržati više agent-a. Neki agent-i su aktivni, i vrše instanciranje transakcija u DUT, dok pasivni agent-i samo posmatraju ponašanje DUT-a. Agent-i treba da budu konfigurabilni, kako bi mogli biti ili pasivni ili aktivni.
- Interface – Služi kao stvarna veza između DUT-a i verifikacionog okruženja. Skup mreže ili žice.
- Scoreboard - Scoreboard predstavlja mehanizam, koji se koristi za dinamičko predviđanje odziva DUT-a, i radi upoređivanje ovih odziva naspram posmatranih odziva na izlazu DUT-a. Ona u suštini sadrži funkcionalnost DUT-a, tj. Referentni model, koje je bitan za trenutni test scenario.
- Environment - UVM komponenta koja objedinjuje jedan ili više agent-a (ili UVC-a), zajedno sa konfiguracionom objektom. Svrha environment-a je da instancira prethodno pomenute komponente, konfiguriše ih i da ostvari konekcije između njih. Konfiguraciona svojstva zapravo vrše prilagođavanje topologije i ponašanja, i na ovaj način omogućavaju da se okruženje ponovo koristi.
- Test - Test predstavlja posebnu klasu koja služi za formiranje specifičnog verifikacionog scenarija. U ovoj klasi se instanciraju verifikacione komponente i vrši se njihova konfiguracija. Konfiguracija se podešava na način koji je osmišljen za test koji se planira izvršiti. U slučaju korišćenja TLM port-ova, potrebno je uraditi povezivanje ovih port-ova unutar test klase.

3. DUT (Design Under Test)

APB2SPI modul je veza između serijske i paralelne komunikacije, APB (Advanced Peripheral Bus) je paralelni deo, a SPI (Serial Peripheral Interface) je serijski. Modul se sastoji od konfiguracionih i statusnih registara, koji su mapirani u APB adresnom prostoru. Preko njih je moguće konfigurisati modul. Modul može biti konfigurisan da radi u master ili u slave modu, veličina podataka koja se šalje ili prima, smer komunikacije, redosled podataka takođje su konfigurabilni, i pored toga postoji i slave select manipulacija.

4. VERIFIKACIONI PLAN

Prvi korak u verifikaciji je izrada verifikacionog plana, koja služi kao putokaz u toku procesa verifikacije U verifikacionom planu su napisani SVA assertion-e za APB i SPI interfejs, potrebni checker-i za proveru dizajna i pokrivenost. U *Tabela 1* moguće je videti assertion-e za APB interfejs.

Tabela 1 : Assertion-e za APB interfejs

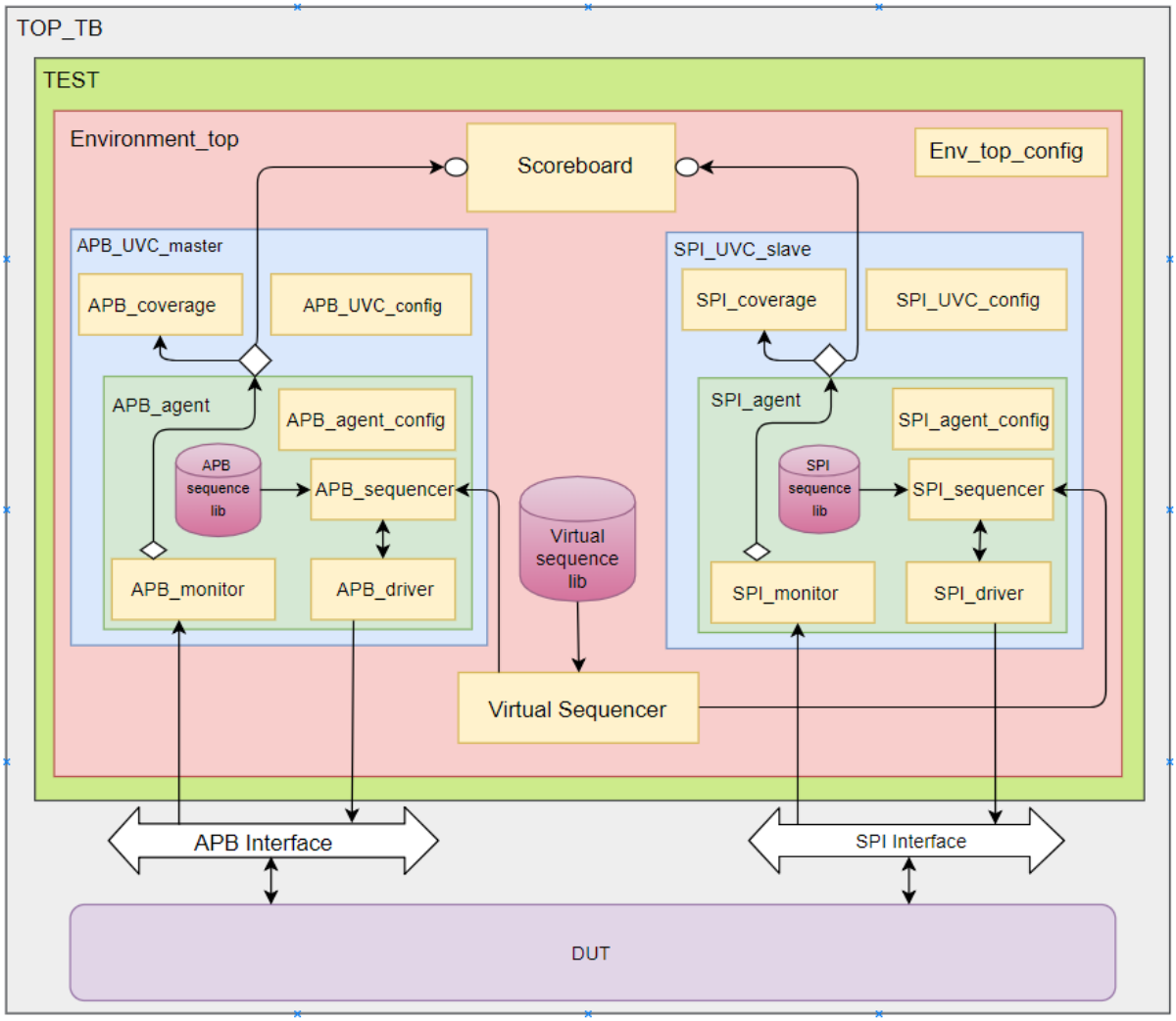
ID	Ime	Opis
1	penable_assert_chk	Enable signal mora biti postavljen posle jednog clock takta psel signala
2	penable_deassert_chk	Enable signal mora biti deaktiviran istovremeno sa signalom pready
3	penable_stability_chk	Ako se signal penable aktivira, onda on mora biti aktivan dok signal pready neće biti deaktiviran
4	pstrb_in_phase_read_chk	Tokom tranzakcije čitanja, signal pstrb mora biti deaktiviran
5	pwdata_stabilty_chk	Ako se signal psel aktivira i signal pwrite ima vrednost WRITE, onda signal pwdata mora biti stabilan dok signal pready neće se deaktivirati
6	paddr_stabilty_chk	Ako se signal psel aktivira, onda signal paddr mora biti stabilan dok se ne deaktiviše signal pready
7	pwrite_unknown_chk	Signal pwrite ne sme imati nepoznate vrednosti
8	pready_unknown_chk	Singal pready ne sme imati nepoznate vrednosti
9	psel_stability_chk	Ako se signal psel aktivira, onda on mora biti aktivan dok signal pready neće biti deaktiviran

U *Tabela 2* možemo videti pokrivenost covergroup-a APB2SPI_REGS.

Tabela 2 : Pokrivenost APB2SPI_REGS-a

ID	Ime	Opis
1	PWRITE_cov	Pokriti vrste tranzakcije
2	PADDR_cov	Pokriti adrese
3	WRITE_x_ADDR_cov	Pokriti na kojim adresama smo probali pisati
4	READ_x_ADDR_cov	Pokriti iz kojeg adresa smo probali čitati

U *Tabela 3* možemo videti checker-i za Status 0 registar.



Slika 2 : Blok dijagram TESTBENCH-a

Tabela 3: Lista checker-a

Br	Ime checker-a	Opis
1	SPIxSTAT0_rst_val_chk	Nakon reset-a Status 0 registar mora imati vrednost 0x00000030
2	SPI_STAT0_rd_only_chk	Status 0 registar je read only. Bilo koji upis ne sme promeniti njihov vrednost
3	transmit_buffer_TFE_flag_chk	TFE flag postavljen je ako predajni bafer prazan
4	transmit_buffer_TNF_flag_chk	TNF flag postavljen je ako predajni bafer nije pun
5	transmit_buffer_TXBEC_cnt_chk	Brojač elemenata predajnog bafera, mora biti manja ili jednaka sa FDEPTH+1
6	receive_buffer_RFF_flag_chk	RFF flag postavljen je ako prijemni bafer pun
7	receive_buffer_RNE_flag_chk	RNE flag postavljen je ako prijemni bafer nije prazan
8	receive_buffer_RXBEC_cnt_chk	Brojač elemenata prijemnog bafera, mora biti manja ili jednaka sa FDEPTH+1

5. VERIFIKACIONA ARHITEKTURA

Testbench se sastoji od DUT-a, od dva interfejsa i od test-a. Blok dijagram testbench-a je prikazan na *Slika 2*.
U okviru testbench-a korišćena je dva interfejsa (APB i SPI).
Test component se sastoji od Environment_top-a, koji se sadrži konfiguracioni objekat i sledeće komponente: UVC-eve, scoreboard i virtualni sequencer.

6. REZULTATI TESTIRANJA

Nakon razvoja verifikacionog okruženja, počela je provera funkcionalnosti, na osnovu verifikacionog plana.

Slika 3 : Rezultat testiranja

Tokom simulacije scoreboard daje povratne informacije o tome, da li funkcionalnost dizajna radi po specifikaciji. U slučaju traženja bug-a, često se koristi i alat SimVision, koji nam daje grafički prikaz trenutne simulacije, koji u nekim slučajevima olakšava identifikaciju greške. Ovo je moguće videti na *Slika 3*.

7. POKRIVENOST

Jedan od težih zadataka u procesu verifikacije je odlučiti kada je verifikacija završena. Da bi smo došli do tog zaključka mora se dati odgovor na dva pitanja: da li su sve osobine dizajna, koje su identifikovane u verifikacionom planu, verifikovane? I, da li postoje delovi koda u dizajnu koji se nikad nisu koristili? Da bi smo dali odgovore na ova pitanja uvodi se nova metrika, pokrivenost (eng. Coverage). Više infomacija se može naći na web stranici [3]. Dve najčešće korišćene coverage metrike su:

- Strukturna pokrivenost (code coverage) - implicitna
- Funkcionalna pokrivenost (functional coverage) – eksplicitna

obe metrike, uz detaljno razrađen plan o prikupljanju pokrivenosti.

Code coverage

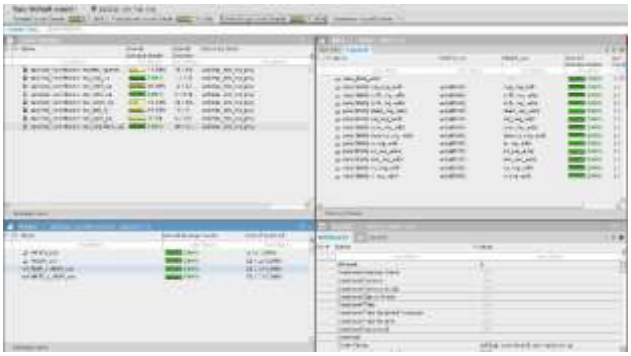
Strukturna pokrivenost ili pokrivenost koda daje informacije o stepenu aktivacije source koda tokom verifikacije čime se omogućava praćenje struktura koje se nikad ne aktiviraju. Glavna prednost ove metrike je što je implicitna odnosno kreiranje modela je automatsko. Za korišćenje pokrivenosti koda nije potrebno dodavati poseban kod i ne zahteva poseban pristup tokom verifikacije. Mana ovog tipa pokrivenosti je što je moguće imati 100% pokrivenosti, a da i dalje postoje greške u dizajnu.Postoji više tipova strukturne pokrivenosti:

- Toggle coverage
- Line coverage
- Statement coverage
- Final state machine coverage
- Branch coverage
- Expression coverage

Functional coverage

Cilj funkcionalne verifikacije je utvrditi da li dizajn implementira sve osobine i funkcioniše na način opisan u funkcionalnoj specifikaciji. Međutim do zaključka o tome da li je neka funkcionalnost stvarno implementirana i da li je verifikovana, ne može se doći na osnovu praćenja pokrivenosti koda. Zbog toga se uvodi nova, eksplicitna metrika - funkcionalna pokrivenost. Cilj ove metrike je merenje progresa verifikacije u odnosu na funkcionalne zahteve dizajna. Jedan od problema korišćenja constrained-random pristupa generisanja stimulus-a je što ne znamo tačno koje funkcionalnosti se verifikuju (šta je tačno dovedeno na ulaz DUT-a) bez da ručno analiziramo waveform-e tokom simulacije. Međutim, praćenje funkcionalne pokrivenosti nam omogućava upravo ovo – određivanje funkcionalnosti koje su verifikovane bez vizuelne analize samih signala. Mane ove metrike su što, pošto nije implicitna, ne može biti automatski implementirana. Implementacija dobrog modela pokrivenosti zahteva dosta vremena pošto je potrebno prvenstveno napraviti dobar plan, identifikovati sve osobine od interesa i odrediti način na koji će se prikupljati podaci o pokrivenosti.

Na *Slika 4* možemo videti analizu covergroup-a spi_registers_cg.



Slika 4 : Pokrivenost SPI_REGISTER_CG-a

7. ZAKLJUČAK

Glavni zadatak ovog rada je bio proučavanje modernih tehnika za verifikaciju, kako bi se napravio verifikacioni plan za modul koji je APB2SPI, i da se implementira i simulira planirani verifikaciono okruženje, koristeći UVM metodologiju i CADENCE alat za simulaciju. Verifikacioni plan i arhitektura su napravljeni uz pomoć projekt zadatka i dizajn specifikacije. Verifikaciono okruženje bazirana na modernim verifikacionim tehnikama, kao što su nasumično generisanje ulaznih promenljivih sa nekim ograničenjima, i funkcionalna pokrivenost. Tokom simulacije neke informacije su sakupljene, kako bi bilo moguće izvršiti analizu verifikovanog modula.

8. LITERATURA

[1] Sharon Rosenberg, Kathleen A Meade; A Practical Guide to Adopting the Universal Verification Methodology (UVM); Cadence Design Systems, Inc; San Jose, USA; 2010;

[2] Chris Spear, Greg Tumbush; SystemVerilog for Verification: A Guide to Learning the Testbench Language Features; Third Edition; Springer; New York, USA; 2012;

[3] http://www.elektronika.ftn.uns.ac.rs; Sajt katedra za elektroniku; Novi Sad; Predmet: Funkcionalna verifikacija hardvera; 06.09.2018;

Kratka biografija:

Čongor Lašu rođen je u Subotici 1993 god. Master rad na Fakultetu Tehničkih Nauka iz oblasti Elektrotehnike i računarstvo - Embedded sistemi i algoritmi odbranio je 2018 god.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.31
DOI: https://doi.org/10.24867/01BE11Ninkovic

PROAKTIVNA DETEKCIJA NEŽELJENOG REAGOVANJA PREKOSTRUJNIH RELEJA U SREDNJENAPONSKOJ DISTRIBUTIVNOJ MREŽI

PROACTIVE DETECTION OF UNWANTED OVERCURRENT RELAY TRIPPING IN MEDIUM VOLTAGE DISTRIBUTION NETWORK

Jelena Ninković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je opisan princip rada prekostrujne zaštite, te osnovne podele prekostrujnih releja i kriterijumi tih podela. Takođe dat je prikaz funkcije prognoza opterećenja distributivne mreže i njihov značaj u eksploataciji i planiranju EES-a. Zatim je na osnovu rezultata kratkoročne prognoze režima mreže izvršena analiza osetljivosti podešenja prekostrujne zaštite na moguća preopterećenja SN izvoda.*

Ključne reči: *prekostrujna zaštita, podešenje releja, prognoza opterećenja.*

Abstract – *This paper describes the principle of overcurrent protection, and the basic criteria and types classification of overcurrent relays. Also, it gives overview of types of load forecast functions used in a power system and its importance in system operation and planning. Further, based on the results of short-term load forecasting of the MV distribution network, possibility of bad overcurrent relay response was analyzed.*

Keywords: *overcurrent protection, relay setting, load forecast.*

1. UVOD

Dostupnost električne energije usko je povezana sa ekonomskim razvojem jedne zemlje, jer većina industrije zavisi u potpunosti od njenog korišćenja. Usled ekonomskog i industrijskog razvoja, kao i jačanja konkurencije u elektroenergetskom sektoru mnogih zemalja, distributivne kompanije se suočavaju sa porastom potreba i zahteva potrošača za električnom energijom što većeg kvaliteta – stalnom, jeftinom i održivom energijom. Kako bi se potrebe za električnom energijom mogle ispuniti potrebno je imati uvid u ponašanje sistema u budućnosti, te prognoza potrošnje igra sve bitniju ulogu u povećanju pouzdanosti i stabilnosti elektroenergetskog sistema (EES) [1].

Ideja ovog rada je analiza rezultata kratkoročne prognoze režima srednjenaponske (SN) mreže u svrhu provere osetljivosti relejne zaštite na eventualna preopterećenja SN izvoda u bliskoj budućnosti. Praktičan benefit bi bila proaktivna detekcija ispravnog, ali neželjenog reagovanja prekostrujnih releja u uslovima kada bi došlo do kratkotrajnog preopterećenja izvoda i alarmiranja/signalizacije o potencijalnom ispadu usled aktivacije releja. Time bi se blagovremeno upozorilo na mogući ispad u bliskoj budućnosti, a samim tim uticalo na povećanje pouzdanosti i sigurnosti napajanja potrošača.

U drugoj glavi data je podela i oblasti primene funkcija prognoze potrošnje električne energije. Posebno je opisana metodologija funkcija kratkoročne prognoze opterećenja na osnovu koje su rađene potrebne analize osetljivosti relejne zaštite.

U trećoj glavi opisane su osnovne podele prekostrujnih releja, kriterijumi tih podela kao i osnovna podešenja releja.

U glavi četiri analizirani su rezultati kratkoročne prognoze za dva karakteristična dana u godini. U petoj glavi izveden je zaključak rada, a u šestoj je navedena korišćena literatura.

2. PROGNOZA POTROŠNJE ELEKTRIČNE ENERGIJE

Prognoza potrošnje električne energije koristi se u svrhu predikcije ukupne satne potrošnje koju sistem mora da zadovolji u narednom periodu. Kvalitetno određena prognoza potrošnje električne energije, bazirana na što tačnijim i detaljnijim podacima, osnova je za proces planiranja distributivne mreže svih naponskih nivoa [2].

Potrebe za prognozom potrošnje električne energije dolaze do izražaja prilikom [2]:

1) planiranja prenosa i distribucije u cilju zadovoljenja rasta potrošnje, stabilnog snabdevanja i veće pouzdanosti,
2) trgovine energijom, pružajući uvid u mogućnost kupovine/prodaje električne energije u budućnosti,
3) održavanja i operativnih poslova omogućujući dispečeru da donosi pravovremene odluke o manipulaciji i planira održavanje,
4) angažovanja agregata u cilju postizanja maksimalne ekonomičnosti u radu sistema uz uvažavanje granica sigurnosti,
5) upravljanja opterećenjem – prognoza potrošnje električne energije pomaže pri donošenju odluka kad su u pitanju regulacija opterećenja i smanjivanje napona,
6) simulacionih „šta ako" analiza (engl. „What if analysis").

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Duško Bekut, red. prof.

2.1 Podela funkcija prognoze

Funkcije prognoze razlikuju se u zavisnosti od vremenskog horizonta za koji se proračunavaju i uobičajeno se dele u sledeće kategorije [1]:

1) NTLF (engl. „Near Term Load Forecast“) – kratkoročna prognoza u užem smislu sa vremenskim horizontom od naredna 24 sata i satnom vremenskom diskretizacijom,
2) STLF (engl. „Short Term Load Forecast“) – kratkoročna prognoza sa vremenskim horizontom od narednih 8 dana i satnom vremenskom diskretizacijom,
3) MTLF (engl. „Medium Term Load Forecast“) – srednjeročna prognoza sa vremenskim horizontom od narednih 8 dana i dnevnom vremenskom diskretizacijom,
4) LTLF (engl. „Long Term Load Forecast“) – dugoročna prognoza sa vremenskim horizontom od 1 do 15 narednih godina i godišnjom vremenskom diskretizacijom.

2.2 Metodologija za proračun kratkoročne prognoze potrošnje električne energije

Rezultati kratkoročne prognoze koji će biti iskorišćeni u ovom radu, dobijeni su na osnovu proračuna koji se zasniva na primeni metode vektora podrške (engl. Support Vector Machine, SVM) kao jedne vrste nadgledanog mašinskog učenja.Kratkoročna prognoza daje predviđanje opterećenja distributivne mreže (DM) za narednih 8 dana počevši od ponoći narednog dana u odnosu na trenutak izvršavanja funkcije. Rezultati prognoze prikazuju se najčešće u jednočasovnim vremenskim intervalima [3].
Funkcija zahteva sledeće ulazne parametre [3]:

1) istorijska opterećenja,
2) istorijski vremenski uslovi,
3) prognozirani vremenski uslovi,
4) tipovi dana u periodu prognoze.

Izlaz iz funkcije predstavljaju satne vrednosti potrošnje odn. aktivnih i reaktivnih opterećenja za narednih 8 dana počevši od ponoći narednog dana u odnosu na vreme pokretanja funkcije.

Istorijska opterećenja

Kvalitet i količina dostupnih istorijskih opterećenja u velikoj meri utiče na kvalitet izlaza koji daje funkcija. Ona se funkciji prosleđuju kao niz istorijskih (estimiranih, ostvarenih) vrednosti opterećenja (aktivnih i reaktivnih), za deo DM ili prenosne mreže (PM) za koji se izvršava funkcija za vremenski period od 8 dana.

Vremenske promenljive

Proces potrošnje električne energije u tesnoj je vezi sa vremenskim uslovima, pogotovu u područjima gde se električni uređaji za klimatizaciju intenzivno koriste. Neke od vremenskih promenljivih na čiju upotrebu se može često naići su temperatura vazduha, relativna vlažnost vazduha, brzina vetra, pokrivenost oblacima [4].
Među svim navedenim vremenskim promenljivim, ubedljivo najviše i najčešće korišćena je temperatura vazduha. U obzir se mogu uzeti i drugi vremenski uslovi (brzina vetra, oblačnost, padavine itd.) u zavisnosti od geografskih i klimatskih osobina oblasti DM i njihovog uticaja na potrošnju električne energije.

Tipovi dana

Karakteristike procesa potrošnje u toku dana mogu biti različite kada se radi o različitim danima u toku jedne nedelje, zbog mnogih uticaja. Na primer, zgrade u kojima se nalaze većinom poslovni prostori mogu tokom vikenda biti zatvorene, što rezultira znatno smanjenom potrošnjom u odnosu na radne dane. Takođe, ljudi ustaju kasnije tokom vikenda, pa je vrhunac jutarnje potrošnje pomeren za jedan do dva časa unapred u odnosu na radne dane. Podele na tipove dana razlikuju se od zemlje do zemlje. S obzirom na različite običaje unutar tih zemalja (radni dani u nedelji, praznici itd.) ponašanje potrošača je drugačije, a konsekventno i proces potrošnje električne energije [4].

3. PODELA I PRINCIP RADA PREKOSTRUJNE ZAŠTITE

Prekostrujna zaštita je uređaj koji reaguje na veličinu struje u štićenom delu mreže, odn. deluje kada struja pređe određenu, unapred podešenu vrednost. Služi za eliminaciju opasnih pogonskih stanja u električnim mrežama. Postoje dve vrste opasnih pogonskih stanja:

1) preopterećenje vodova,
2) kratki spojevi i zemljospoj.

Prekostrujna zaštita je najjednostavnija i najčešće primenjivana zaštita u srednjenaponskim mrežama. Primenjuje se uspešno u mrežama sa radijalnom konfiguracijom. Pobudni element zaštite je prekostrujni relej.
Postoje dve grupe prekostrujnih releja prema vremenu reagovanja [5]:

1) *trenutni,* kod kojih je delovanje trenutno (takav način delovanja koristi se u slučaju struja velikih intenziteta, a ovaj tip releja se uobičajeno obeležava sa I>>),
2) *sa vremenskim članom,* kod kojih delovanje sledi nakon određene vremenske pauze zadate vremenskim podešenjem (uobičajena oznaka je I>).

U zavisnosti od načina priključenja na kontrolisanu veličinu prekostrujni releji se mogu podeliti na [5]:

1) primarne prekostrujne okidače,
2) sekundarne prekostrujne releje.

3.1 Sekundarni prekostrujni releji

Sekundarni prekostrujni releji koriste se u kombinaciji sa strujnim transformatorima za galvansko razdvajanje kola. Prekostrujni relej (I>) se vezuje u sekundarno kolo strujnog transformatora, a jedan od sekundarnih krajeva strujnog transformatora se uzemljuje iz razloga sigurnosti. Ako struja premaši podešenu vrednost, zatvaraju se kontakti releja čime se zatvaraju kontakti kola koje se napaja iz izvora napajanja (izvora komandnog napona). Na taj način vrši se signalizacija ili se deluje na komandno kolo prekidača [5].

Prema vremenskoj karakteristici sekundarni prekostrujni releji se dele na:

1) trenutne prekostrujne releje,
2) releje sa strujno nezavisnom vremenskom karakteristikom,
3) releje sa strujno zavisnom vremenskom karakteristikom.

Prekostrujni releji sa strujno nezavisnom karakteristikom pobuđuju se kada struja premaši određenu vrednost, ali se njima deluje tek nakon nekog vremena.

Upravo po tome se ovi releji razlikuju od trenutnih gde uglavnom nema razlike između pojmova pobuđivanje i delovanje. Vreme delovanja ne zavisi od iznosa struje koja teče kroz relej [5].
Pomoću prekostrujnih releja sa nezavisnom karakteristikom može se jednostavno postići selektivnost delovanja zaštite stepenovanjem vremena delovanja zaštite od potrošača ka izvoru napajanja. Između svake dve na red postavljene zaštite mora se postići određen stepen selektivnosti koji se kreće od 0.25 do 0.5s [5].

Struja podešenja releja I_{pod} definiše se na sledeći način (relacija 3.1.1) [5]:

$$I_{pod} = \frac{k_{sigurnosti} \cdot k_{spoja} \cdot I_{radno\max}}{a \cdot n_{SMT}}, \qquad (3.1.1)$$

gde su:
$k_{sigurnosti}$ – koeficijent sigurnosti (od 1.1 do 1.2),
k_{spoja} – koeficijent spoja releja na strujne merne transformatore (kod spoja na fazne struje vrednost koeficijenta je 1, a kod spoja na razliku struja dva transformatora kvadratni koren iz 3 ,
$I_{radno\ max}$ – najveća pogonska struja,
a – koeficijent otpuštanja releja (od 0.85 do 0.95) i
n_{SMT} – odnos transformacije strujnog TR za koji je vezan relej.
Vrednosti struja podešenja prekostrujne zaštite i struja podešenja releja brojno se razlikuju za odnos transformacije strujnog mernog transformatora n_{SMT}.
Nakon izbora vremenskog i strujnog podešenja prekostrujne zaštite, potrebno je proveriti osetljivost. Struja podešenja releja mora biti manja od minimalne vrednosti struje kratkog spoja kod kvara na kraju osnovne i rezervne zone delovanja. Osetljivost zaštite definisana je koeficijentom osetljivosti k_{os} (relacija 3.1.2):

$$k_{os} = \frac{I_{k\min}}{I_{pod} \cdot n_{SMT}}, \qquad (3.1.2)$$

gde je:
I_{kmin} – minimalna vrednost struje kratkog spoja.

Keficijent osetljivosti bi trebalo da je veći od 1.5 kod kvara na kraju osnovne zone štićenja, a veći od 1.2 kod kvara na kraju rezervne zone štićenja. Minimalna struja kratkog spoja se određuje deterministički. Bira se takvo uklopno stanje za koje se na kraju zone štićenja ima najmanja struja kratkog spoja, a takođe se bira i tip kratkog spoja za koji je struja minimalna [5].

4. ANALIZA REZULTATA PROGNOZE U SVRHU DETEKCIJE (NEŽELJENOG) REAGOVANJA RELEJA

U ovom delu rada biće analizirani prognozirani režimi test DM za naredna 24h sa praktičnim benefitom proaktivne detekcije neželjenog reagovanja prekostrujnih releja na moguće preopterećenje SN vodova. Ideja je da se razmatranjem dijagrama opterećenja u bliskoj budućnosti blagovremeno detektuje i upozori na potencijalne ispravan, ali neželjen rad zaštite.

4.1 SN test mreža

Mreža se sastoji iz transformatorske stanice TS 110/20 kV/kV, 4 distributivna SN izvoda 20 kV i TS 20/0.4 kV/kV kojima je predstavljena potrošnja. Mreža sadrži uređaje relejne zaštite postavljene na početku svakog SN izvoda, što je uobičajeno u mrežama evropskog tipa. Primenjeni releji su sa strujno nezavisnom karakteristikom. Kako je mreža građena radijalno, u TS 20/0.4 kV/kV na ostalim deonicama ugrađeni su samo rastavljači. Prilikom pojave kvara na bilo kom radijalnom izvodu zaštita na njegovom početku isključuje ceo izvod trenutno ili sa malim vremenskim kašnjenjem.

4.1 Podešenja prekostrujnih zaštita u test mreži

U tabeli 4.1.1 date su proračunate maksimalne radne struje po svakom izvodu.

Tabela 4.1.1 – Vrednosti maksimalnih radnih struja po svakom SN izvodu

Oznaka izvoda	M1	M2	M3	M4
$I_{radnomax}$ [A]	101	115.5	170.3	57.7

Podešenja prekostrujnih ($I_{>z}$) zaštita izračunata su prema relaciji 3.1.1, a vrednosti struja podešenja prikazane su u tabeli 4.1.2 kao i parametri koji figurišu u relaciji 3.1.1.

Tabela 4.1.2 – Podešenja i parametri prekostrujnih zaštita po izvodima

Oznaka izvoda	I>>z [A]	$k_{sigurnostia}$	a	$I_{radnomax}$
M1	131.3	1.3	0.85	101
M2	173.2			115.5
M3	255.5			170.3
M4	86.6			57.7

Rezultati kratkoročne prognoze režima mreže za vremenski opseg od 24 sata analizirani su za dva karakteristična dana. To su:
1) karakterističan letnji dan,
2) karakterističan praznični zimski dan.

Analiza se sastoji u sledećem: Za svako stanje (sat) za koje se vrši prognoza, uzima se vrednost struje kroz prekidač na početku izvoda i poredi sa vrednošću struje podešenja prekostrujne zaštite. Ukoliko je vrednost struje kroz prekidač veća od podešenja zaštite, proverava se da li je trajanje intenziteta te struje u granicama dozvoljenih vrednosti u termičkom smislu. Ukoliko se ispostavi da je preopterećenje kratkotrajno i u granicama dozvoljenih vrednosti, a zaštita bude osetljiva na njega, zaključuje se da bi došlo do neželjenog reagovanja releja i prestanka napajanja potrošača na analiziranom izvodu.

4.3 Analiza dijagrama opterećenja za letnji dan

Za analizu je odabran radni dan, jer je uobičajeno potrošnja izraženija i veća radnim nego neradnim danom.

Slika 4.4.1 – Prognozirani dijagram potrošnje za karakteristični zimski dan

Odabran letnji dan karakteriše i izrazito visoka dnevna temperatura (sa dnevnim prosekom oko 30℃) što rezultuje u povećanom korišćenju rashladnih uređaja, a samim tim znatno utiče i na potrošnju struje i oblik dnevnog dijagrama opterećenja. Pomenuti vremenski uslovi uvaženi su prilikom izvršavanja funkcije prognoze opterećenja.

Analizom prognoziranih vrednosti opterećenja SN izvoda i poređenjem intenziteta radne struje za svako stanje (sat) sa podešenjem zaštite, zaključeno je da nema perioda sa preopterećenjem te da za ovaj slučaj (analiziran letnji dan) ne bi došlo do ispravnog, ali neželjenog reagovanja releja usled preopterećenja.

4.4 Analiza dijagrama opterećenja za zimski dan

Za analizu je odabran novogodišnji praznik, odn. 31.decembar i 1. januar. Pretpostavljeno je da izabrane zimske dane karakteriše izrazito niska temperatura sa dnevnim prosekom od -10℃ i ta činjenica je uzeta u obzir prilikom proračuna prognoze opterećenja. Niske temperature utiču na povećanu upotrebu grejnih tela (TA peći, grejalice, toplotne pumpe) koja su najveći potrošači u domaćinstvu. Na dodatno povećanje potrošnje u ovoj situaciji ima uticaja i upotreba ulične i druge novogodišnje rasvete.

Analizom prognoziranih dijagrama opterećenja SN izvoda ustanovljeno je da je funkcija prognozirala preopterećenje izvoda M1 u periodu od 21:00 do 23:59 31. januara, i od 18:00 do 20:00 1. januara. Dijagram opterećenja za izvod M1 prikazan je na slici 4.4.1. Intenzitet radne struje izvoda M1 u periodu preopterećenja, 31. decembra od 22:00 do 23:59, kreće se u opsegu (107 ÷111 A). Ove vrednosti su veće od vrednosti maksimalne radne struje proračunate za izvod M1 (101 A), ali opet manje od vrednosti podešenja zaštite. Međutim kako je preopterećenje kraćeg trajanja, a prosečna temperatura vazduha -10℃ (vod se sporije zagreva), štićeni element bi se kratkoročno mogao preopteretiti u periodu povećane potrošnje, te je zaključak da bi došlo do preopterećenja ali ne i do nepotrebnog reagovanja zaštite i ispada izvoda M1 iz pogona.

5. ZAKLJUČAK

Funkcije prognoze potrošnje električne energije prvobitno su najveću ulogu imale u planiranju EES-a, odnosno u istraživanju i određivanju optimalnih mera razvoja EES-a u budućnosti (planovi izgradnje novih kapaciteta, angažovanje agregata, itd). Međutim poslednjih decenija razvojem preciznijih algoritama kratkoročnog predviđanja potrošnje, ova funkcija počela je da dobija sve veći značaj i u pogledu održavanja i operativnih poslova omogućujući donošenje pravovremenih odluka o manipulaciji. Na taj način utiče se i na povećanje pouzdanosti i sigurnosti elektroenergetskog sistema u neposrednoj budućnosti.

U ovom radu je opisan i analiziran još jedan benefit funkcija prognoze, a to je njihova primena u analizi pogona mreže. Proaktivna detekcija reagovanja relejnih uređaja na preopterećenje u mreži omogućuje preventivno delovanje i poboljšanje performansi jedne mreže. Primena adaptivne zaštite bi svakako bila sigurnije rešenje, ali je nekada upitna njena ekonomska isplativost. Uvidom u stanje mreže u budućnosti moglo bi se blagovremeno reagovati na prognozirana preopterećenja, izvršiti prepodešnje releja ili adekvatna promena topologije, u zavisnosti da li je reagovanje releja opravdano ili ipak bezrazložno.

6. LITERATURA

[1] Ilija Vujošević, "*Eksploatacija i planiranje elektroenergetskih sistema*", Elektrotehnički fakultet, Podgorica, 2005.

[2] Milan Ćalović, Andrija Sarić i Predrag Stefanov, „*Eksploatacija elektro-energetskih sistema u uslovima slobodnog tržišta*", Tehnički fakultet, Čačak, 2005.

[3] Miloš Božić „Kratkoročna prognoza potrošnje električne energije zasnovana na metodama veštačke inteligencije", doktorska disertacija, Niš, 2014.

[4] Slobodan Ilić, „*Kratkoročno predviđanje električne energije u velikim elektroenergetskim sistemima*", doktorska disertacija, Novi Sad, 2013.

[5] Duško Bekut, „*Relejna zaštita*", FTN izdavaštvo, Novi Sad, 2013.

Kratka biografija:

Jelena Ninković rođena je u Šapcu 1992. godine. Diplomirala je na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Elektroenergetski sistemi 2016. godine i nakon toga upisala master studije na istom fakultetu.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.31
DOI: https://doi.org/10.24867/01BE12Gajunovic

PRIMENA NETWORK PROTECTOR-A U PAMETNIM DISTRIBUTIVNIM MREŽAMA

APPLICATION OF NETWORK PROTECTORS IN SMART DISTRIBUTION NETWORKS

Marko Gajunović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je analiziran rad Network Protector-a i njihova primena u pametnim distributivnim mrežama. Opisana je funkcionalnost i način rada Network Protector-a, kao i Load Transfer Management funkcija koja se zasniva na radu Network Protector-a. Razmatranja su ilustrovana odgovarajućim primerom.*

Ključne reči: *Network Protector, pametna distributivna mreža.*

Abstract – *This paper presents analysis of Network Protectors operation and their use in Smart Distribution Grids. The functionality and operaton of Network Protectors is described, as well as Load Transfer Management function which is based on Network Protectors operation. Considerations are illustrated by an example.*

Keywords: *Network Protector, Smart distribution grid.*

1. UVOD

Potreba za električnom energijom je u stalnom porastu pa stoga raste i potrošnja električne energije. Porast potrošnje električne energije utiče na složenost elektroenergetskih sistema i mreža, što za posledicu ima kompleksnost pri određuvanju mesta kvara, uklanjanju kvara i restauraciji napajanja.

U ovom radu je obrađena tema Network protector-a (u daljem tekstu samo protektori), i njihova primena u pametnim distributvnim mrežama.

U drugom poglavlju objašnjen je pojam pametne distributivne mreže, prednosti i benefiti modernizacije elektroenergetskih sistema i korišćenja savremenih informacionih tehnologija u upravljanju istim.

Treće poglavlje obuhvata protektore, njihov sastav i princip rada. Važno je bilo opisati način primene protektora u distributivnim mrežama.

Četvrti deo se bavi analizom metoda rada proračuna koje koriste protektore. Konkretno, ovde je opisana Load Transfer Management funkcionalnost koja je razvijena u cilju poboljšanja pouzdanosti napajanja i smanjenja perioda neisporučivanja energije potrošačima usled kvarova. Protektori ovde imaju ključnu ulogu.

U petom poglavlju opisan je uticaj distributivnih generatora na struje kratkih spojeva i na rad Protektora.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Duško Bekut, red. prof.

Distributivni generatori se sve češće i sve više ugrađuju u distributivne niskonaponske mreže kao kao vetrogeneratori i solarni paneli, pa je stoga bilo važno pomenuti i njihov uticaj na rad zaštite.

U šestoj celini je predstavljeno više primera korišćenja protektora. Ovde je prikazana prethodno opisana Load Transfer Management funkcionalnost na konkretnom primeru. Primeri su urađeni sa ciljem da se lakše razume kako zapravo protektori deluju prilikom kvarova u srednjeponskim mrežama i kako se pomoću funkcije upravljanja prebacivanjem opterecenja – Load Transfer Management-a vrši prebacivanje opterećenja sa jednog srednjenaponskog voda na drugi.

U sedmom poglavlju su doneti zaključci o prednostima upotrebe protektora i benefitima koje donose.

Spisak korišćene literature za izradu ovog rada dat je u poslednjem delu.

2. PAMETNA MREŽA

Pametna mreža se definiše kao elektroenergetski sistem koji je u mogućnosti da integriše navike i akcije svih korisnika, uključujući proizvođače, distribuciju i potrošače električne energije, kako bi se osigurala ušteda električne energije i energetska efikasnost sa manjim gubicima i redovnim snabdevanjem potrošača [1]. Ovakav sistem omogućuje praćenje, analizu kontrolu i komunikaciju unutar elektroenergetskog sistema.

Klasične elektroenergetske sisteme je karakterisao tok električne energije u jednom smeru, od velikih proizvođača ka potrošačima, što se vremenom pokazalo kao neefikasna infrastruktura koja nije mogla da zadovolji potrebe sve većeg broja potrošača. Sa svakim proširenjem ovi sistemi su postajali sve neekonomičniji, sve teži za održavanje i sve nesigurniji.

Sa druge strane, pametne mreže karakteriše dvosmeran protok električne energije kao i dvosmerna komunikacija u realnom vremenu. Ono što je još karakteristično za pametne distributivne mreže su uređaji za skladištenje električne energije i pametni merni uređaji. Pametne mreže i dalje zavise od velikih elektrana ali sadrže i veliki broj uređaja za skladištenje električne energije kao i veliki broj manjih obnovljivih izvora električne energije koji su najčešće implementirani u distribuciju [2]. U pametnim mrežama važno mesto zauzimaju senzori, kontrolni centri i direktno učešće potrošača u upravljanju energetskim tokovima.

Pametna mreža se u velikoj meri oslanja na prikupljanje podataka, nadzor i komunikaciju, stoga je potrebno informacionu tehnologiju integrisati u postojeću strukturu

elektroenergetskog sistema. To podrazumeva korišćenje telekomunikacionog sistema i radio frekvencije kao vida komunikacije između proizvođača, distributera i potrošača.

Automatizacija distribucije se uglavnom povezuje sa daljinskim nadzorom i upravljanjem srednjenaponskih postrojenja, ali implementacija u distribuciju omogućuje korišćenje većeg broja složenih funkcija kao što su npr. funkcije upravljanja kvarovima, funkcije planiranja itd.

3. PROTEKTORI

3.1 Princip i sastav protektora

Protektori predstavljaju zaštitne uređaje koji se koriste u distributivnim mrežama. Zadatak protektora je da automatski odvoji servisni transformator koji napaja niskonaponsku sekundarnu mrežu kada se pojavi tok snage u suprotnom smeru, tj. od niskonaponske ka srednjenaponskoj mreži.

Protektor predstavlja sklop od više elemenata. Sastoji se od vazdušnog prekidača i releja koji reaguje na suprotan smer toka snage. Ukoliko se desi kvar na primarnoj strani servisnog transformatora, dolazi do toka struje iz sekundarne distributivne mreže ka mestu kvara koje se napaja i taj tok struje se odvija preko servisnog transformatora. Uloga protektora je da spreče napajanje mesta kvara sa strane sekundara servisnih transformatora, odnosno iz niskonaponske distributivne mreže. U slučajevima kada se pojavi obrnut tok struje na servisnom transformatoru, na protektorima se energizuju uređaji koji daljim akcijama automatski otvaraju prekidač čime se prekida veza između distributivne mreže i mesta kvara. Jedna od osnovnih manifestacija kratikih spojeva u mrežama je pad napona koji se dešava tokom istih. U slučaju kratkog spoja na srednjenaponskom napojnom vodu, na servisnom transformatoru će doći do osetnog pada napona na njegovoj primarnoj strani. Kada se kvar ukloni i kada se napon na primarnoj strani vrati u granice u kojima je bio pre nastanka kvara, na protektoru će se zatvoriti prekidač čime se ponovo uspostavlja veza između distributivne mreže i napojnog voda.

3.2 Niskonaponske distributivne mreže u kojima se koriste protektori

Postoji više različitih struktura i konfiguracija distributivnih mreža.

Najčešće korišćena i najrasprostranjenija je radijalna niskonaponska mreža koju karakteriše jednostavnost i relativno visok nivo puzdanosti [3]. Upotreba protektora u ovim mrežama je izuzetno retka i obično se kao zaštita koristi neusmerena prekostrujna zaštita.

Dvostruko napajana prstenasta mreža je mreža koju karakteriše mogućnost rezervnog napajanja koje se postiže uz pomoć rezervnih vodova. Protektori se često koriste kao zaštita u ovim mrežama.

Petljaste niskonaponske distributivne mreže se koriste u uslovima velike gustine potrošača i karakteriše ih velika upetljanost. Protektori se i kod ovih mreža često susreću kao vidovi zaštite [4].

3.3 Kontrola i simulacija rada protektora u pametnim distributivnim mrežama

Protektori, odnosno prekidači koji su u njihovom sklopu, mogu biti daljinski ili lokalno upravljani. Kod daljinski upravljanih prekidačkih elemenata, preko SCADA sistema, operator može iz kontrolnog centra da ima uvid u status prekidača. Kod lokalnih prekidačkih elemenata operator u kontrolnom centru ne može da ima uvid u njihove trenutne statuse, što dovodi do netačne topologije u softveru, što dalje dovodi do netačnih rezultata izvršavanja nekih funkcija, kao što su naprimer tokovi snaga. Da bi se izbegla nepodudaranje između topologije u softveru i aktuelne topologije na terenu potrebno je da softver simulira stanja lokalnih prekidačkih elemenata.

Simulacija promene statusa protektora na sistemu koji radi u realnom vremenu (Realtime) se vrši za lokalne prekidače, dok se na simulacionim i test sistemima vrši simulacija promene statusa protektora za sve prekidače, i lokalne i daljinsko upravljane. Simulacija promene stanja prekidača se inicira promenom topologije mreže. Ako softver, tj. aplikacija utvrdi da promena topologija inicira reagovanje protektora, desiće se odgovarajuća prekidačka akcija. Te prekidačke akcije se u sistemu prosleđuju određenim servisima i skladište se na određene lokacije. Funkcija simuliranja promene statusa protektora ima svoje ulaze, inicijatore i izlaze. Ulazi su topologija, trenutno stanje mreže i aktuelni statusi protektora, odnosno njihovih prekidačkih elemenata. Glavni izlaz predstavljaju prekidačke akcije prekidačkih elemenata koje promena topologije inicira. Promene u topologiji predstavljaju inicijatore za izvršavanje ove funkcije.

Deenergizacija pomoću protektora u radijalanim distributivnim mrežama zahteva jednu prekidačku akciju. Međutim, u slučaju slaboupetljanih i upetljanih distributivnih mreža potrebno je dve, a često i više akcija da bi se izolovao kvar. Za radijalnu distributivnu mrežu potrebno je samo da se otvori prekidač koji napaja niskonaponsku mrežu, dok je u upetljanim distributivnim mrežama potrebna složena prekidačka akcija koja se sastoji od:

1. Otvaranje prekidača na vodu kojem se desio kvar.
2. Otvaranje svih prekidača koji imaju protektore na servisnim trenasformatorima koji se napajaju sa tog voda.

3.4 Relejna zaštita u sklopu protektora

Releji koji se koriste u sklopu protektora mogu biti različitih funkcionalnosti u zavisnosti od potrebe i isplativosti njihove ugradnje. Najčešće se koriste strujni releji i releji snage koji za ulazne veličine imaju i struju i napon, i to uglavnom usmereni releji snage, zatim releji napona, odnosno podnaponski releji koji detektuju pad napona pri kreatkim spojevima, dok se u poslednje vreme sve više koriste multifunkcionalni i adaptivni releji koji mogu da kontrolišu više veličina.

4. LOAD TRANSFER MANAGEMENT

S ciljem smanjenja vremena trajanja kvara i obezeđivanja brzog povratka napajanja koriste se automatske logike koje vrše kontrolu prekidačkih elemenata.

Automatska logika podrazumeva programabilnu logiku koja izvršava operaciju tako što se implementira u uređaj. Cilj automatske logike je koja se koristi u okviru Load Transfer Management funkcije je da u slučaju kvara na srednjenaponskom vodu njegovo opterećenje prebaci na rezervni vod uz pomoć prekidača koji imaju implementiranu prekidačku logiku. Load Transfer Capacity Check i simulacija kontrolne logike su glavne funkcionalnosti Load Transfer Management-a.

4.1 Algoritam funkcionisanja Load Transfer Management-a

Simulacija kontrolne logike se okida nakon promene topologije što inicira analizu kontrolne logike za sve elemente koji se nalaze unutar afektovanog dela mreže. Ako aplikacija, odnosno softver utvrdi da promena topologije inicira operaciju automatske kontrolne logike, odgrovarajuća prekidačka akcija će se predložiti. Prekidačka akcije se šalju odgovarajućim servisima i upisiju se u odgovarajuće liste. Svaka operacija se upisuje nakon vremena kašnjenja koje je specifirano kontrolnom logikom. Ako se prime novi rezultati za istu automatsku kontrolnu logiku pre nego što vreme kašnjenja istekne, novi rezultati neće biti prihvaćeni. Nakon isteka vremena kašnjenja, stanje automatske kontrolne logike ponovo je dostupno. Ako je stanje još uvek aktivno, prekidačka sekvenca se šalje na izvršenje. U suprotnom, ona će biti uklonjena. Nakon slanja sekvence na izvršenje čeka se potvrda od komandnog servisa koja potvrđuje da je sekvenca uspešno izvršena. Nakon dobijanja takve potvrde, sekvenca se označava kao uspešno izvršena i briše se iz liste.

Nakon upešnog izvršavanja prekidačke sekvence automatske kontrolne logike, topologija se menja. Pošto je topologija promenjena, aplikacija će biti pokrenuta ponovo, inače softver neće uzeti u obzir rezultate automatske kontrolne logike.

4.2 Podaci i rezultati Load Transfer Management funkcije

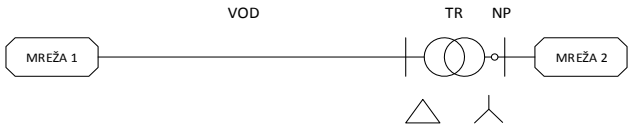
Glavni podaci Load Transfer Capacity Check funkcionalnosti su topologija, stanje mreže i aktuelni statusi kontrolnih logika u mreži. Glavni izlazi koji se dobijaju izvršavanjem ove funkcionalnosti su: omogućavanje ili onemogućavanje kontrolne logike u zavisnosti od kapaciteta rezervnog voda, povratna informacija o vrednosti kapaciteta koji nedostaje rezervnom vodu (ukoliko rezervni vod nema dovoljno kapaciteta), odgovarajući alarm i notifikacija.

5. UTICAJ DISTRIBUTIVNIH GENERATORA NA STRUJE KVARA I PROTEKTORE

Problem koji se dešava kod protektora prilikom povećanja penetracije distributivnih generatora u distributivnim mrežama je problem obnutog toka snage od mreže ka transformatoru. Prilikom kratkog spoja na srednjenaponskom vodu, dešava se obrnuti strujni tok od niskonaponske disrtributivne mreže ka transformatoru. Međutim, dešava se, i to najčešće u slučajevima minimalnog opterećenja, da se pojavi tok snage od niskonaponske distributivne mreže ka servisnom transformatoru.
Posledica toga jeste delovanje protektora iako se kvar zapravo nije ni desio. Na osnovu ovoga se lako može zaključiti da povećanje broja distributivnih generatora u mreži utiče na povećan broj delovanja protektora. Distributivni generatori mogu imati i pozitivan uticaj na rad protektora.
Primer toga je operacija protektora tokom Load Transfer Management funkcije. Pozitivan uticaj distributivnih generatora je taj što mogu da smanje opterećenje rezervnog voda, pa je time mogućnost preopterećenja voda manje verovatna.

6. PRIMER RADA PROTEKTORA PRILIKOM KRATKOG SPOJA NA NAPOJNOM VODU

U ovom delu prikazan je jedan jednostavan elektroenergetski sistem koji se sastoji od dve mreže od kojih jedna pripada 10 kV, a druga 0.4 kV naponskom nivou. Mreže su povezane preko srednjenaponskog voda 10 kV na čijem se početku simuliraju kratki spojevi i servisnog 10/0.4 kV/kV transformatora.



Slika 6.1 – Primer elektroenergetskog sistema sa 10 kV i 0.4 kV naponskim nivoima

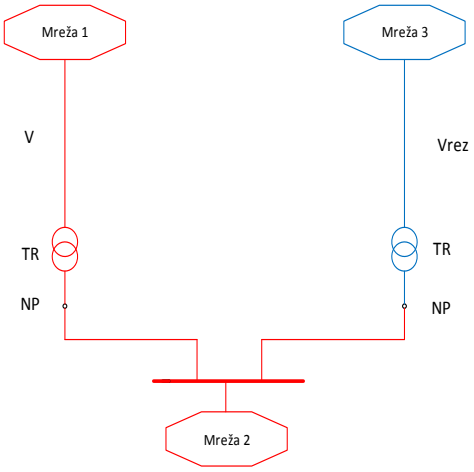
Cilj zadatka je da se izračunaju struje kratkih spojeva koje će kroz transformator da teku iz niskonaponske mreže ka mestu kratkog spoja na vodu.

Tabela 6.1 – Vrednosti struja kratkih spojeva kroz servisni transformator

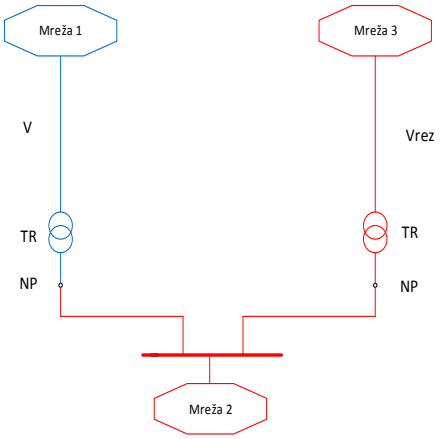
	Struja kratkih spojeva po fazama kroz SN/NN transformator [A]		
	L1	L2	L3
I_{1PKS}	120.52	60.26	60.26
I_{2PKS}	0	174.81	174.81
I_{3PKS}	201.85	201.85	201.85

U drugom delu primera je prikazan način funkcionisanja Load Transfer Management funkcija, odnosno prebacivanje napajanje sa glavnog na rezervni napojni vod. Pored već prikazanog dela mreže, postoji i rezervni vod preko kojeg bi se napajala niskonaponska distributivna mreža u slučaju ispada glavnog voda.

Distributivna mreža je sa rezervnim vodom povezana na isti način kao i sa glavnim, preko servisnog transformatora, s tim što je prekidač u sklopu protektora otvoren, pa je taj transformator praktično u praznom hodu.



Slika 6.2 – Napajanje distributivne niskonaponske mreže preko glavnog napojnog voda



Slika 6.3 – Napajanje distributivne niskonaponske mreže preko rezervnog napojnog voda

7. ZAKLJUČAK

Problem kod dvostrano napajanih distributivnih mreža i upetljanih distributivnih mreža koje se napajaju iz više različitih srednjenaponskih vodova nastaje u slučaju kratkog spoja na nekom od tih vodova koji napaja mrežu. Tada postoji tok struje ka mestu kratkog spoja ne samo iz delova sistema koji su direktno povezani sa tim vodom nego i sa ostalih vodova i to preko upetljanih i dvostrano napajanih distributivnih mreža koje ti vodovi napajaju. Iz tog razloga upotreba protektora je ključna u zaštiti niskonaponskih distributivnih mreža od spoljnih kratkih spojeva.

Benefit upotrebe protektora jeste ne samo zaštita niskonaponskih mreža nego i zaštita susednih srednjenaponskih vodova od preopterećenja usled kratkih spojeva. Protektorima se, takođe, smanjuje vreme trajanja kvara, pouzdanost i kvalitet napajanja, jer pomoću protekrora je moguće prebacivanje opterećenja sa voda na kojem se desio kvar na susedni, rezervni vod.

8. LITERATURA

1. F. Skopik, P. Smith, Smart Grid Security: Inovative Solution for Modernized Grid, 2015.
2. K. A. Wannan, Enhancement of Distribution Networks through utilization of Smart Grid, Dubai Electrecity & Water Authority (DEWA) .
3. D. Bekut, Relejna zaštita, Novi Sad: FTN Izdavaštvo, 2009.
4. M. Behnke, Secondary Network Distribution Systems Background and Issues Related to the Interconnection of Distributed Resources, National Renewable Energy Labaratory, 2005.

Kratka biografija:

Marko Gajunović rođen je 1991. godine u Gradačcu. Fakultet tehničkih nauka u Novom Sadu upisao je školske 2010/2011, smer energetika, elektronika i telekomunikacije. Na osnovnim strudijama diplomirao 2015. godine. Diplomski-master rad iz oblasti elektrotehnika i računarstvo odbranio 2018. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38:620.9
DOI: https://doi.org/10.24867/01BE13Nikolic

MOGUĆNOST KORIŠĆENJA SOLARNE ENERGIJE ZA NAPAJANJE SISTEMA ZA ODLAGANJE PEPELA I ŠLJAKE U TE KOSTOLAC

POSSIBILITY OF USING SOLAR ENERGY FOR POWER SUPPLY OF THE ASH AND SLAG DISPOSAL SYSTEM IN THE THERMAL POWER PLANT KOSTOLAC

Zoran Nikolić, Vladimir Katić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu prikazano je idejno rešenje fotonaponske (FN) elektrane na deponiji pepela i šljake "Ćirikovac" termoelektrane Kostolac B, kao alternativa konvencionalnom sistemu za odlaganje pepela i šljake, koji se napaja iz sopstvene potrošnje termoelektrane. Opisan je način funkcionisanja ovog sistema, kao i njegova ukupna potrošnja. U sklopu idejnog rešenja obrađeni su svi ključni elementi FN elektrane. Obrađena je godišnja proizvodnja električne energije čitavog sistema i upoređeno sa zadatom potrošnjom. Na kraju je urađena je procena isplativosti gradnje FN elektrane.*

Ključne reči: *termoelektrana, deponija pepela i šljake, fotonaponska elektrana.*

Abstract – *In this paper, the preliminary design of the photovoltaic (PV) power plant is presented at the landfill of ash and slag "Ćirikovac" in the thermal power plant Kostolac B, as an alternative to a conventional system for depositing ash and slag powered by own system in thermal power plant. The way of functioning of disposal system of the ash and slag is shown. All the main elements of the PV plant that are being worked out during the design process are presented. The annual electricity production of the entire system was processed and compared with the consumption of the conventional system. At the end, an estimate of the feasibility of the PV system has been made.*

Keywords: *thermal power plant, ash and slag landfill, photovoltaic power plant*

1. UVOD

Jedan od značajnijih izvora gasova staklene bašte (vodena para H_2O, ugljen dioksid CO_2, metan CH_4, azot suboksid N_2O, gasovi koji sadrže fluor HFC, PFC, SF_6 i dr.), koji se smatraju uzrocima klimatskih promena i pojava ekstremnih vremenskih prilika, je energetski sektor, a posebno termoelektrane. Termoelektrane imaju veliki uticaj na životnu sredinu, ako se uzmu u obzir i ostali aspekti zagađivanja, kao što su drugi dimni gasovi, a posebno čađ i sumpor dioksid SO_2, čestice ugljene prašine, pepeo, šljaka, otpadne vode i dr. [1,2].

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladimir Katić, red.prof.

Međutim, s obzirom da se radi o objektima koji su od vitalnog značaja za jednu državu, posebno što se radi o elektroenergetskom sektoru, ovom problemu se pristupa sa velikom pažnjom. Preduzimaju se brojne metode smanjivanja negativnog uticaja, kao što su ugradnja filtera, smanjivanje sadržaja sumpora u uglju, odlaganje pepela i šljake, zaštita pepelišta ovlaživanjem i dr., što poskupljuje izlaznu cenu električne energije iz termoelektrana. Posebno su složeni sistemi za odlaganje pepela i šljake, koji zauzimaju velike površine i zahtevaju brojne sisteme za rad i održavanje.

S tim u vezi, aktuelizovano je pitanje prelaska na druge izvore energije, jer je evidentno da se ovakav trend ne može održati. Najčešće se razmatra prelazak na obnovljive izvore energije, od kojih vetar, sunce i hidro energija imaju najveće mogućnosti. Ipak, sunčeva energija predstavlja najperspektivniji oblik energije u smislu konstrukcije, pouzdanosti, pa i ekologije.

U ovom radu je prikazana mogućnost izgradnje fotonaponske (FN) elektrane koja će služiti za pokrivanje potrošnje sistema za odlaganje pepela i šljake u termoelektrani Kostolac B. Ovaj rad ima za cilj da ukaže na opravdanost gradnje ovakve elektrane u kompleksu termoelektrane, a takođe i da pokaže ekonomsku isplativost jednog ovakvog projekta.

2. OPIS RADA TERMOELEKTRANE

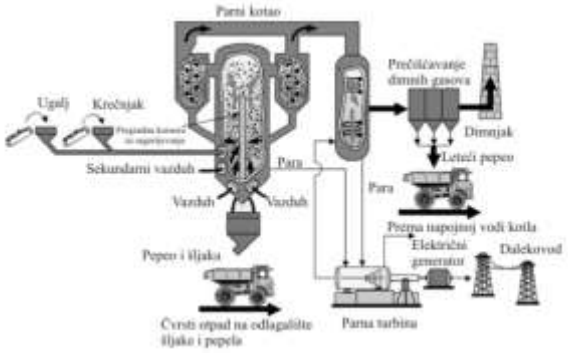
Termoelektrana je postrojenje u kome se hemijska ili nuklearna energija goriva pretvara u toplotnu energiju, a zatim se ta toplotna energija koristi za proizvodnju vodene pare, koja pod visokim pritiskom pokreće turbine i obezbeđuje mehaničku rotacionu energiju, koja se dalje koristi za pokretanje generatora električne energije [1].

Klasična termoenergetska postrojenja za svoj pogon koriste fosilna ili nuklearna goriva, koja dobijenu unutrašnju energiju goriva u tehnološkom procesu sagorevanja pretvaraju u električnu energiju (kondenzacione elektrane), energiju toplote (toplane) ili električnu i toplotnu energiju (termoelektrane-toplane, nuklearne toplane-elektrane i sl.). Skica termoelektrane sa ugaljem kao gorivom na kojoj se može videti princip rada data je na slici 1.

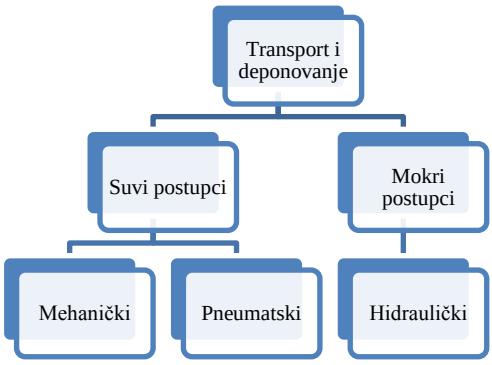
3. SISTEM ZA ODLAGANJE PEPELA I ŠLJAKE U TE KOSTOLAC

Kao ostatak iz procesa sagorevanja uglja javlja se pepeo i šljaka, a iz procesa filtriranja dimnih gasova i leteći pepeo. Ova količina otpadne materije kod nas uglavnom nema upotrebnu vrednost, mada se posebnim postupcima

mogu od nje praviti neki građevinski materijali ili koristiti na neki drugi način. Način deponovanja pepela išljake može se vršiti po postupcima prikazanim na slici 2. Postoje suvi i mokri postupci, sa daljim obradama. U svim termoelektranama u Srbiji, kao najbolji način deponovanja primenjuje se hidraulički transport, pri čemu postoji tehnologija guste ili retke hidromešavine.

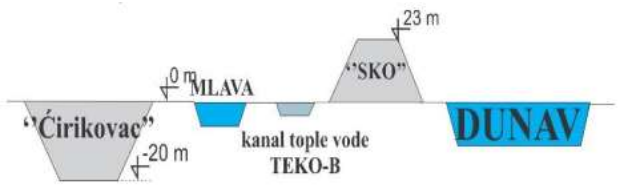


Slika 1. Princip rada termoelektrane [1]



Slika 2 . Načini deponovanja pepela i šljake

U cilju stvaranja uslova za bezbedno odlaganje i znatno poboljšanje zaštite životne sredine od štetnog uticaja deponije, u termoelektrani „Kostolac B“ tehnologija hidrauličkog transporta retke hidro-mešavine sa odnosom „čvrsto:tečno=1:10“ zamenjena je tehnologijom guste kontrolisane hidrosmeše odnosa 1:1 (50%:50%) sa mogućnošću povratka vode sa deponije. Transport guste hidromešavine od silosa do deponije Ćirikovac vrši se sa 4 cevovoda i jednim cevovodom za povratnu vodu. Dužina trase cevovoda od silosa do deponije iznosi 6.060 m. Deponija pepela i šljake je sa nivelacijom -20 m, kako je pokazano na slici 3.

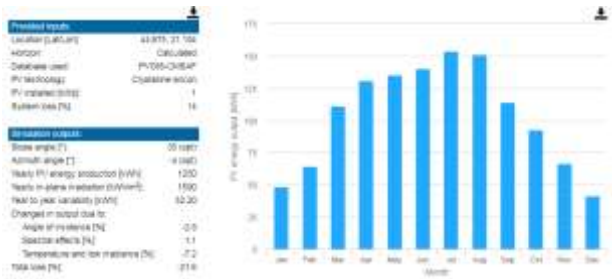


Slika 3. Prikaz deponije kroz pesek terena.

Čitav kompleks sistema guste hidrosmeše postavljen na deponiji u Ćirakovcu, sastoji se iz 4 osnovne celine: unutrašnji sistem za pepeo i šljaku, kompleks silosa, spoljašnji sistem transporta i deponija pepela i šljake. Ukupna potrošnja električne energije ova 4 sistema iznosi 58.377 MWh na godišnjem nivou, dok su troškovi zagađenja procenjeni na 336.485 USD godišnje. U ovom radu su pretpostavljene minimalne cene troškova emisije CO_2 u atmosferu, a to je 6,55 USD/t i emisija CO_2 od 0,88 t/MWh u skladu sa studijom opravdanosti koja je rađena prilikom donošenja odluke o izgradnji bloka B3 na Kostolcu [3].

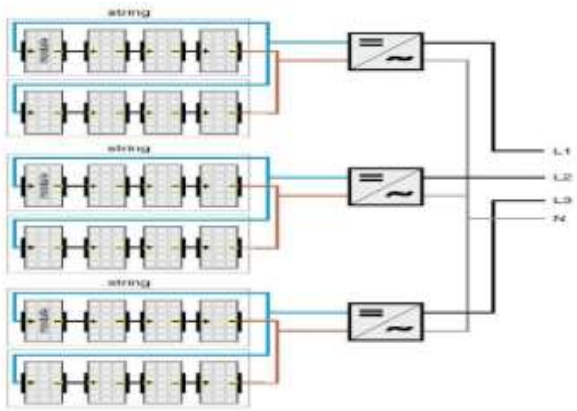
4. IDEJNO REŠENJE FN SISTEMA

Potencijalna FN elektrana na deponiji pepela i šljake u Ćirikovcu ima za cilj da napomesti sopstvenu potrošnju u sistemu za odlaganje pepela i šljake i da doprinese smanjenju emisije CO_2 u atmosferu. S tim u vezi, idejno rešenje je urađeno kroz četiri koraka: procena energetskog potencijala lokacije, izbor koncepta FN elektrane, određivanje položaja panela i ostalih komponeneti i izbor optimalnih komponenti. Za procenu energetskog potencijala korišćen je softver PVGIS i dobijeni su sledeći rezultati na slici 4.
Na slici 4 se može videti da se za 1 kWp instalisane snage može proizvesti 1.250 kWh električne energije godišnje što je veoma bitan podatak prilikom dimenzionisanja FN elektrane. Takođe, optimalni ugao montaže je 35°, dok je optimalni azimutni ugao jednak -4°.



Slika 4. Prikaz prosečne proizvedene energije po mesecima za 1 kWp instalisane snage

Prilikom izbora koncepta FN elektrane, izabran je koncept da se jedan invertor koristi za povezivanje vise stringova kako je prikazano na slici 5. Ovaj princip primenjuje se kod velikih FN elektrana, gde su FN polja dosta velika i podeljenja na više delova. Svaki od tih delova može biti priključen na sopstveni invertor. Obično se ovakvi invertori zovu distribuirani invertori. Broj invertora koji će biti ugrađen zavisi od instalisane snage same elektrane.



Slika 5. Jedan invertor za više stringova

Tip modula koji je izabran u ovom slučaju je: Schutten solar STP6-300/72 od polikristalnog silicijuma snage 300

Wp, dok su izabrani invertori tipa Ingeteam INGECON SUN 100 snage 110 kW. Noseća konstrukcija koja je izabrana za ovaj slučaj je fiksna, proizvođača Nika-Solar, koja ima mogućnost montaže dva reda modula pri čemu je broj modula u svakom redu proizvoljan i zavisi od projektovanog rešenja. Ostale komponente kao što su DC i AC kablovi, zaštitni uređaji, sklopni uređaji, osigurači i dr. birani su u skladu sa izvršenim proračunom.

Dimenzionisanje FN elektrane izvršeno je tako što je preklopljen dijagram potrošnje sistema za odlaganje pepela i šljake i dijagram procenjene proizvodnje FN elektrane na godišnjem nivou. Na taj način, snaga FN elektrane koja bi pokrila potrošnju navedenog sistema iznosi 46.701 kWp.

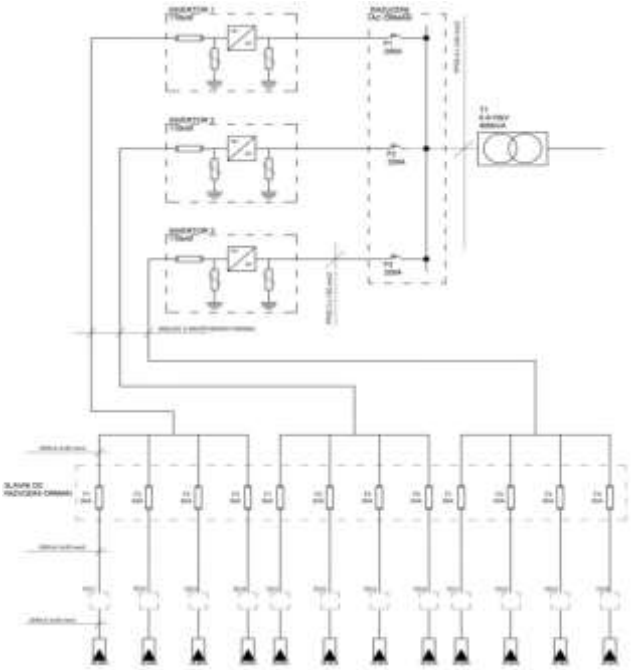
U skladu sa snagom FN elektrane, biće potrebno 155.670 modula, 19.459 m nosećih konstrukcija i 425 komada invertora. FN paneli su projektovani tako da se sastoje iz 8 modula, pri čemu su raspoređeni u 2 reda po 4 modula i montirani na noseću konstrukciju koja je fiksna i ima mogućnost promene nagiba preko zubaca sa zadnje strane konstrukcije. Rastojanje između panela, tj. nosećih konstrukcija je optimizovano na osnovu položaja sunca u kritičnim danima, 21. decembra i 21. juna, pri čemu minimalno rastojanje iznosi 6,45 m. Za predviđena rastojanja može se instalirati 540 kWp/ha. Kako je snaga FN elektrane 46.701 kWp, biće potrebno otprilike 86 ha za instalaciju panela. Ideja je da se paneli postave na konstrukciju iznad deponije. FN elektana navedene snage može u potpunosti da se instalira na deponiji pepela i šljake u Ćirikovcu jer je površina cele deponije 130 ha. Takođe, to ne bi ugrozilo ni funkcionisanje same deponije. Na slici 6 prikazana je jedna od varijanti rasporeda panela na deponiji:

Slika 6. Predlog rasporeda panela na deponiji

Paneli se mogu rasporediti i na drugi način. Kako paneli zauzimaju oko 86 ha, preostali prostor može biti iskorišćen u budućnosti ukoliko bude bilo potrebe za širenjem FN elektrane.

Celokupna FN elektrana sastoji se iz više manjih delova (blokova) da bi se dobilo na pouzdanosti. Svaki blok se sastoji od 3 invertora snage 110 kW koji su vezani na blok transformator od 400 kVA. Ovi transformatori moraju biti povezani sa trafostanicom 110/10 kV/kV tj. sa srednje-naponskom mrežom. Ukupan broj blokova koji pokriva potrebnu snagu FN elektrane iznosi 142. Jednopolna šema jednog bloka od 330 kW FN elektrane prikazana je na slici 7.

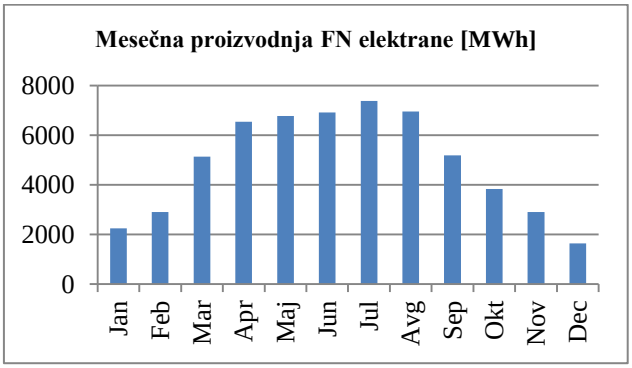
Razmatrajući predloženi koncept FN elektrane, može se zaključiti da je predloženo rešenje jako pouzdano, jer je čitava FN elektrana podeljena na male celine od 330 kW. Ispadom jednog invertora u ovom slučaju će odreagovati zaštita blok transformatora i isključiće transformator 10/0,4 kV/kV. Ovim ispadom se neće značajnije ugroziti ukupan rad FN elektrane, jer je ispala snaga predstavlja samo 0,7% instalisane snage.



Slika 7. Jednopolna šema blokaFN elektrane

5. GODIŠNJA PROIZVODNJA ELEKTRIČNE ENERGIJE FN ELEKTRANE PO MESECIMA

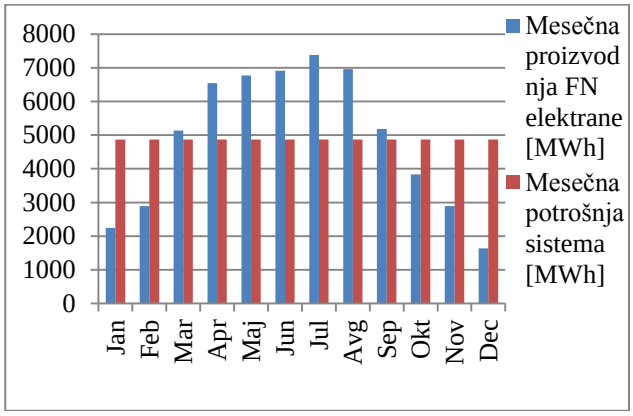
Prilikom procene isplativosti gradnje jedne FN elektrane uvek je važno pitanje da li će se investicija vratiti kroz naplatu proizvedene električne energije kroz određeni broj godina. U skladu sa procenjenom proizvodnjom električne energije za instalirani 1 kWp, na slici 8 prikazan je dijagram procene proizvodnje FN elektrane.



Slika 8. Mesečna proizvodnja FN elektrane.

Pošto se išlo na celokupnu kompenzaciju potrošnje električne energije sistema za odlaganje pepela i šljake, projektovana FN elektrana proizvodiće upravo onoliko energije koliko navedeni sistem u termoelektrani troši, tj. 58.377 MWh. Tokom letnjih meseci proizvodnja FN

elektrane je daleko veća od potrošnje sistema, dok je tokom zimskih meseci situacija obrnuta. Višak energije tokom letnjih meseci se injektira u mrežu, dok se tokom zime povlači iz mreže što je ilustrovano na slici 9.



Slika 9. Odnos između proizvodnje i potrošnje

6. TEHNO-EKONOMSKA ANALIZA

Prilikom planiranja izgradnje nekog elektroenergetskog objekta uvek se polazi od procene ekonomske isplativosti. Ekonomska isplativost se zasniva na periodu vraćanja investicije prilikom gradnje objekta. U sledećoj tabeli je izračunata okvirna cena gradnje jednog bloka FN elektrane.

Tabela 1. Okvirna specifikacija opreme za jedan blok

OPREMA:	**CENA (RSD):**
OPREMA FN SISTEMA	52.850.940
RAZVODNI DC ORMANI U POLJU	5.204.410
GLAVNI DC RAZVODNI ORMAN	693.426
AC RAZVODNI ORMAN	1.477.999
KABLOVI	2.615.652
PLC, SCADA i komunikacije	1.689.480
BLOK TRANSFORMATOR 0,4/10 kV/kV	3.330.000
Ukupno: 64.535.507 RSD	

Kako je snaga projektovane elektrane 46.701 kW i sastoji se iz 142 bloka, navedena cena iz prethodne tabele mora biti pomnožena sa 142. Na taj način dobija se prosečna cena gradnje od 1,64 €/W što je u granicama očekivanih vrednosti za velike FN elektrane. Prema uredbi o podsticajnim merama za proizvodnju električne energije iz obnovljivih izvora, podsticajna otkupna cena električne energije za velike solarne elektrane je 14,6 evrocenti [3]. Ovo je takozvana feed-in tarifa. Ukoliko se uvaži navedena cena i troškovi emisije CO_2 u atmosferu period vraćanja investicije bi bio 8,6 godina, što je sasvim korektno i u granicama je očekivanih vrednosti. Problem koji se ovde javlja je ukupna kvota koja iznosi 10 MW do 2020. godine i koja je do sada već popunjena. Nova strategija razvoja energetike Republike Srbije do 2025. godine sa projekcijama do 2030. godine predviđa višestruko povećanje kvota za solarne elektrane na 100 MW do 2025. godine, što može omogućiti isplativost ovakve investicije [4].

7. ZAKLJUČAK

Kao alternativa potrošnji električne energije iz sopstvene proizvodnje TE Kostolac B za rad elektromotornog pogona deponije pepela i šljake „Ćirkovac“ predložen je FN sistem. Sistem je zamišljen da bude postavljen na konstrukciju iznad deponije i da bude snage snage 46.701 kWp. Izvršen je kompletan proračun sistema i dato predviđanje njegove mesečne proizvodnje.
Predloženi FN sistem je kapaciteta da može da proizvede količinu električne energije, koja je neophodna za rad elektromotornog pogona. Međutim, u letnjim mesecima proizvodnja je veća od potreba, dok je u zimskim obrnuto, što je i očekivano. Tehno-ekonomska analiza je pokazala da se sistem može samo-isplatiti za 8,6 godina, ako se koristi feed-in tarifa i uvažavaju troškovi emisije CO_2. Međutim, ukoliko se ne bi uvažile feed-in tarife, period vraćanja investicije je jako dug i kreće se oko 20 godina što je svakako neisplativo.

8. LITERATURA

[1] Z. Milovanović, “*Termoenergetska postrojenja – teoretske osnove“,* Univerzitet u Banja Luci, Mašinski fakultet, Banja Luka, 2011.
[2] A. Stanković, *„Uticaj termoelektrana na kvalitet životne sredine - studija slučaja TA „Kostolac“*, Diplomski rad, Univerzitet u Beogradu, Fakultet bezbednosti, Beograd, 2017.
[3]https://bankwatch.org/wpcontent/uploads/2017/03/briefing-Balkans-CO2-29Mar2017-bih.pdf
[4] Ministarstvo rudarstva i energetike, “*Strategija razvoja energetike Republike Srbije do 2025. godine sa projekcijama do 2030. godine*“, Beograd 2016

Kratka biografija:

Zoran Nikolić rođen je u Prokuplju 1993. godine. Na Elektronskom fakultetu u Nišu diplomirao je 2016. godine na studijskom programu Elektroenergetika. Na Fakultetu tehničkih nauka u Novom Sadu upisao je master studije na studijskom programu Elektro-energetika – Elektroenergetski sistemi. Master rad na temu obnovljivih izvora električne energije odbranio je 2018. godine.

dr Vladimir A. Katić, red.prof. rođen je 1954. godine u Novom Sadu. Doktorirao je na Univerzitetu u Beogradu 1991. godine. Oblasti interesovanja su mu energetska elektronika, obnovljivi izvori električne energije, kvalitet električne energije i električna vozila.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE14Kupco

PODRŠKA VIZUALIZACIJI JEZIKA KREIRANIH UPOTREBOM TEXTX BIBLIOTEKE U OKVIRU VISUAL STUDIO CODE EDITORA

LANGUAGE VISUALIZATION SUPPORT FOR TEXTX-BASED LANGUAGES IN VISUAL STUDIO CODE EDITOR

Daniel Kupčo, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad predstavlja implementaciju podrške za vizualizaciju bilo kog DSL-a (eng. Domain Specific Language) kreiranog pomoću textX biblioteke. Realizovana je kao ekstenzija za Visual Studio Code editor u kombinaciji za posebno kreiranim DSL-om za opis vizualizacije pod nazivom viewX i različitim bibliotekama koje omogućavaju vizualizaciju modela.*

Ključne reči: *DSL, vizualizacija, textX, viewX, Visual Studio Code, ekstenzija*

Abstract – *This paper presents implementation of a support for the visualization of any textX-based DSL (Domain Specific Language). It is realized as an extension for Visual Studio Code editor in combination with specially created DSL for visualization description called viewX and a variety of libraries which enable model visualization.*

Keywords: *DSL, visualization, textX, viewX, Visual Studio Code, extension*

1. UVOD

U ovom radu predstavljena je implementacija ekstenzije za *Visual Studio Code* editor, čija je uloga da omogući vizualizaciju jezika specifičnih za domen, ili kraće DSL-ova (*eng. Domain Specific Languages*) kreiranih upotrebom *textX* biblioteke.

Problematika se ogleda u vizualizaciji struktura podataka tipa grafa, što predstavlja zasebnu oblast izučavanja u matematici i računarskim naukama. Pored toga, rešenje mora biti generičko, odnosno, potrebno je omogućiti vizualizaciju bilo kog modela jezika kreiranog pomoću *textX* biblioteke, i to na takav način da se korisniku pruža mogućnost modifikovanja načina vizualizacije i samog izgleda grafa kao njenog rezultata.

Značaj alata koji se bave problemom vizualizacije ogleda se u činjenici da korisnicima omogućavaju lakše i bolje razumevanje semantike napisanih modela, oslanjajući se na vizualne elemente i metode prikazivanja. Dodatno, vizualizacija programerima može da olakša uočavanje grešaka unutar modela čime se smanjuje vreme potrebno za njihovo ispravljanje. Vizualno prikazivanje takođe može da omogući i bolju komunikaciju između programera i domenskih eksperata jer je moguće predstaviti apstrakciju modela na takav način da se ključna svojstva i relacije koje su definisane unutar njega mogu jasno uočiti i interpretirati. Jedna od prednosti implementacije koja je predstavljena u ovom radu jeste mogućnost vizualizacije modela bilo kog jezika kreiranog pomoću *textX* biblioteke. Druga prednost koja se može izvdvojiti jeste izolacija načina vizualizacije od njenog sadržaja; sadržaj je deo samog modela koji se prikazuje, a način na koji se prikazuje definisan je *viewX* modelom što implicitno omogućava da se jedan model može prikazati na više različitih načina u zavisnosti od potrebe, a sa druge strane, više modela se mogu prikazati na isti način.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Igor Dejanović, vanr. prof.

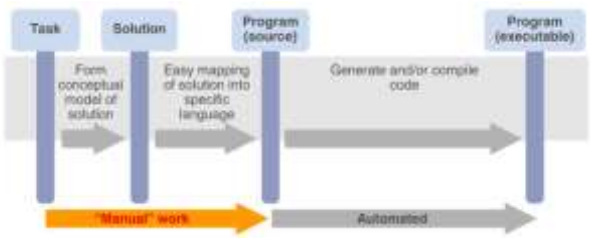
2. TEORIJSKE OSNOVE

2.1. Grafovi

Strukturu modela nekog jezika najpogodnije je predstaviti strukturom tipa grafa. Graf omogućava jednostavan prikaz koncepata definisanih u modelu i njihovih međusobnih relacija jer se koncepti mogu predstaviti čvorovima grafa, a njihove međusobne relacije kao veze između čvorova. Najveći izazov prilikom vizualizacije grafova predstavlja odabir pogodnog algoritma za raspoređivanje čvorova koji će zadovoljiti potrebne estetičke kriterijume.

2.2. Jezici specifični za domen

Jezici specifični za domen predstavljaju posebno kreirane jezike namenjene za upotrebu u okviru usko specifičnog domena. Konstruktivni elementi jezika predstavljeni su konceptima iz domena koji omogućavaju da konceptualan model rešenja bude direktno preslikan na konkretan model DSL-a koristeći koncepte iz domena problema.



Slika 1. *Proces razvoja rešenja upotrebom DSL-a [1]*

Za razliku od modela napisanih u jezicima opšte namene, DSL modeli su jednostavniji za razumevanje, što doprinosi boljoj komunikaciji između programera i

domenskih eksperata. čime se značajno umanjuje vreme potrebno za razvoj rešenja.

3. KORIŠĆENE TEHNOLOGIJE I ALATI

3.1. Python, textX i Arpeggio

Deo ekstenzije koji se oslanja na upotrebu *textX* biblioteke realizovan je u *Python* okruženju. *TextX* predstavlja alat koji omogućava kreiranje jezika specifičnih za domen [2]. Pomoću *textX* meta-jezika definiše se gramatika jezika na osnovu koje se konstruiše *Arpeggio [3]* parser i meta-model koji, u suštini, predstavljaju kreiran DSL spreman za korišćenje.

3.2. JavaScript, TypeScript i Visual Studio Code

Visual Studio Code editor, kao i svi moderni editori izvornog koda, imaju modularnu arhitekturu i podržavaju koncept proširivosti putem ekstenzija [4]. Jezgro *ovog* editora se zasniva na *JavaScript* okruženju, odnosno pokreće ga *Node.js* server. Stoga, *Visual Studio Code* editor propisuje svoj interfejs proširivosti putem pisanja *TypeScript/JavaScript* ekstenzija.

4. IMPLEMENTACIJA

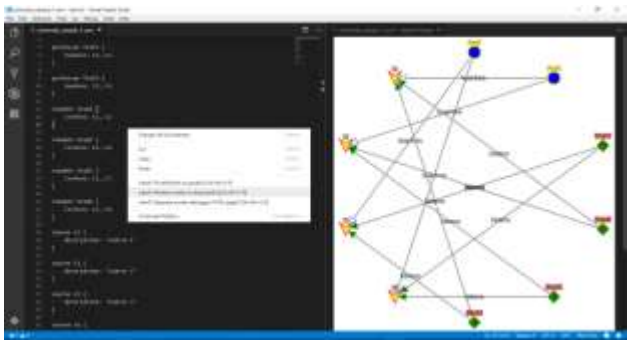
Implementacija ovog rada je realizovana kao ekstenzija za *Visual Studio Code* editor u kombinaciji sa posebno razvijenim *DSL*-om za opisivanje vizualizacije razvijenim pomoću *textX* biblioteke pod nazivom *viewX*. Iako je osnova ovog rešenja *Visual Studio Code* ekstenzija ono zapravo predstavlja integraciju nekoliko alata i biblioteka iz više okruženja.

4.1. viewX DSL

Sastavni deo ovog rada jeste i *viewX*, jezik kreiran upotrebom *textX* biblioteke. Njegova uloga je da opiše vizualizaciju modela nekog drugog *textX* baziranog DSL-a.
Kako bi se obezbedila što kompletnija kontrola vizualizacije, gramatika *ViewX* jezika kreirana je s ciljem da, pored osnovnih vizuelnih karakteristika elemenata modela, pruži i mogućnost definisanja izgleda grafa kao celine putem odabira različitih algoritama za raspoređivanje elemenata grafa.

4.2. viewX ekstenzija

Editor je pomoću ekstenzije proširen u vidu komandi i opcija putem kontekstnog menija kojima je omogućeno kreiranje novih projekata i interakcija sa ekstenzijom i mehanizmima vizualizacije unutar editora (Slika 1).



Slika 2. *Primer vizualizacije modela*

Bitan deo ekstenzije predstavlja i logika koja se bavi interpretacijom *viewX* modela koji opisuje elemente vizualizacije. Kako se ovaj rad bavi vizualizacijom modela jezika kreiranih pomoću *textX* biblioteke, ovaj deo ekstenzije je realizovan pomoću *Python* jezika jer je i sama *textX* biblioteka realizovana u okviru *Python* okruženja. Ovaj deo je ključan u procesu vizualizacije jer predstavlja vezu između konkretnog modela nekog jezika i grafa koji predstavlja rezultat vizualizacije. Ovaj graf se konstruiše tako što se najpre interpretira konkretan model jezika koji želimo da vizualizujemo a potom i *viewX* model. Kombinacijom ove dve interpretacije i na osnovu prikupljenih informacija iz oba modela kreira se model grafa koji predstavlja apstrakciju konkretnog modela koji se vizualizuje a koji odgovara parametrima definisanim u okviru *viewX* modela.

Polazna tačka i osnova u kojoj je smeštena logika za interpretaciju modela i konstruisanje modela grafa jeste *ViewXInterpreter.py* klasa. Uloga ove klase je ključna i ona obuhvata sve od interpretacije oba modela putem algoritama za prolazak kroz stabla objekata modela dobijenih nakon interpretacije, preko konstruisanja modela grafa sa svim informacijama vezanim za vizualna svojstva elemenata grafa, pa sve do generisanja modela grafa i propratnog koda u vidu HTML fajla, čijim će se zapravo učitavanjem graf vizualizovati. Iako je većina logike koncentrisana u ovoj klasi, stvari poput generisanja fajlova, pomoćnih metoda i adaptacije modela sa ciljnom bibliotekom za vizualizaciju su ipak smeštene u drugim *Python* skriptama zarad bolje organizacije koda.

4.3. Python-shell

Kako se ekstenzija izvršava u *JavaScript* okruženju, potrebno je nekako povezati ekstenziju sa kodom napisanim u *Python*-u. U ovu svrhu se koristi *python-shell* biblioteka [5]. Ova biblioteka omogućava izvršavanje *Python* skripti unutar novog procesa kreiranog u okviru *JavaScript* koda čime se omogućava komunikacija između ova dva okruženja. Ova komunikacija se zasniva na slanju podataka putem standardnih ulazno-izlaznih tokova podataka. Pozivanje koda unutar *Python* skripti putem ove biblioteke je vrlo jednostavno (Slika 2) a u slučaju nastalih grešaka u kodu koji se poziva podržan je i mehanizam za upravljanje izuzecima.

4.4. Jinja2

Kao što je već pomenuto, rezultat interpretacije modela jeste generisanje HTML fajla, čijim učitavanjem započinje vizualizacija modela, a koji sadrži opis strukture modela grafa zajedno sa *CSS* i *JavaScript* kodom koji obezbeđuju željeno stilizovanje i interakciju grafa. Ovaj kod adaptiran je, i obezbeđuje integraciju sa ciljnom bibliotekom za vizualizaciju grafa.

Generisanje ovog fajla se odvija korišćenjem *Jinja2* biblioteke. *Jinja2* predstavlja *Python* modul koji omogućava generisanje fajlova korišćenjem šablona (*eng. template*). Šablon omogućava generisanje fajlova tako što se dinamički sadržaj razdvaja od statičkog sadržaja blokovima koda koji će se potom generisati na osnovu prosleđenih *Python* objekata.

4.5. Cytoscape.js

Alat čija je uloga vizualizacija grafa jeste *Cytoscape.js* biblioteka. Ovo je *JavaScript* biblioteka zasnovana na teoriji grafova i namenjena je za analizu i vizualizaciju grafova. Postoji nekoliko biblioteka koje se bave vizualizacijom grafova i svaka je specifična na svoj način. Neke od osobina koje izdvajaju *Cytoscape.js* u od ostalih alata jesu mogućnost prilagođavanja putem mnogobrojnih parametara kao što su zumiranje, kretanje po grafu, raspored iscrtavanja čvorova grafa itd., zatim stilizovanja izgleda čvorova grafa i njihovih veza, kreiranje kompozitnih čvorova (čvorovi mogu sadržavati druge čvorove), potom izvršavanje različitih algoritama nad elementima grafa, animacije, mogućnost čuvanja različitih informacija unutar elemenata grafa kao i korišćenje tih informacija za različite potrebe.

Pored navedenih osobina, *Cytoscape.js* biblioteka podržava i proširivanje putem sopstvenih ekstenzija. Korisnik može, putem definisanog interfejsa, da proširi skup podržanih funkcionalnosti biblioteke u vidu dodatnih algoritama za raspoređivanje elemenata grafa, novih elemenata korisničkog interfejsa kao i funkcionalnih elemenata u vidu *clipboard* i *undo-redo* mehanizama. Takođe, podržan je i mehanizam reagovanja na događaje od strane korisnika. Određene akcije korisnika emituju događaje na koje je moguće registrovati metode koje će se pozvati kao reakcija na ove događaje. Ulazni parametar ovih metoda predstavlja objekat koji nosi osnovne informacije o nastalom događaju poput vremena nastanka, tip događaja, koordinate, element grafa ukoliko je nad njim načinjen itd. Implementacijom ovih metoda zapravo definišemo ponašanje kojim želimo da reagujemo na nastanak ovih događaja.

4.6. Socket.io

Budući da postoji mehanizam reagovanja na događaje i u okviru editora i na nivou grafa, postavlja se pitanje kako ove događaje propagirati sa jedne na drugu stranu i reagovati na njih, imajući u vidu da se ekstenzija i vizualizacija grafa izvršavaju unutar dva odvojena procesa. Odgovor na ovo pitanje donosi *Socket.io* biblioteka.

Socket.io je *JavaScript* biblioteka koja omogućava bidirekcionu komunikaciju u realnom vremenu između dva procesa oslanjajući se na mehanizam utičnica (*eng. socket*). Ova komunikacija je realizovana implementacijom *publish and subscribe* šablona nad klijent-server arhitekturom.

Navedena arhitektura je u implementaciji ovog rada primenjena tako da sama ekstenzija u okviru editora, kao i prozori unutar kojih se učitava generisani HTML fajl, a samim tim i vizualizuje graf koji je u njemu definisan, imaju ulogu klijenata. Svaki klijent, u zavisnosti od njegove uloge da li je to ekstenzija ili klijent u kome se vizualizuje graf, je registrovan na određene događaje a istovremeno, svaki od njih šalje određene događaje koji nastanu u njemu ostalim klijentima. Ovi događaji se prosleđuju *Socket.io* serveru koji ih potom prosleđuje svim drugim klijentima istovremeno. Klijenti sve vreme slušaju na događaje i, ukoliko se desi neki od njih na koji su oni registrovani, poziva se metoda koja je registrovana kao reakcija na te događaje. Ovim je omogućena obostrana komunikacija između ekstenzije i klijenata unutar kojih se vrši vizualizacija grafa.

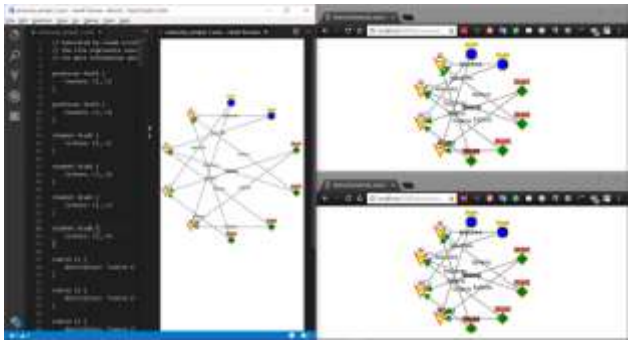
4.7. Browsersync

Svrha vizualizacije je da u svakom trenutku oslikava stvarno stanje modela pa se, iz ovog razloga, nakon sačuvanih izmena unutar modela iznova generiše HTML fajl sa novim modelom grafa koji reflektuje novonastale izmene. Problem koji se nameće jeste kako obavestiti sve klijente u kojima se graf vizualizuje da je došlo do promena i da se njihov prikaz treba osvežiti. U ovu svrhu koristi se *Browsersync*.

Browsersync je *JavaScript* biblioteka koja omogućava pristup statičkim fajlovima pute jedinstvene IP adrese. Svi klijenti koji vizualizuju graf, pristupaju fajlovima koji omogućavaju prikazivanje njegovog sadržaja pristupajući IP adresi putem *Browsersync* servera. *Browsersync* poseduje informaciju o svim klijentima koji su mu pristupili i, kada dode do izmene sadržaja ovih fajlova, *Browsersync* server javlja svakom klijentu da treba da osveži učitani sadržaj i svoje prikazivanje. Na ovaj način je omogućeno da svaki od aktivnih klijenata u svakom trenutku vizualizuje graf koji predstavlja verodostojni prikaz aktuelnog stanja modela.

4.8. Demonstracija ekstenzije

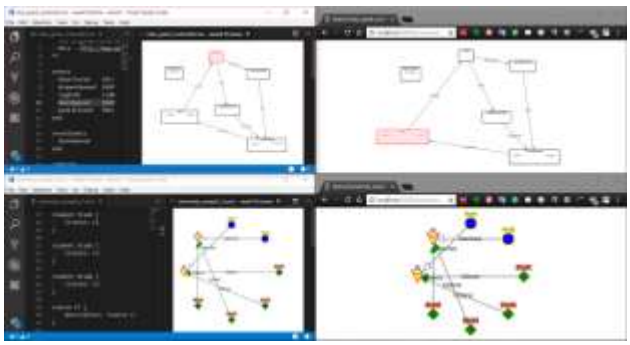
Iako je osnovni cilj ovog rada vizualizacija modela u okviru *Visual Studio Code* editora, ekstenzija poseduje nekoliko osobina i funkcionalnosti koje je izdvajaju od ostalih sličnih alata. Jedna od takvih funkcionalnosti je i mogućnost višestruke vizualizacije (Slika 3).



Slika 3. *Primer višestruke vizualizacije istog modela*

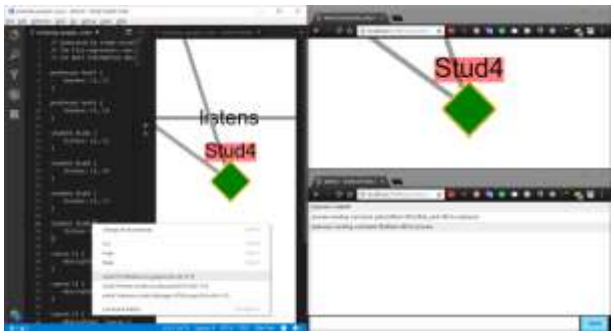
Svaka od ovih vizualizacija, bilo u okviru editora ili internet pretraživača, je potpuno nezavisna. Time je omogućeno da više korisnika istovremeno vizualizuju model i vrše interakciju nad grafom a da pri tome ne utiču na vizualizacije drugih korisnika.

Pored višestruke vizualizacije istog modela, moguće je i pokrenuti više instanci *Visual Studio Code* editora od kojih svaka može da istovremeno vizualizuje svoj model (Slika 4). Ovim je omogućena višestruka vizualizacija više modela što može da bude vrlo korisno kada je potrebno istovremeno analizirati nekoliko modela ako npr. želimo da uporedimo dva modela ili da prikažemo isti model na dva različita načina.



Slika 4. Višestruke *viewX* instance u paraleli

Interakcija sa grafom je ključna tokom analize i njegove interpretacije zarad boljeg razumevanja modela. Interakcija inicira događaje na koje ekstenzija reaguje, čime se uspotavlja komunikacija oslanjajući se na *Socket.io* server. Ukoliko se nešto ne dešava onako kako se očekuje, ne postoji mogućnost da se otkrije u kom momentu je došlo do problema u komunikaciji između delova ekstenzije.
Iz ovog razloga je vizualizacija dodatno podržana i sa mogućnošću otkrivanja grešaka, odnosno *debug* sesije (Slika 5). Ukoliko se *debug* opcija omogući za neku *viewX* instancu, moguće je otvoriti posebno kreiranu stranicu u okviru internet pretraživača u kome je u realnom vremenu moguće pratiti svaku poruku koja se šalje između klijenata i servera *Socket.io* sistema u oba smera.



Slika 5. *Demonstracija debug sesije*

Debug prozor, koji takođe predstavlja klijenta u okviru *Socket.io* klijent-server arhitekture, se nakon aktivacije i njegove konekcije sa *Socket.io* serverom registruje na sve poruke koje stižu do i odlaze od njega, te se njihov sadržaj prikazuje u glavnom delu prozora u hronološkom redosledu.

5. ZAKLJUČAK

Glavni cilj ovog rada bio je da se korisniku omogući vizualizacija modela nekog JSD-a kreiranog pomoću *textX* biblioteke ali na takav način da se nadomeste nedostaci sličnih alata za vizualizaciju modela koji su trenutno dostupni za upotrebu. Ekstenzija je od starta implementirana tako da omogućava generičnost, konfiguraciju i eventualna proširenja od strane korisnika kako bi odgovorila na njegove zahteve.

Jedna od najznačajnijih osobina *viewX* ekstenzije koja se izdvaja među ostalim rešenjima jeste generičnost. Može se posmatrati i kao nedostatak jer je za primenu potrebno da korisnik uloži više truda kako bi se generičnost konkretizovala, što smanjuje jednostavnost i brzinu upotrebe. Međutim, rezultat i sloboda dobijeni tom generičnošću nadmašuju uloženi napor.

Za dalji razvoj, u prvom redu, neophodno je obezbediti bolju integraciju sa *Cytoscape.js* bibliotekom i obogaćivanje *viewX* gramatike, čime će se obezbediti bolja kontrola nad generisanjem grafa i njegovom vizualizacijom, kao i mogućnost kasnije konfiguracije. Time bi se mogla omogućiti implementacija različitih funkcionalnosti poput boljeg raspoređivanja elemenata grafa, podršku za animaciju tokom vizualizacije, prikazivanje kompozitnih grafova (grafovi unutar grafova), reagovanje na različite događaje i još mnogo toga.

6. LITERATURA

[1] Sergey Dmitriev, JetBrains onBoard,“Language oriented programming: The next programming paradigm”, (2004)
[2] Dejanović Igor, Vaderna Renata, Milosavljević Gordana, Vuković Željko, “TextX: A Python tool for Domain-Specific Languages implementation”, Knowledge-Based Systems, vol. 115, 1-4, 2017.
[3] Igor Dejanović, Gordana Milosavljević, Renata Vaderna, „Arpeggio: A flexible PEG parser for Python“, Knowledge-Based Systems, vol. 95, 71-74, 2015
[4] Microsoft, “Extending Visual Studio Code”, Visual Studio Code, https://code.visualstudio.com/docs/extensions/overview (datum pristupa: 02.08.2018.)
[5] “Python-Shell”, GitHub, https://github.com/extrabacon/python-shell (datum pristupa: 02.08.2018.)

Kratka biografija:

Daniel Kupčo rođen je u Novom Sadu 1992. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Softversko inženjerstvo, odbranio je 2018.god. kontakt: kupcodanex@outlook.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE15Elero

SERVER I EKSTENZIJA ZA VS CODE OKRUŽENJE ZA PODRŠKU JEZICIMA BAZIRANIM NA TEXTX ALATU

LANGUAGE SERVER AND VS CODE EXTENSION FOR DOMAIN SPECIFIC LANGUAGES BASED ON TEXTX

Daniel Elero, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu implementirana je ekstenzija za Visual Studio Code okruženje koja jezicima baziranim na alatu textX omogućava: prijavljivanje grešaka, dopunu koda, skok na definiciju pravila, prikaz svih referenci i code lens. Ekstenziju čine dve komponente: server i klijent, a njihova komunikacija odvija se preko Language Server Protocol-a.*

Ključne reči: *ekstenzija, textX, DSL, Language Server Protocol, Visual Studio Code, dopuna koda, prijavljivanje grešaka, navigacija do definicije pravila, prikaz svih referenci*

Abstract – *This paper presents implementation of a Visual Studio Code extension which provides linting, code completion, go to reference, find all references and code lens for domain specific languages based on textX. Extension has two components: language server and client which uses Language Server Protocol for their communication.*

Keywords: *extension, textX, DSL, Language Server Protocol, Visual Studio Code, code completion, linting, go-to definition, find all references*

1. UVOD

Ovaj rad obuhvata dizajn i implementaciju ekstenzije za VS Code okruženje koja jezicima definisanim u alatu textX nudi prijavljivanje grešaka, dopunu koda, „skok“ na definiciju pravila, prikaz svih referenci pravila i *code lens*. Ekstenzija se sastoji iz dve komponente: servera i klijenta koji podržavaju i komuniciraju preko *Language Server Protocol*-a (LSP). Ideja LSP-a jeste razdvajanje implementacione logike za pružanje „pametnih“ funkcionalnosti od njihovog prikazivanja i upotrebe u nekom od editora. Na ovaj način, svi editori koji podržavaju LSP (VS Code, Atom, Vim i dr.), vrlo lako se mogu integrisati sa serverom.

Jezici specifični za domen (DSL – *Domain Specific Language*) predstavljaju programske jezike ograničene ekspresivnosti, fokusirane na određenu oblast [1]. DSL-ovi imaju brojne prednosti u odnosu na jezike opšte namene:

- **Povećana produktivnost** programera, rezultat je velike ekspresivnosti i konciznosti jezika specifičnog za domen. Postoje studije koje pokazuju da produktivnost može narasti i 1000%
- **Kvalitetniji izvorni kod** rezultat je smanjenja broja linija programskog koda potrebnih da se reši problem.
- **Duži životni vek aplikacije** ogleda se u tome da je rešenje na višem nivou apstrakcije, odnosno, implementacija rešenja ne zavisi od izbora platforme i tehnologija
- **Angažman domenskih eksperata** omogućen je dizajnom jezika koji isključivo sadrži koncepte iz oblasti za koju je namenjen. Domenski eksperti često samostalno mogu da koriste ove tipove jezika.
- **Samodokumentujući** - zbog upotrebe koncepata iz domena

Martin Fovler u [1], deli DSL-ove u 3 kategorije: **eksterni**, **interni** i **jezičke radionice**.

Interni DSL nastaje proširivanjem jezika opšte namene, najčešće kreiranjem biblioteka. Ovakav pristup omogućava jednostavniju implementaciju i korišćenje integrisanih razvojnih okruženja (IDE) jezika na kome je zasnovan, ali na uštrb sintaksne ograničenosti.

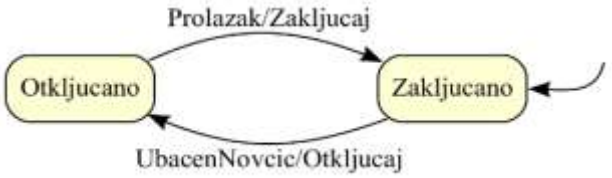
Eksterni DSL ima posebno definisanu sintaksu, kao i svoj interpreter ili kompajler. Semantika ovih tipova jezika dobija se prevođenjem na neki od jezika opšte namene ili njihovom interpretacijom.

Fovler u [2] uvodi termin jezičke radionice koja predstavlja okruženje za razvoj, testiranje i evoluciju jezika i propratnih alata. Najpoznatiji pripadnici ove grupe su Xtext i MPS.

Nezavisno od kategorije, svaki jezik specifičan za domen sastoji se od jedne ili više **konkretnih sintaksi**, **apstraktne sintakse** i **semantike.**
Konkretna sintaksa predstavlja vizuelnu reprezentaciju jezika preko koje korisnik vrši interakciju sa računarom i definiše izgled iskaza na nekom jeziku [3]. Ona može biti tekstualna, grafička, tabelarna, hibridna (kombinacija više vrsta), itd. Pri parsiranju tekstualne sintakse, kreira se stablo konkretne sintakse (eng. *Concrete Syntax Tree* - CST). Na slici 1 prikazane su tekstualna i grafička sintaksa koje imaju isto značenje.
Apstraktnu sintaksu čine koncepti domena, njihove osobine i korelacije. Ona predstavlja metamodel i služi za validaciju iskaza. Stablo apstraktne sintakse (eng.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Igor Dejanović, vanr. prof.

Abstract Syntax Tree - AST), takođe, nastaje parsiranjem ulaznog teksta, ali sadrži manje informacija od CST.




```
machine Kapija {
    state Zakljucano:initial, Otkljucano;
    event UbacenNovcic, Prolazak;
    action Otkljucaj, Zakljucaj;

    transition
        Zakljucano (UbacenNovcic/Otkljucaj) Otkljucano,
        Otkljucano (Prolazak/Zakljucaj) Zakljucano;
}
```

Slika 1 Grafička i tekstualna konkretna sintaksa [4]

Apstraktni semantički graf (*Abstract Semantic Graph* - *ASG*) predstavlja (uglavnom) direktan acikličan graf i na višem je nivou apstrakcije od AST.
Semantika definiše smisao jezičkih iskaza [4]. Takođe, semantiku jezika definiše skup ograničenja.

S obzirom da je tema rada usko vezana za alat textX, fokus rada usmeren je ka parserima i eksternim DSL-ovima sa tekstualnom konkretnom sintaksom.
Parseri su softverske komponente koje od ulaznog niza znakova i simbola, kreiraju strukture pogodne za dalju obradu (najčešće CST ili AST), a na osnovu formalno definisane gramatike. Za izdvajanje tokena iz ulaznog teksta, najčešće se koristi odvojen leksički analizator, ali postoje i parseri koji vrše tokenizaciju i parsiranje u jednom koraku (eng. *scannerless parsers*). Parseri mogu biti automatski ili polu-automatski generisani parser generatorima (eng. *parser generators*) [5].
Parsiranje se najčešće odvija u tri koraka [5]:

1. **Leksička analiza** vrši kreiranje tokena iz ulaznog teksta.
2. **Sintaksna analiza** vrši validaciju i proveru redosleda tokena u zavisnosti od definisane gramatike i kreira stablo parsiranja.
3. **Semantička analiza** daje značenje parsiranom ulazu, interpretirajući ga ili prevodeći na jezik razumljiv računaru. U ovoj fazi se, nad čvorovima stabla parsiranja, izvršavaju semantičke akcije.

Prema načinu parsiranja, Aho u [6] deli parsere na dve grupe:

- *top-down*
- *bottom-up*

Bottom-up **parseriranje** počinje identifikacijom terminalnih simbola analizirajući ulazne tokene sleva na desno. Terminali ujedno postaju i listovi stabla parsiranja. Nakon toga se, kombinacijom terminala, a u skladu sa produkcionim pravilima gramatike, grade čvorovi stabla koji predstavljaju neterminale. Ovaj proces se ponavlja dok se ne stigne do korenskog pravila (korena stabla parsiranja).

Top-down **parseri** kreću od korena stabla i kreiraju neterminale (čvorove) primenjujući redukciona pravila gramatike. Parsiranje traje sve dok se svi neterminali ne zamene terminalima (listovi stabla).

Metamodel čini specifikacija koncepata, korelacija i ograničenja potrebnih za konstrukciju modela. Proces kreiranja metamodela naziva se metamodelovanje [7]. Metamodeli su specificirani za određeni domen, odnosno služe za kreiranje modela za navedeni domen. Ako se podignemo za još jedan nivo apstrakcije, potreban nam je formalizam za opis metamodela, a to je meta-metamodel. Domen za koji je meta-metamodel namenjen jeste kreiranje metamodela.

2. PREGLED STANJA IZ OBLASTI

2.1 Arpeggio

Arpeggio [8] predstavlja opadajući rekurzivni parser sa *backtracking*-om i memoizacijom (tzv. *pacrat parser*). Razvijen je u programskom jeziku Python od strane profesora Igora Dejanovića na Fakultetu Tehničkih Nauka u Novom Sadu. Gramatika jezika zadaje se PEG notacijom (slika 2) ili Python funkcijama (slika 3), a, za razliku od parser generatora, Arpeggio se konfiguriše „u letu".

```
robot = 'begin' command* 'end' EOF
command = UP/DOWN/LEFT/RIGHT
UP = 'up'
DOWN = 'down'
LEFT = 'left'
RIGHT = 'right'
```

Slika 2 Primer gramatike definisane PEG notacijom

```
def robot():    return 'begin', ZeroOrMore(command), 'end', EOF
def command():  return [UP, DOWN, LEFT, RIGHT]
def UP():       return 'up'
def DOWN():     return 'down'
def LEFT():     return 'left'
def RIGHT():    return 'right'
```

Slika 3 Primer gramatike definisane Python funckijama

Parsiranjem odgovarajućeg modela dobijamo AST i stablo parsiranja nad kojim je moguće definisati semantičke akcije. Definisanje semantičkih akcija vrši se nasleđivanjem klase ***PTNodeVisitor***, u okviru koje se navode metode oblika ***visit_*** iza čega sledi ime pravila. Pored samog parsiranja, Arpeggio ima podršku za debagovanje i vizualizaciju modela i stabla parsiranja.

2.2 Xtext

XText je razvojni okvir namenjen za kreiranje tekstualnih DSL-ova. Za opis gramatika koristi se metajezik Xtext, koji sadrži predefinisane tipove, anotacije, enumeracije, a dozvoljeno je i kombinovanje više gramatika. Pored opisa gramatike, ovaj razvojni okvir omogućava dodavanje različitih semantičkih validacija, prilagođavanje i modifikacije prikaza strukture koda, dopune koda i ostalih karakteristika editora kao i pisanje generatora za mapiranje modela na, najčešće, jezike opšte namene.

XText metajezik je poslužio kao inspiracija za kreiranje alata textX, a skup alata vezanih za generisanje i funkcionisanje *Eclipse* editora kreirao je ideju o proširenju textX-a u istom pravcu, što je istraženo i implementirano kroz ovaj master rad.

2.3 textX

TextX [9] je alat namenjen brzom kreiranju eksternih jezika specifičnih za domen. Implementiran je u programskom jeziku Python na Fakultetu tehničkih nauka u Novom Sadu od strane profesora Igora Dejanovića, a moguće ga je instalirati pip komandom na sledeći način: *pip install textX*. TextX ujedno predstavlja i metajezik čija je notacija veoma slična XText-ovoj, dok u pozadini koristi Arpeggio parser.

Za razliku od XText-a koji je poslužio kao inspiracija za kreiranje ovog alata, a zahvaljujući dinamičkoj prirodi jezika Python, textX ima sposobnost da "u letu" konfiguriše Arpeggio, kreira metamodel i parsira tekstualne modele koji odgovaraju specificiranoj gramatici.

Metamodel čine Python klase koje odgovaraju pravilima gramatike i sadrže njihove relevantne informacije, a proces parsiranja ulaznog modela kreira objektni graf čiji čvorovi su instance klasa metamodela.

Kreirani objektni graf je dalje moguće interpretirati, ili iz njega generisati izvorni kod na nekom od jezika opšte namene.

Čuvajući glavnu karakteristiku textX-a, ideja ovog master rada je da "u letu", na osnovu gramatike i dodatnih specifikacija, prilagodi jezički server DSL-u.

2.4 *Language server protocol*

Veliki broj IDE-a i tekstualnih editora za upravljanje izvornim kodom, sa jedne strane, i sve veći broj programskih jezika i njihov aktivni razvoj, sa druge strane, povećava resurse potrebne za razvoj i održavanje alata za „pametnu" podršku tim jezicima. LSP je baziran na JSON RPC (*Remote Procedure Call*) protokolu i specificira skup metoda i notifikacija za komunikaciju između servera za podršku određenom jeziku i različitih klijenata (IDE-a) i time rešava navedeni problem.

3. IMPLEMENTACIJA

Ekstenziju čine server i klijent. Implementacija klijenta je trivijalna i sadrži samo logiku potrebnu za pokretanje i uspostavljanje komunikacije sa serverom ukoliko se desi jedan od sledećih događaja:

- Kreiranje ***.txconfig*** fajla u okviru radnog prostora
- Otvaranje fajlova sa ekstenzijom ***.tx***

Fajl ***.txconfig*** (slika 4) predstavlja konfiguracioni fajl koji sadrži informacije kao što su: ekstenzije fajlova ciljanog DSL-a, putanja do gramatike jezika, putanje do ostalih konfiguracionih fajlova i dr.

Nakon konfiguracije i aktivacije, server je spreman za korišćenje.

```
dsl MyDSL [mydsl, my] {
    general {
        author: "Daniel Elero"
        version: "1.0.0"
    }
    paths {
        grammar: "./mydsl/grammar.tx"
        classes: "./mydsl/lang.py:get_classes"
        builtins: "./mydsl/lang.py:get_builtins"
        model_processors: "./mydsl/lang.py:get_model_proc"
        object_processors: "./mydsl/lang.py:get_obj_proc"
        match_filters: "./mydsl/lang.py:get_filters"
    }
}
```

Slika 4 Konfiguracioni fajl (***.txconfig***)

3.1 „Skok" na definiciju pravila

"Skok" na definiciju pravila (eng. ***go to definition***) podrazumeva da se preko imena instance pravila, "skoči" na njegovu definiciju u okviru istog ili drugog fajla. Ova funkcionalnost nam omogućava brzo kretanje po izvornom kodu u toku razvoja, ili za vreme pregleda koda. Implementacija opisane funkcionalnosti zahtevala je i proširenje alata textX kome je dodato polje za čuvanje i lak pristup svim pravilima na osnovu njihove pozicije. Korišćenjem binarne pretrage, algoritam na osnovu trenutne pozicije kursora u editoru, pretražuje pravila i, ukoliko je pravilo pronađeno, vraća fajl i poziciju na kojoj se definicija nalazi.

3.2 Pretraživanje referenci

Pretraživanje i pregled referenci (eng. ***find all references***) predstavlja inverznu funkcionalnost u odnosu na „skok" do definicije pravila, a pruža nam uvid koliko puta i gde se instanca nekog pravila koristila u modelu. Funkcija koja vrši pretragu svih referenci pravila sadrži sledeće korake:

1. Za trenutnu poziciju kursora u editoru, dobavlja se instanca pravila
2. Iz liste svih referenciranih pravila u modelu, biraju se ona pravila koja se poklapaju sa pravilom dobijenim u koraku 1
3. Dobijena lista se mapira na objekte koji odgovaraju LSP specifikaciji

3.3 Prijavljivanje grešaka

Prijavljivanje grešaka tokom pisanja koda predstavlja jednu od najznačajnijih funkcionalnosti ekstenzije. Implementacija se oslanja na alate textX i Arpeggio i njihove mehanizme hvatanja izuzetaka. Na svaku promenu teksta ili fajla, server proverava trenutno stanje modela na osnovu zadate gramatike i putem notifikacija obaveštava klijenta o greškama.

3.4 Dopuna koda

Dopuna koda (eng. ***code completion***) je funkcionalnost koju klijent poziva da bi, na osnovu trenutne pozicije kursora u tekstu, dobio predlog o potencijalnim tokenima koji mogu da se nađu na tom mestu. Trenutna implementacija oslanja se na textX *i* Arpeggio, odnosno, za listu stavki dopune koda, predlažu se tokeni koje parser zahteva na datom mestu.

Algoritam za kreiranje liste predloga izvršava se u sledećim koracima:

1. Klijent šalje zahtev za dopunu koda
2. Server na poziciji kursora ubacuje karaktere koji će model učiniti nevalidnim
3. textX preko Arpeggio parsera dobija moguće validne tokene na mestu greške
4. U zavisnosti od toga da li je greška sintaksna ili semantička, textX dopunjuje informacije o grešci
5. Greška sa listom predloga se vraća do servera
6. Server filtrira i proširuje listu predloga i vraća ih klijentu
7. Predlozi se prikazuju u okviru korisničkog interfejsa

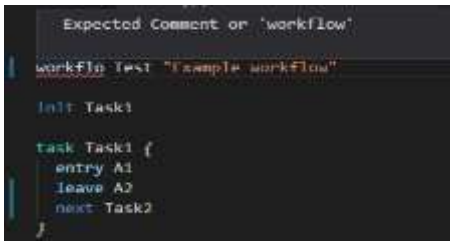
3.5 Generisanje ekstenzije

Generisanje ekstenzije predstavlja funkcionalnost koja na osnovu konfiguracionih fajlova vrši kreiranje i pakovanje ekstenzije specifične za određeni domenski jezik i omogućava njihovu distribuciju.

4. VERIFIKACIJA I UPOTREBA EKSTENZIJE

Za verifikaciju ekstenzije odabran je jezik *Workflow* čija je namena definisanje aktivnosti, akcija, i zadataka za opis proizvoljnog posla. Nakon kreiranja gramatike i konfiguracionog fajla, izgenerisana je namenska ekstenzija sa sledećim funkcionalnostima:

Prijavljivanje grešaka (slika 6), predstavljeno je podvlačenjem mesta na kome se nalazi greška.

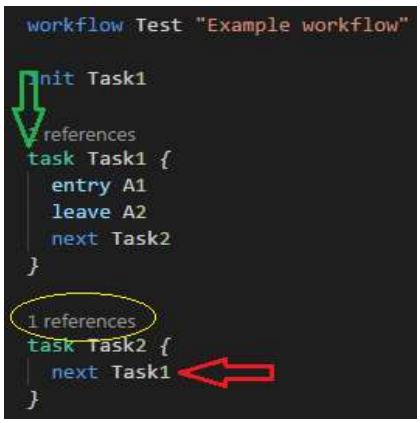


Slika 6 Prikaz sintaksne greške

Dopuna koda (slika 7) predstavlja prikaz validnih tokena na trenutnoj poziciji kursora.

Slika 7 Prikaz dopune koda

„Skok“ na definiciju pravila i **prikaz svih referenci** povezane su funkcionalnosti i prikazane su na slici 8. Kombinacijom tastera CTRL i pritiskom levog klika miša na *Task1* (crvena strelica), kursor se prebacuje na početak njegove definicije (zelena strelica), dok žuta elipsa naznačuje koliko je puta neko pravilo referencirano u modelu.



Slika 8 „Skok“ na definiciju pravila i prikaz svih referenci

5. ZAKLJUČAK

Cilj ovog master rada bio je istraživanje postojećih rešenja iz oblasti parsera, metajezika, namenskih IDE-a i implementacija generičkog jezičkog servera za podršku jezicima definisanim u alatu textX. Karakteristika implementiranog jezičkog servera da se dinamički prilagođava prema metamodelu jezika i dodatnim specifikacionim fajlovima čuva prednosti textX-a u odnosu na konkurentske alate.

Implementirana ekstenzija pruža prijavljivanje grešaka, dopunu koda, „skok“ na definiciju pravila, pretraživanje referenci pravila i *code lens*.

LITERATURA

[1] M. Fowler, “Domain specific languages.”, Addison-Wesley Professional, September 24, 2010.

[2] M. Fowler, “Language Workbenches: The Killer-App for Domain Specific Languages?” Dostupno na http://www.issi.uned.es/doctorado/generative/Bibliografia/Fowler.pdf. Pristupano: Avgust 2018.

[3] M. Völter, “DSL Engineering”, 2010-2013.

[4] http://www.igordejanovic.net/courses/jsd/uvod.html. Pristupano: Avgust 2018.

[5] https://en.wikipedia.org/wiki/Parsing. Pristupano: Avgust 2018.

[6] A. V. Aho, M. S. Lam , R. Sethi, J. D. Ullman, “Compilers : principles, techniques, and tools” - 2nd ed.Pearson Education, 2007

[7] I. Dejanović, “Prilog metodama brzog razvoja softvera na bazi proširivih jezičkih specifikacija. PhD thesis, Faculty of Technical Sciences, University of Novi Sad, January 2012”

[8] I. Dejanović, B. Perišić, G. Milosavljević, “ARPEGGIO: Pacrat parser interpreter”, Fakultet tehničkih nauka, Univerzitet u Novom Sadu, 2010.

[9] I. Dejanović, R. Vaderna, G. Milosavljević, Ž. Vuković, “textX: A Python tool for Domain-Specific Languages Implementation”. Faculty of Technical Sciences, University of Novi Sad, 2016.

Kratka biografija:

Daniel Elero rođen je u Novom Sadu 1993. godine. Master rad na Fakultetu tehničkih nauka u Novom Sadu, iz oblasti Elektrotehnike i računarstva – softversko inženjerstvo, odbranio je 2018. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/01BE16Stanic

WI-FI KRIPTOPROCESOR

WI-FI CRYPTOPROCESSOR

Stefan Stanić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj –*Ovaj rad ima za cilj da upozna čitaoca sa CCMP kriptografskim protokol korišćenim u mnogim WiFi sistemima, kao i da dokumentuje i opiše projektovani sistem koji implementuje ovaj protokol u hardveru i softveru.*

Abstract –*This paper strives to provide the reader with a basic introduction to the CCMP cryptographic protocol, in addition to documenting the accompanying system that implements the protocol in question in hardware and software.*

Ključne reči: *AES, CCMP, HLS, WiFI.*

1. UVOD

CCMP-a (Counter mode Cipher block chaining Message authentication code Protocol, alternativno Counter mode CBC-MAC Protocol ili jednostavno CCM mode Protocol) je prvobitno imao namenu da unapredi i nasledi WEP (Wired Equivalent Privacy) enkripcioni protokol korišćen u 802.11 sistemima pre dodatka 802.11i amandmana. CCMP definiše metod enkripcije kao AES, mod operacije kao CCM (Counter mode CBC-MAC) i enkapsulaciju paketa kao dodavanje CCMP zaglavlja i MIC vrednosti.

Implementacija sistema je izvršena koristeći se Cadence-ovim Stratus HLS paketom alata.

2. PROCES DIZAJNA

Proces dizajna koristeći Stratus paket alata je dat na slici 1. Početni model na višem nivou apstrakcije se piše ili modifikuje tako da se sastoji samo od sintezibilnog podskupa SystemC naredbi [1], ovo predstavlja bihevioralni model sistema koji se dizajnira [2]. Dodatno se konstruiše Testbench koji ce se dalje koristiti za validaciju i debagovanje svih generisanih modela uključujući i trenutni bihevioralni model.

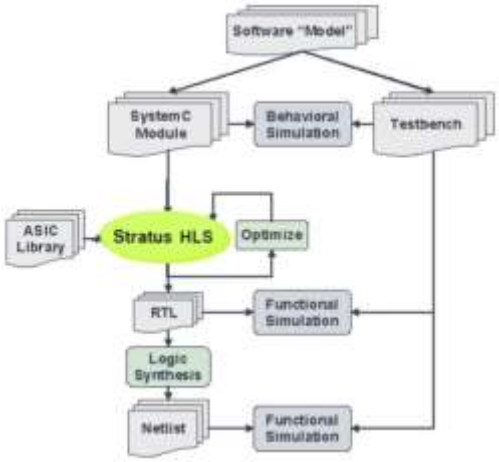
Sledeći korak je definisanje HLS parametara kao sto su frekvencija clock-a, clock slack i tip memorija koje se koriste. Dodatno ovo podrazumeva definisanje ASIC tehnologije koja će se koristiti kao i dodavanje HLS direktiva u SystemC kod bihevioralnog modela [2]. Ove direktive služe da definišu željene osobine konačnog RTL modela, primer često korišćene direktive je direktiva koja definiše vreme potrebno za izvršavanje određenog dela koda ili direktive koje definišu koji nizovi predstavljaju memorije a koji registre ili registarske banke.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bo prof. dr Rastislav Struharik.

Pri definisanju potrebnih direktiva pokreće se Stratus HLS alat za sintezu koji na osnovu SystemC koda, HLS direktiva kao i specificirane ASIC tehnologije sintetiše RTL model. Na osnovu karakteristika generisanog RTL modela se odlučuje da li je RTL dizajn zadovoljavajući, u slučaju da nije direktive se modifikuji i siteza se pokreće ponovo. Ovaj korak se ponavlja sve dok se ne postigne RTL model zadovoljavajućeg kvaliteta. Bitno je spomenuti da se prilikom ocene kvaliteta RTL modela može koristiti pre konstruisan Testbench za bihevioralni model u nepromenjenoj formi kako Stratus ima mogućnost korišćenja odgovarajućih wrapper-a za povezivanje SystemC testbench-a i sintetisanog RTL koda [2].
Generisani RTL kod se dalje koristi za logičku sintezu netliste. Ova netlista se takođe može validirati i debagovati koristeći početni Testbench kako i u ovom slučaju Stratus ima mogućnost generisanja potrebnih wrapper-a. Alternativno Stratus se može povezati sa eksternim RTL sintetizerima ili procesima dizajna [2]. U tom slučaju generisani RTL predstavlja kraj Stratusovog procesa dizajna i ovaj korak označava poziv nekog drugog RTL-to-GDSII procesa dizajna (GDSII predstavlja standardizovan format layout-a integrisanih kola).
Osim osnovnih prednosti HLS metodologije tj. značajno kraćem procesu dizajna kao i velikoj slobodi izmene arhitekture, Stratus paket alata pruža mogućnost korišćenja jedinstvenog Testbench-a na svim modelima nezavisno od nivoa abstrakcije koristeći odgovarajuće wrapper-e. Dodatno ovaj paket pruža gotove biblioteke interfejsa kao i floating point operacija veoma visokog stepena optimizacije [2].
Kako je deo CCMP algoritma implementiran u i softveru i hardveru postoji potreba za validacijom i testiranjem CCMP sistema u celosti. Ovo je izvedeno korišćenjem Cadence-ovog VSP (Virtual System Platform) alata. Ovaj alat ima mogućnost kosimulacije hardvera i softvera. SystemC kod hardvera se simulira u TLM (Transaction-Level Model) modu i to omogućava veoma brze simulacije, dovoljno brze da se ceo sistem, uključujući i softver, može efikasno simulirati [3].

3. AES

AES algoritam se bazira na principima permutacije, substitucije i linearne transformacije i prvobitno je bio opisan od strane NIST (National Institute of Standards and Technology) agencije Sjedinjenih Američkih Drzava u FIPS (Federal Information Processing Standards) 197 publikaciji[4]. Kako se ove 3 osnovne operacije veoma efikasno implementiraju u hardveru kao i u softveru ta činjenica predstavlja glavnu prednost AES algoritma nad njegovim konkurentima kao sto je DES.



Slika 1. *Stratus proces dizajna[2]*

Ovaj algoritam se sastoji iz iterativne primene određenih operacija nad trenutnim vrednostima state matrice. Ova matrica, prikazana na slici 2. podrazumeva stanje unutrasnjih podataka koji se menjaju izvrsavanjem svake od definisanih operacija.

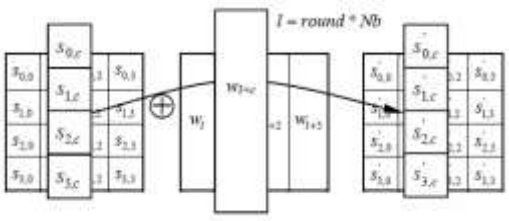
$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}$$

Slika 2. *State matrica*

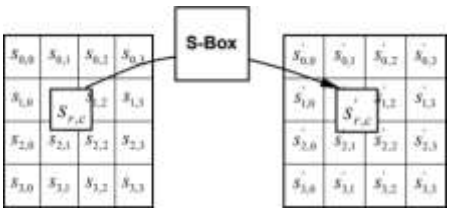
Kao sto se vidi sa slike 3 koja prikazuje pseudo kod AES algoritma primecuje se da se algoritam bazira na 4 osnovne operacije ilustrovane na slikama 4, 5, 6 i 7.

```
Cipher(byte in[4*Nb], byte out[4*Nb], word key_schedule[Nb*(Nr+1)])
begin
    byte state[4,Nb]
    state = in

    AddRoundKey(state, key_schedule[0, Nb-1])

    for round = 1 step 1 to Nr-1
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, key_schedule[round*Nb, (round+1)*Nb-1])
    end for

    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, key_schedule[Nr*Nb, (Nr+1)*Nb-1])

    out = state
end
```

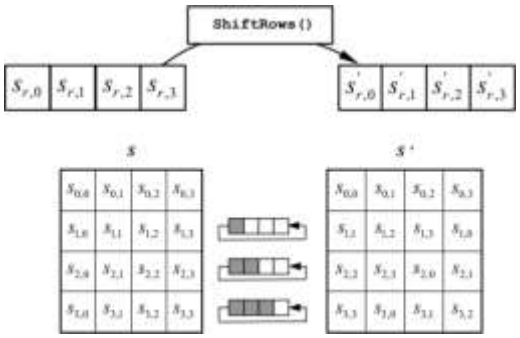
Slika 3. *Pseudo kod AES algoritma*



Slika 4. *Ilustracija AddRoundKeys operacije*



Slika 5. *Ilustracija SubBytes operacije*



Slika 6. *Ilustracija ShiftRows operacije*

$$S'_{0,c} = (\{0x02\} \bullet S_{0,c}) \text{ xor } (\{0x03\} \bullet S_{1,c}) \text{ xor } S_{2,c} \text{ xor } S_{3,c}$$
$$S'_{1,c} = S_{0,c} \text{ xor } (\{0x02\} \bullet S_{1,c}) \text{ xor } (\{0x03\} \bullet S_{2,c}) \text{ xor } S_{3,c}$$
$$S'_{2,c} = S_{0,c} \text{ xor } S_{1,c} \text{ xor } (\{0x02\} \bullet S_{2,c}) \text{ xor } (\{0x03\} \bullet S_{3,c})$$
$$S'_{3,c} = (\{0x03\} \bullet S_{0,c}) \text{ xor } S_{1,c} \text{ xor } S_{2,c} \text{ xor } (\{0x02\} \bullet S_{3,c}),$$

Slika 7. *Ilustracija MixColumns operacije nad c-tom kolonom*

Primecuje se da je za AES algoritam potreban key_schedule niz koji se generise putem KeyExpansion procedure opisane na slici 8. SubWord je naziv za SubBytes operaciju kojoj je operand jedna kolona dok RotWord predstavlja shift-ovanje bajtova reci za jedno mesto ka vise znacajnom bajtu. Rcon predstavlja niz konstantnih predefinisanih vrednosti prikazanih na slici 9.

```
KeyExpansion(byte key[4*Nk], word key_schedule[Nb*(Nr+1)], Nk)
begin
    word temp
    i = 0

    while (i < Nk)
        key_schedule[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
        i = i+1
    end while

    i = Nk
    while (i < Nb * (Nr+1))
        temp = key_schedule[i-1]
        if (i mod Nk = 0)
            temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
        else if (Nk > 6 and i mod Nk = 4)
            temp = SubWord(temp)
        end if
        key_schedule[i] = key_schedule[i-Nk] xor temp
        i = i + 1
    end while
end
```

Slika 8. *Pseudo kod KeyExpansion procedure*

```
unsigned char Rcon[51] = {
    0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40,
    0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a,
    0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a,
    0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39,
    0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25,
    0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a,
    0x74, 0xe8, 0xcb
}
```

Slika 9. *Vrednosti Rcon niza*

4. CCMP

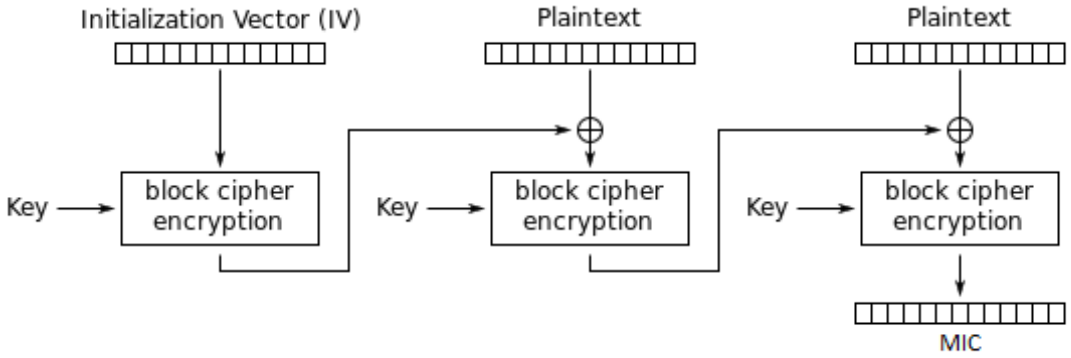
CCMP bazira se na CCM modu operacije AES enkripcionog metoda. Ovaj mod zahteva definiciju dva parametra kojima se definišu dužine određenih polja i vrednosti. Ovi parametri, L i M su definisani CCMP-om. Odabirom dužine kljuca AES enkripcionog metoda se definisu dve najčešće varijante CCMP-a tj CCMP-128 i CCMP-256 koje takodje predstavljaju odabir M parametra kao 8 ili 16 respektitivno dok L parametar ima fiksiranu vrednost 2 [5].

CCM mod operacije se sastoji iz dva dela. Prvi deo je zadužen za operaciju autentikacije tj provere integriteta poruke i koristi CBC-MAC mod operacije prikazan na slici 10 dok drugi deo pruža mogućnost enkripcije i dekripcije putem CTR moda operacije prikazanom na slici 11a i 11b respektitivno [5].

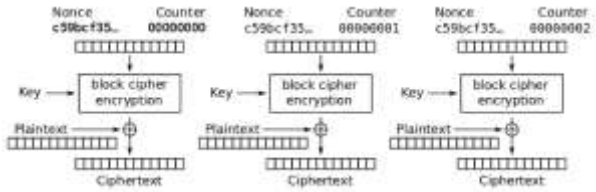
CCMP definise AAD (Additional Authentication Data) koji predstavlja podatke koji ce biti uzeti kao plaintext za CBC-MAC deo CCM moda ali ne i u CTR delu. Rezultat ovoga je da se ovi podaci samo autentikuju. AAD se konstruiše izbacivanjem Duration ID polja iz 802.11 MAC zaglavlja, kako je ovo polje promenljivo i nema potrebe da se autentikuje [5,6]. Konstrukcija AAD-a je prikazana na slici 12.

CCMP definise svoje zaglavlje koje je prikazano na slici 13. Ovo zaglavlje se smešta na početak rezultujućeg CCMP paketa i služi za transmisiju PN (Packet Number) vrednosti [5].

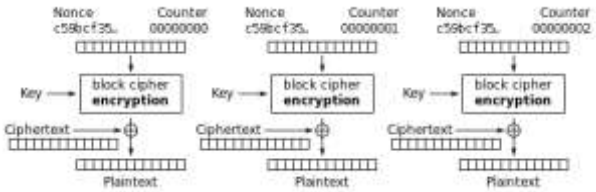
Nonce vrednost takođe definisana CCMP-om se koristi pri konstrukciji CTR brojača i CBC-MAC inicijalizacionog vektora [5]. Sadržaji nonce vrednosti i CBC-MAC inicijalizacionog vektora su prikazani na slikama 14 i 15.



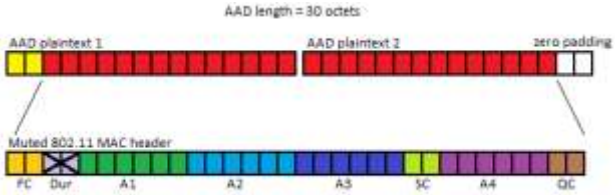
Slika 10. *CBC-MAC mod operacije[7]*



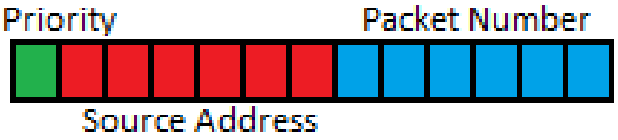
Slika 11a. *CTR mod operacije, enkripcija[7]*



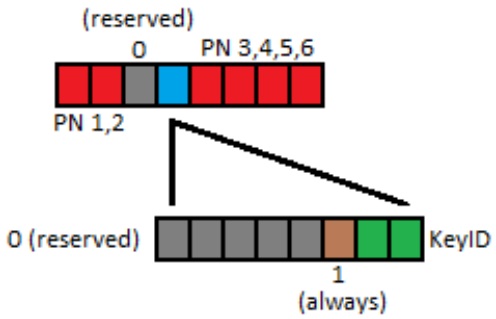
Slika 11b. *CTR mod operacije, dekripcija[7]*



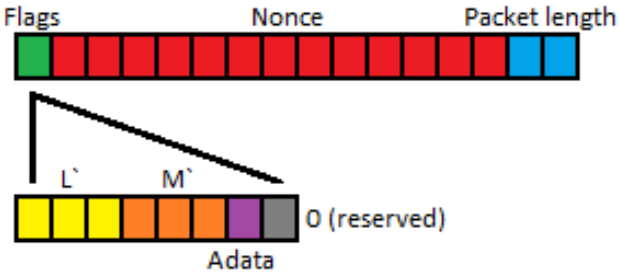
Slika 12. *Konstrukcija AAD-a*



Slika 14. *Konstrukcija nonce vrednosti*



Slika 13. *Konstrukcija CCMP zaglavlja*



Slika 15. *Konstrukcija CBC-MAC inicijalizacionog vektora*

5. IMPLEMENTACIJA

AES modul tj. AES 256 je sistem koji izvršava AES enkripciju nad dva ulazna podatka tj. operanda. Sistem takođe ima mogućnost izvršavanja procedure razvijanja ključa tj KeyExpansion procedure, za veličine ključa od 128 ili 256 bitova. Ova procedura će interno sačuvati raspored kljuceva tj key_schedule, i kasnije, pri operaciji enkripcije, koristiti jednistveni key_schedule za oba operanda, što podrazumeva jedinstvenu key_schedule RAM memoriju.

Glavni izazov implementacije ovog modula sistema je bio paralelizacija potrebnih pristupa memorijama. Tokom pre spomenute KeyExpansion procedure u key_schedule RAM se upisuju vrednosti. Za AddRoundKeys operaciju je potrebno izvršiti 4 citanja iz tog RAM-a sto znaci da je za jednu rundu AES-a minimalno potrebno 4 takta. Osim pristupa key_schedule RAM-u SubBytes operacija se sluzi pristupom S-box ROM memoriji tj memorijama. Sadrzaj ovih ROM memorija je konstantan i moze se paralelizovati pod pretpostavkom da su nam sve potrebne adrese ovih pristupa poznate što jesu. SubBytes operacija je implementirana tako da tokom pristupa key_schedule RAM-u stigne da izvrši svojih 16 pristupa S-Box ROM memorijama. Kao je pristup RAM-u 4 takta potrebno su 4 instance ROM memorija po operandu AES operacije.

WiFi kriptoprocesorski sistem je namenjen da uz pomoć softvera za formatiranje izvrsi CCMP operaciju nad podacima koji se nalaze u memoriji na koju je povezan, rezultat ove operacije smesti u tu memoriju kao i određenim Interrupt signalom javi sirem sistemu da je operacija završena.

Interfejs ka softverskom delu predstavlja AXI4 Lite magistrala putem koje se, pre pokretanja operacije, upisom u interne registre sistema definišu određeni parametri CCMP operacije kao sto su enkripcioni ključ, AAD, nonce, veličine i adrese ulaznih i izlaznih podataka [8]. Pri završetku ovih upisa operacija se pokreće upisom na određenu adresu ove magistrale.

Kao sto je pre spomenuto hadver ovog sistema ne izvršava CCMP u celosti iz ovog razloga je kontrolnom softveru potrebno dodati određene operacije kao što su formatiranje AAD podataka i upis CCMP zaglavlja na odgovarajuće mesto u memoriji.

Interfejs sa memorijom je implementiran putem AXI3 magistrale. Ne uzimajući u obzir kašnjenje pristupa sistemskoj memoriji kašnjenje ovog sistema je u velikoj meri određeno kašnjenjem AES operacija CCM moda. Činjenica da su kašnjenja prilikom pristupa sistemskoj memoriji daleko od zanemarljivih je podržala ideju da se pristupi memoriji tako vremenski rasporede da se uticaj ovih kašnjenja na kašnjenje cele operacije umanji. Ovo je izvršeno koristeći burst funkcionalnost AXI3 magistrale kao i implementacijom ovih pristupa tako da se oni dešavaju između AES operacija koje imaju fiksno kašnjenje definisano AES 256 DUAL modulom [8]. Kako je kašnjenje AES operacije značajno veće od kašnjenja idealnog pristupa sistemskoj memoriji, ovaj pristup ima olakšano minimalno vreme pristupa za postizanje minimalnog kašnjenja. Ovo znači da ce se i pri manjim zastojima usled zauzetka pristupa sistemskoj memoriji postići idealno kašnjenje CCMP operacije.

Ocekivana povrsina sistema je pri pocetku projekta iznosila 35.000 um^2. Vremensko ograničenje je definisano nad podatkom dužine 512 bajtova, podrazumevajuci frekvenciju sistemskog takta od 32 MHz i iznosi 160 us. Konacna površina sistema od 41.600 um^2 je zadovoljavajuća iako u maloj meri ne odgovara početnoj proceni dok su zahtevi brzine u potpunosti ispunjeni maksimalnim vremenom operacije od 73.25 us.

Kako je vreme izvršavanja značajno manje od zahtevanog, može se postaviti pitanje optimalnosti sistema jer je površina prevazišla procenu. Razlog iza ovog disbalansa jeste da su delovi sistema koji najviše utiču na ove karakteristike različiti. Na vreme izvršavanja skoro ekskluzivno utice AES sistem dok oko 75% površine sistema zauzimaju ostali delovi sistema. Upravo zbog toga je dalja optimizacija nemoguća kako bi smanjenjem površine bilo uvedeno neprihvatljivo kašnjenje.

6.ZAKLJUČAK

Kriptoprocesor je razvijen u cilju da bude integrisan u WiFi komunikacionim rešenjima kompanije Methods2Business. Kao deo ovih rešenja on je u potpunosti validiran kao deo WPA2 (WiFi Protected Access 2) sistema.

Pre opisanim optimizacijama su uspešno postignuti projektni ciljevi brzine rada kao i površine sistema. Očekuje se da ovo rešenje bude unapređeno podrškom za GCMP (Galois Counter Mode Protocol) kako postoji mogućnost da on bude prihvaćen kao brža alternativa CCMP-a u nasledniku WPA2 standarda tj. WPA3.

7. LITERATURA

[1] Ashfaq Khan, i drugi (Januar 2015). SystemC Synthesizable Subset, Version 1.4 Draft. Accellera Systems

[2] Stratus HLS User Guide (2017). Cadence

[3] Virtual System Platform User Guide (2016). Cadence

[4] NIST (November 26, 2001). FIPS Publication 197: Announcing the Advanced Encryption Standard (AES). Arhiviran original iz Marta 12. 2007.

[5] Whiting, i drugi (2003). Counter with CBC-MAC (CCM). The Internet Society

[6] IEEE Std 802.11ah-2016 (2016). IEEE Standards.

[7] Block cipher mode of operation (2013). Wikipedia. Dostupno na: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation. Pristupljeno 17. Juna 2018.

[8] AMBA AXI and ACE Specification (2011). ARM.

Kratka biografija:

Stefan Stanić rođen je 22.8.1994. u Novom Sadu. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu, na Katedri za elektroniku u septembru 2017. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE17Niksic

PREDIKCIJA INDEKSA KORISNOSTI IGRAČA U EVROLIGI

PREDICTION OF THE PERFORMANCE INDEX RATING OF BASKETBALL PLAYERS IN THE EUROLEAGUE

Bakir Nikšić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Problem kojim se bavi ovaj rad jeste predikcija indeksa korisnosti igrača u euroligi. Ovaj rad nastoji da iskoristi veliku količinu podataka, koji se nalaze na oficijalnom sajtu evrolige, obradi ih i pripremi za upotrebu algoritama mašinskog učenja. Algoritmi koji su razmatrani u ovom radu su Naive Bayes, SVM (Support Vector Machine) i KNN (K nearest neighbours). Za svaki algoritam izvšena je optimizacija parametara koji najviše utiču na samu klasifikaciju. Osim navedenih algoritama mašinskog učenja, razmatrana su još dva jednostvna algoritma za predikciju indeksa korisnosti koje je implementirao autor. Najbolju tačnost od 92% postigao je algoritam SVM, dok su algoritmi Naive Bayes i KNN imali manju tačnost od 87%, odnosno 72%. Namenski razvijeni algoritmi imali su značajnije manju tačnost od algoritama mašinskog učenja.*

Abstract – *This paper presents and evaluates machine learning (ML) as well as simple custom made approaches for the prediction of the Performance Index Rating of basketball players in the Euroleague. The approaches rely on the large amounts of data, available on the official website of the Euroleague. The data are processed and prepared in order to create a training and evaluation datasets for the application of ML algorithms. The algorithms which were considered are: Naive Bayes, SVM (Support Vector Machines) and KNN (K nearest neighbours). Parameters of each of the algorithms were optimized using a validation set. Besides the ML models, two simple custom made algorithms were designed and implemented by the author of this paper. The best accuracy of 92% was achieved by the SVM algorithm, while the Naive Bayes and KNN algorithms had a lower accuracy of 87% and 72% respectively. The accuracy of the custom made algorithms was significantly lower when compared to the machine learning models.*

Keywords: *prediction in sport, machine learning*

1. UVOD

U današnje vreme, veliki procenat populacije priklonjen je praćenju sporta. Kao jedan od najpopularnijih sportova današnjice, izdvaja se košarka. Kako bi praćenje košarke učinilo uzbudljivijim, određeni broj ljudi se upušta u pojedine vidove takmičenja. Fantazi (eng. Fantasy), predstavlja jedan od najpopularnijih vidova ovakve vrste takmičenja.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kovačević, vanr.prof.

U ovom takmičenju, učesnici pokušavaju da sastave što bolji tim uz ograničeni budžet. Takmičari kupuju igrače za koje smatraju da će ostvariti veliki indeks korisnosti. Ostvareni indeks korisnosti svih igrača se sabiraju i dobija se ukupni broj bodova koje je takmičar osvojio u trenutnom kolu. Ovo takmičenje traje kroz čitavu sezonu. Svaki učesnik ima svoj tim, i na kraju sezone, pobednik je učesnik koji je ostvario najviše bodova u sezoni. Ovaj rad bavi se predikicijom ostvarenog indeksa korisnosti igrača u Evroligi, na osnovu statistika igrača sa prethodno odigranih utakmica. U narednoj sekciji, biće opisani neki od radova koji su se bavili sličnim problemom. Nakon toga, u trećoj sekciji, biće opisana definicija problema i metodologije za njegovo rešavanje. Četvrta sekcija se bavi eksperimentalnom evaluacijom dobijenih rezultata, dok će u petoj sekciji biti reči o sumarizaciji rada i pravcima budućeg razvoja.

2. PRETHODNA REŠENJA

Veoma je mali broj radova koji se bavi rešavanjem problema predikcije indeksa korisnosti. Međutim, postoje radovi koji rešavaju slične probleme, koji mogu da posluže i koji predstavljaju na neki način vodilju, odnosno, put kojim bi trebalo ići, kako bi se došlo do rešenja ovog problema.

Prvo rešenje, čiji je problem rešavanja bio najsličniji rešavanju problema predikcije indeksa korisnosti u Evroligi, bio je rad [1]. Tema ovog rada bila je predikcija indeksa korisnosti košarkaša u NBA ligi. Na osnovu podataka već odigranih utakmica i metoda mašinskog učenja, prediktuje se indeks korisnosti igrača. Takođe, sabiranjem prediktovanih indeksa, vrši se predikcija i pobednika utakmice. Ovaj rad koristio je skup podataka u kojem su se nalazili statistički podaci igrača u predhodnim sezonama NBA lige. Sam skup podataka sadržao je podatke o broju pogođenih šuteva, promašenih šuteva, skokova, asistencija, blokadi, ukradenih lopti, izgubljenih lopti i postignutih poena. Ovo rešenje koristilo je metode mašinskog učenja, linearnu regresiju, slučajne šume i mašine potpornih vektora (Support vector machine, SVM). U radu se došlo do zaključka da se u predikciji najbolje pokazao metod SVM, koji je u najboljem slučaju pogađao indeks u opsegu +- 3.2 indeksna poena.

Sledeće rešenje koje je razmatrano bio je rad [2]. Tema ovog rada je predikcija indeksa najboljeg tima koji se sastoji od 8 igrača, sa ograničenjem da postoji ukupan budžet u koji se cena tih igrača mora uklopiti. Skup podataka koji je ovde korišćen, formiran je na osnovu

podataka iz sezona 2014/2015 i 2015/2016. Obeležja koja se nalaze u ovom dataset-u su statističke brojke koje je igrač ostvario tokom odigravanja utakmice. Tu spadaju procenti šuteva, broj skokova, asistencija, kao i mnogobrojni parametri koji utiču na način računanja indeksa korisnosti. Ovaj rad koristio je metode mašinskog učenja linearnu regresiju i naivni Bajesov klasifikator. U radu su se dobijeni rezultati upoređivali sa rezultatima koje je predvideo DraftKings [3] sajt i kao najbolji algoritam, pokazao se naivni Bajesov klasifikator.

Rad [4], bavi se predikcijom košarkaša koji će biti MVP (Most Valuable player). Model čija je tačnost 69%, izdvaja 5 najvažnijih atributa, odnosno atributa koji po autoru najviše utiču na indeks korisnosti. Najvažnijim atributima se dodeljuju težine, koje su izračunate na osnovu medijana prethodnih pet dobitnika ove nagrade. Takođe, model u obzir uzima i težinu koja se odnosi na tim za koji igrači nastupaju. Dodeljivanje težina se u ovom radu pokazalo kao izuzetno poboljšanje prethodnih modela čija je preciznost bila ispod 50%. Na osnovu toga, ustanovljeno je da i u rešavanju ovog problema takođe postoji prostor gde bi se određenim stvarima dodeljivalje težine.

3. METODOLOGIJA I ALATI

U ovoj sekciji predstavljene su primenjene metodologije za određivanje indeksa korisnosti igrača u Evroligi. Izložen je pristup u procesu dobavljanja i pretprocesiranja podataka, kao i svaki od 3 korišćena algoritma za mašinsko učenje. Pored algoritama mašinskog učenja, predstavljena su i dva namenski napisana algoritma od strane autora ovog rada. Dobijeni skup podataka sadržao ~70.000 redova. Ovaj skup podeljen je tako da 70% čini trening skup podataka, dok ostalih 30% čine testni skup podataka.

3.1. Prikupljanje podataka

Prvi i osnovni problem koji je bilo potrebno rešiti jeste, prikupiti dovoljno podataka, koji će se dalje obrađivati i pripremati, i na osnovu kojih će se dolaziti do prediktovanih vrednosti ostvarenog indeksa korisnosti igrača u Evroligi. Nije bilo moguće dobaviti skup podataka u već donekle pripremljenom formatu i kao takvog ga preuzeti sa interenta. Međutim, na oficijalnom sajtu Evrolige [5], nalazi se statistika igrača sa 17 prethodnih sezona (počevši od sezone 2000/01, pa do sezone 2017/18). S obzirom na to, bilo je potrebno iskoristiti tehnike web scraping-a (tehnike skidanja podataka sav eb stranica), kako bi se željeni podaci dobavili. Za ovako nešto, potrebno je bilo pronaći odgovarajući alat, sa kojim je moguće izvršiti *web scraping*. Istraživajući, pronađen je alatat *DataToolbar*. Ovaj alat radi tako što se definiše skript u kojoj se specificira, koji podaci se kojim redosledom dobavljaju sa veb stranice.

3.2. Predprocesiranje podataka

Nakon uspešno preuzetih podataka, naredni korak koji je bilo potrebno uraditi, jeste obraditi podatke i spremiti ih za dalju upotrebu. Potrebno je bilo primeniti razne tehnike *data wranglinga*, i pomoću njih doći do skupa podataka koji je spreman za dalje korišćenje. *Data wrangling* tehnike, su tehnike čišćenja, strukturiranja i prenošenja sirovih podataka u format koji je poželjan za dalje korišćenje podataka. Za ovako nešto, razvijen je alat koji je automatski rešavao sve probleme obrade podataka. Nakon automatske obrade skupa podataka, primećeni su neki specifični propusti prilikom obrade podataka i ovi problemi rešeni su ručno naknadno.

3.3 Redukcija dimenzionalnosti

Nakon završetka procesa opisanog u sekciji 3.3, skup podataka sadržao je 28 kolona, od kojih određeni broj kolona nije bio numeričkog tipa, poput npr. imena i prezimena igrača, dok veći deo jeste bio numeričke prirode. S obzirom da indeks korisnosti predstavlja numeričku vrednost, u formulu indeksa korisnosti ulaze samo numerički atributi, kojih je u ovom slučaju velik broj. Indeks korisnosti računa se po sledećoj formuli preuzetoj iz [6]:

$$\textbf{INDEX} = Pts + OReb + DReb + As + St + Rv + Fv - To - Pf - Ag - (FGA - FG) - (FTA - FT)$$

Za redukciju dimenzionalnosti iskorištena je linearne regresija, pomoću koje se tražila zavisnost određenih numeričkih atributa. Pomoću linearne regresije, jasno je mogla da se vidi zavisnost između atributa i da se pronađu atributi koji ne utiču na vrednost indeksa. Od 22 numerička atributa, linearna regresija je pokazala da 7 atributa nema nikakvu zavisnost sa ciljnom varijablom - indeksom, odnosno koeficijent korelacije iznosi 0 te su te kolone redukovane.

3.4 Diskretizacija ciljnog atributa

Nakon redukcije dimenzionalnosti i smanjenja broja atributa, dodat je novi atribut – grupa. Grupa je kolona koja predstavlja u kom opsegu se nalazi indeks koji je ostvario igrač na utakmici. Ovaj atribut je uveden kako bi se diskretizovao ciljni atribut, odnosno kako bi se mogli primeniti klasifikacioni algoritmi. Postoji 8 grupa:

- Grupa 1 [-50, 0)
- Grupa 2 [0, 5)
- Grupa 3 [5, 10)
- Grupa 4 [10, 15)
- Grupa 5 [15, 20)
- Grupa 6 [20, 25)
- Grupa 7 [25, 30)
- Grupa 8 [30, 35)
- Grupa 9 [35, 100)

3.5 Algoritmi za mašinsko učenje

Za potrebe primene algoritama mašinskog učenja, potrebno je bilo napraviti novi skup podataka, odnosno modifikovati već pripremljen skup podataka. Ova modifikacije bila je neophodna jer algoritmi mašinskog učenja rade sa istorijskim podacima. U skupu ne postoje ulazni parametri na osnovu kojih se predikutuje indeks korisnosti, s obzirom da indeks korisnosti zavisi od vrednosti parametara koje su ostvarene na istoj utakmici. U ovom radu je odabrano da se kao ulazni parametri koriste srednje vrednosti tih parametara ostvarenih na prethodnim utakmicama. Ovakav pristup očuvava najviše informacija i ne favorizuje davne ili skorije utakmice. Primenjena su 3 algoritma mašinskog učenja: Naive Bayes, SVM i KNN.

3.5.1 Naive Bayes

Klasifikator naivnog Bajesa (*Naïve Bayes*) spada u klasu statističkih klasifikatora. Zasniva se na Bajesovoj teoremi. Međutim, ovaj klasifikator zanemaruje uslovnu nezavisnost atributa u Bajesovoj teoremi - on

podrazumeva da vrednost jednog atributa u okviru jedne klase ne zavisi od vrednosti ostalih atributa u toj klasi. Ovo se naziva *naivna pretpostavka* [6] [7].

3.5.2 SVM (Support Vector Machines)

Mašine potpornih vektora (*Support Vector Machines - SVM*) su skup algoritama za mašinsko učenje, koji se koriste za klasifikaciju i regresiju. Ideja algoritma jeste da se u prostoru u kome su podaci predstavljeni, pronađe razdvajajuća hiper-ravan, tako da su svi podaci iz date klase sa iste strane ravni [8]. Prilikom evaluacije SVM klasifikatora isprobane su različite vrednosti optimizacionih parametara. U ovom istraživanju najbolje se pokazala *C-SCV* implementacija klasifikatora SVM sa radijalnom (*RBF*) kernel funkcijom. Klasifikator je davao različite tačnosti u zavisnosti od odabira parametra *C*. Na osnovu validacionog skupa, eksperimentalno je ustanovljeno da se najveća tačnost dobija kada je parametar *C*=80.

3.5.2 KNN (K Nearest Neighbour)

K najbližih komšija (*K nearest neighbour - KNN*) je algoritam mašinksog učenja koji se zasniva na primerima, tzv. lenjo učenje. Objekat se klasifikuje na osnovu broja najbližih komšija. K najbližih komšija nekog objekta, su k tačaka koje su od objekta udaljene manje od ostalih tačaka [7]. Prilikom evaulacije KNN klasifikatora isprobane su različite vrednosti optimizacionog parametra *k*. Na osnovu validacionog skupa, eksperimentalno je ustanovljeno da se najveća tačnost dobija kada je parametar *k*=5.

3.6 Namenski razvijeni algoritmi

U ovom poglavlju, biće opisani algoritmi koji su razvijeni od strane autora ovog master rada, za rešavanje problema predikcije ostvarenog indeksa korisnosti košarkaša u Evroligi. Motivacija za razvoj ovih algoritama je, pored postizanja velike tačnosti, i istraživačke prirode. Algoritmi mašinskog učenja mogu biti veoma kompleksni i vremenski zahtevni, kako za obučavanje, tako i za primenu. Iz tog razloga, izvršeni su eksperimenti sa dva jednostavna algoritma za predikciju indeksa korisnosti koji se oslanjaju na srednju vrednost ostvarenih statistika i potragu za već postojećom pojavom sekvence u skupu podataka. U ovom poglavlju biće opisana dva algortima koja su razvijena, a to su:

1. Algoritam srednje vrednosti
2. Algoritam slične sekvence

3.6.2 Algoritam srednje vrednosti

Algoritam srednjih vrednosi se zasniva na računanju srednjih vrednosti svih parametara zasebno, ostvarenih na prethodnim utakmicama. Osim toga, ideja ovog algoritma jeste da posebno favorizuje poslednjih pet utakmica, kao i utakmice koje su odigrane protiv ekipe sa kojom igrač igra utakmicu u kojoj se prediktuje njegov indeks korisnosti. Sasvim je očekivano da forma košarkaša varira tokom sezone. Zbog toga je poslednjih pet utakmica potrebno posebno favorizovati, u odnosu na ostatak utakmica, jer su to utakmice koje najobjektivnije opisuju trenutnu formu igrača. Isto tako, utakmica protiv ekipe sa kojom je već igrao ima poseban značaj, te bi i njih na neki način trebalo favorizovati.

3.6.2 Algoritam slične sekvence

Ideja algoritma slične sekvence je da pokuša da pronađe sličnu sekvencu ostvarenih indeksa korisnosti. Cilj je prediktovati *n*+1 utakmicu jednog igrača, tako što će se u skupu podataka naći *n* sličnih ostvarenih indeksa koje je ostvario na prethodnih *n* utakmica igrač. Ukoliko je pronađenih sekvenci više, računa se srednja vrednost svih dobijenih grupa pete utakmice iz pronađenih sekvenci i ona se prediktuje. Eksperimentalno je utvrđeno da je ovaj algoritam bilo moguće primeniti (samim tim i evaluirati) samo ako se predviđaju indeksi korisnosti do prvih 9 utakmica. Bilo je i pojedinačnih slučajeva kod kojih je predikcija bila moguća i posle 9 utakmica, ali je takvih slučajeva bilo vrlo malo.. Algoritam je davao rezultate dakle samo za prvih ~9 utakmica.

4. EVALUACIJA I REZULTATI

Eksperimentalna evaluacija se vrši kako bi se odredile performanse svih algoritama. Postoje različite tehnike evaluacije u sistemima analize podataka. U ovom istraživanju performanse klasifikatora evaluirane su upotrebom test skupa podataka. Modeli su se obučavali na 70% podakata iz skupa, dok se testiranje vršilo na ostalih 30%. Ovih 30% podataka nije bilo korišćeno ni na koji način tokom obučavanja. Metrike evaluacije performansi klasifikatora koje su korišćene u ovom radu su preciznost (eng. *precision*), povrat (eng. *recall*) i tačnost (eng. *accuracy*) [9]. Preciznost i povrat su metrike koje se određuju zasebno za svaku klasu. Tačnost je globalna metrika za određivanje mere uspešnosti klasifikatora [9].

Tabela 1. *Preciznost povrat i tačnost klasifikatora Naive Bayes po grupama*

Grupa	Preciznost	Povrat	Tačnost
Grupa1	85.23%	**93.91%**	85.23%
Grupa2	85.52%	78.57%	85.52%
Grupa3	87.21%	85.47%	87.21%
Grupa4	88.92%	89.45%	88.92%
Grupa5	91.41%	89.91%	91.41%
Grupa6	**94.47%**	94%	**94.47%**
Grupa7	92.94%	90.29%	92.94%
Grupa8	90%	91.30%	90%
Grupa9	71.86%	91.67%	71.86%

Ukupna tačnost klasifikatora naivnog Bajesa iznosi 87.16%. Preciznost je u opsegu 85%-95%, sa izuzetkom grupe 9, dok je povrat je takođe u opsegu 85%-95%, sa izuzetkom grupe 2.

Tabela 1. *Preciznost povrat i tačnost SVM klasifikatora po grupama*

Grupa	Preciznost	Povrat	Tačnost
Grupa1	**99.68%**	97.38%	**99.68%**
Grupa2	97.49%	99.70%	97.49%
Grupa3	94.44%	**99.95%**	94.44%
Grupa4	90.05%	91.46%	90.05%
Grupa5	75.87%	82.85%	75.87%
Grupa6	43.91%	47.75%	43.91%
Grupa7	0%	0%	0%
Grupa8	0%	0%	0%
Grupa9	0%	0%	0%

Ukupna tačnost klasifikatora mašina potpornih vektora iznosi 91.99%. Preciznost i povrat su u opsegu 44%-100%, sa izuzecima grupa 7, 8, i 9.

Ukupna tačnost klasifikatora k najbližih komšija iznosi 72.77%. Preciznost i povrat variraju od grupe do grupe i to u opsegu od 24%-100% sa izutkom grupe 9.

Tačnost je izračunata i za namenski napisane algoritme. Za algoritam srednje vrednosti tačnost je merena za dva slučaja. Prvi slučaj obuhvata prediktovanje indeksa

korisnosti igrača na osnovu njegovih rezultata u svim sezonama unazad. Drugi slučaj obuhvata prediktovanje indeksa na osnovu samo trenutno aktuelne sezone.

Tabela 2. *Preciznost povrat i tačnost KNN klasifikatora po grupama*

Grupa	Preciznost	Povrat	Tačnost
Grupa1	**99.65%**	**99.42%**	**99.65%**
Grupa2	92.31%	78.08%	92.31%
Grupa3	63.66%	63.31%	63.66%
Grupa4	51.44%	56.98%	51.44%
Grupa5	41.56%	49.05%	41.56%
Grupa6	32.76%	38.25%	32.76%
Grupa7	26.34%	36.57%	26.34%
Grupa8	23.38%	26.09%	23.38%
Grupa9	8%	8.33%	8%

Tabela 4. *Tačnost namenskog algoritma srednjih vrednosti*

Broj sezona	Tačnost
Sve sezone	31%
Trenutna sezona	**42%**

Tačnost je izračunata i za namenski napisan algoritam slične sekvence, kako bi se on mogao porediti sa rezultatima algoritama mašinskog učenja. Dobijena tačnost u slučaju kada ovaj algoritam vraća rešenje, iznosi 36%.
Analizirajući dobijene rezultate svih modela, ustanovilo se da se tačnost modela kreće u opsegu od 36% do 92%. U tabeli 6, može se videti tačnost svih iskorišćenih algoritama u rešavanju problema predikcije indeksa korisnosti igrača u Evroligi ponaosob.

Tabela 6. *Uporedni rezultati evaluacije svih algoritama*

Algoritam	Tačnost
Naive Bayes	87%
SVM	**92%**
KNN	72%
Algoritam srednjih vrednosti	42%
Algoritam slične sekvence	36%

5. ZAKLJUČAK

Problem koji je rešavan u ovom master radu je predikicija indeksa korisnosti igrača koji nastupaju u najkvalitetnijem evropskom takmičenjenu – Evroligi. U rešavanju ovog problema, ispitan je rad algoritama za mašinsko učenje: Naive Bayes, SVM i KNN. Korišćene su implementacije ovih algoritama u RapidMiner alatu. Osim ovih algoritama mašinskog učenja, namenski su implementirana dva algoritma: algoritam srednjih vrednosti i algoritam slične sekvence. Tačnost koja je dobijena u predikciji indeksa korisnosti igrača u Evroligi varira od algoritma do algoritma. Najveću tačnost ostvario je klasifikator mašina potpornih vektora, koji je prediktovao sa tačnošću od 92%, dok su se ostali algoritmi pokazali lošijim. Najmanju tačnost ostvario je namenski napisan algoritam slične sekvence.
Na osnovu prethodno dobijenih rezultata, može se zaključiti da ovaj rad predstavlja dobar korak ka rešavanju sličnih problema, koji se mogu rešavati. U svakom sportu postoji mera koja određuje koliko je neki igrač dobar, te bi ovaj rad mogao da bude slično rešenje takvih problema.

Ono što treba imati na umu, jeste to da na osnovu svega navedenog, idalje postoji prostora za unapređenje i pronalaženje boljeg i preciznijeg rešenja. Samo povećanje skupa podataka, utiče na rad pojedinih algoritama, te se očekuje da se njihov kvalitet rešenja poboljšava iz godinu u godinu, kako novi podaci pristižu. Osim toga, postoji prostora za unapređivanje poput upotrebe još nekih od algoritama mašinskog učenja. Takođe, u budućnosti bi moglo da se pokuša sa rigoroznijim opsegom prihvatljivosti onoga šta je tačno. Trenutne grupe pravljene su u opsezima od 5, dok bi u budućnosti bilo poželjno da se opsezi grupe smanje na recimo 3. Ovako nešto bi definitivno smanjilo tačnost trenutnih algoritama, te bi bilo poželjno razvitili ili iskorisiti neki algoritam koji bi za ovakve uslove davao veliku tačnost. Osim toga, sama tema bi mogla da se proširi i da se prediktuje odabir čitavog tima za fantasy takmičenje. Ovo bi uključilo i određena ograničenja poput budžeta i pozicija igrača. Takođe, bilo bi poželjno napraviti web stranicu koju bi budući korisnici mogli da iskoriste za svoje potrebe.

6. LITERATURA

[1] Chan-Hu-Shivakumar (2015) *Learning to Turn Fantasy Basketball Into Real Money Introduction to Machine Learning* https://shreyasskandan.github.io/files/report-ChanHuShivakumar.pdf

[2] Eric Hermann and Adebia Ntoso (2015), *Machine Learning Applications in Fantasy Basketball* http://cs229.stanford.edu/proj2015/104_report.pdf

[3] Draft kings official website https://www.draftkings.com/

[4] Mason Chen (2017) *Predict NBA Regular Season MVP Winner* http://ieomsociety.org/bogota2017/papers/9.pdf

[5] Euroleague official website https://euroleague.net/

[6] K. M. Leung, "*Naive Bayesian Classifier*", Polytechnic University, Department of Computer Science, Finance and Risk Engineering, November 2007.

[7] A. Kovacevic, Predavanja iz predmeta "*Sistemi za istraživanje i analizu podataka*", školska 2015/2016., Fakultet tehnilkih nauka, Univerzitet u Novom Sadu

[8] "Support Vector Machines", SciKit, http://scikit-learn.org/stable/modules/svm.html

[9] D. Petrović, "Pretprocesiranje podataka i generisanje skupa atributa za sentiment analizu Tviter poruka", Fakultet tehničkih nauka, Univerzitet u Novom Sadu, decembar 2016.

Kratka biografija

Bakir Nikšić rođen je 17.04.1994. u Prištini. Osnovnu školu „Miloš Crnjanski" u Subotici završio je 2009. godine. Tehničku školu „Ivan Sarić" u Subotici, završio je 2013. godine. Iste godine upisao je osnovne akademske studije na smeru Računarstvo i automatiku, Fakulteta tehničkih nauka u Novom Sadu. Osnovne studije je završio 2017. godine, nakon čega upisuje master akademske studije na Fakultetu tehničkih nauka, smer Elektronsko poslovanje. Položio je sve ispite propisane planom i programom.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE19Zaric

FUNKCIONALNO PROGRAMIRANJE I PROGRAMSKI JEZIK *F#*

FUNCTIONAL PROGRAMMING AND F# PROGRAMMING LANGUAGE

Jovica Zarić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu su opisani osnovni koncepti funkcionalne programske paradigme i njeno poređenje sa objektno-orijentisanom odnosno proceduralnom paradigmom. Opisan je programski jezik F# kroz svoje najvažnije osobine.*

Ključne reči: *funkcionalno programiranje, objektno-orijentisano programiranje, proceduralno programiranje, F#*

Abstract – *This paper describes key functional programming concepts and compares functional programming with objected-oriented and procedural programming. It presents key features of F# programming language.*

Keywords: *functional programming, object-oriented programming, procedural programming, F#*

1. UVOD

Programiranje je stvaranje računarskih programa (aplikacija) za obavljanje zadataka kroz niz definisanih, automatskih ili manuelnih, operacija. Računari su prisutni u modernom društvu u različitim oblicima: *PC* računari, mobilni i *tablet* uređaji, pametni satovi, računarski sistemi u prevoznim sredstvima, kućnim aparatima, poljoprivrednim mašinama, fabrikama.

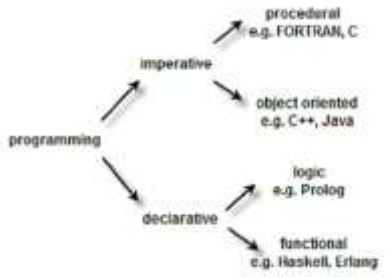
Pored velikog broja računarskih platformi, postoji i veliki broj programskih jezika, načina i alata za razvoj programa. Neki jezici su prilagođeni samo određenim platformama, neki se mogu koristiti za više njih. Razvijeno je i nekoliko programskih tehnika (paradigmi) za pisanje programa. Programska paradigma predstavlja pogled koji programeri imaju nad programom i njegovim izvršavanjem.

Paradigmom se definiše način pisanja, organizacije i izvršavanja koda kako bi isti bio pogodan i za čovjeka koji ga piše, čita i treba za razumije, tako i za računar koji ga izvršava. Slika 1.1 predstavlja osnovnu podjelu programskih paradigmi, podjelu na imperativnu i deklarativnu grupu.

Imperativna u koju spadaju proceduralno programiranje [1] i objektno-orijentisano programiranje (OOP) [2], akcenat stavlja na način na koji će se zadatak obaviti, dok deklarativno (funkcionalno i logičko) na sam zadatak, šta se treba obaviti, a ne i kako.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac, vanr. prof.



Slika 1.1 *Podjela programskih paradigmi*

2. UVOD U FUNKCIONALNO PROGRAMIRANJE

Na samom početku ere računara i programskih jezika (1940-tih godina), programski jezici su razvijani u skladu za arhitekturom računara. Jedan od prvih pristupa u razvoju programa je bio imperativni jer je konceptualno blizak hardveru računara. Izvršavanje programa predstavlja izvršavanje instrukcija programa od strane procesora koje mijenjaju sadržaj dodjeljene memorije i tok izvršavanja programa.

Funkcionalno programiranje [3] je nastalo primjenom matematičkih principa u programiranju. Razvojem matematičkog stila programiranja dolazi do udaljavanja od konkretne računarske arhitekture koja izvršava program što za posljedicu ima slabije performanse u odnosu na jezike imperativne paradigme. Funkcionalno programiranje posmatra pojedinačne operacije kao izračunavanja metamatičkih funkcija. Spada u deklarativne programske paradigme jer akcenat stavlja na to šta posmatrani kod radi, a ne i kako radi. Podržava funkcije koje koriste samo svoje ulazne parametre tako da pozivanje funkcije sa istim ulaznim argumentima uvijek vraća isti rezultat. Ovakav pristup omogućava lakše razumijevanje ponašavanja aplikacije, eliminaciju izmjena vrijednosti rezultata prethodnih izračunavanja (*mutable data*) i stanja aplikacije odnosno izazivanje *side* efekata. Programski jezici *C#, F#, Common Lisp, Clojure, Erlang, Haskell, Java, Javascript, Kotlin* pripadaju *impure* grupi funkcionalnih jezika jer pored funkcionalnog koncepta podržavaju i objektno-orijentisani i imperativni. *Pure* funckionalni jezici, npr. *Clean, Curry, Elm, Mercury,* podržavaju samo funkcionalno programiranje, ali su mnogo manje zastupljeni u razvoju komercijalnih softverskih aplikacija.

Lamdba račun [4], nastao u 1930-tim godina od strane američkog matematičara Alonza Čerča, predstavalja osnovu funkcionalnog programiranja. Lambda račun uvodi neimenovane, anonimne, funkcije koje opisuju neophodne ulazne parametre i način izračunavanja rezultata (listing 2.1).

square_num(x, y) = $x^2 + y^2$
(x, y) -› $x^2 + y^2$

Listing 2.1 *Koncept anonimne funkcije*

Lambda račun podržava samo funkcije sa jednim ulaznim parametrom koristeći *curring*. Funkcija sa dva parametra, može se predstaviti kao funkcija koja prima jedan parametar i vraća novu funkciju koja, takođe, prima jedan parametar. Pomoću *curring*-a, funkcija sa više parametara se predstavlja kao niz funkcija od kojih svaka prima samo po jedan parametar. Lambda račun posmatra funkcije kao vrijednosti prve klase (*first-class citizen*), te se funkcije mogu koristiti kao ulazni parametri ili biti vraćene kao rezultat drugih funkcija.

3. OSNOVNI KONCEPTI FUNKCIONALNOG PROGRAMIRANJA

Primjeri koda u ovom dijelu su pisani korišćenjem *Javascript* programskog jezika jer isti posjedu i funkcionalne elemente.

3.1 *Pure* (čiste) funkcije

Pure funkcijom se smatra jednostavna funkcija koja zadovoljava sljedeće kriterijume:

- Prima najmanje jedan ulazni parametar
- Pristupa vrijednostima samo svojih ulaznih parametara
- Uvijek vraća rezultat
- Za iste ulazne argumente uvijek vraća isti rezultat

Pure funkcija koristi vrijednosti samo svojih ulaznih parametara i ne pristupa ili mijenja vrijednost drugih promjenljivih. Ovim principom se izbjegavaju nepoželjni *side* efekti u programiranju jer utiču na izvršavanje drugih funkcija, unose *bug*-ove u kod i otežavaju otkrivanje istih. Funkcija koja ne prima nijedan ulazni parametar i ne može da mijenja vrijednosti promjeljivih van svog *scope*-a je beskorisna u programiranju jer može da vraća samo konstantnu vrijednost. Takođe, funkcija mora da vrati vrijednost kako bi manifestovala svoju namjenu i izvršavanje. *Pure* funkcija za iste ulazne paramatre uvijek vraća isti rezultat, te je neophodno da i funkcije koje ona poziva budu, takođe, *pure*. Listing 3.1.1 predstavlja primjer *pure* funkcije.

```
function Sum(a, b) {
return a + b;
}
```

Listing 3.1.1 *Pure* funkcija

Pure funkcije same po sebi nisu dovoljne za razvoj modernih softverskih aplikacija. Aplikacije podrazumijevaju upis i čitanje iz baze podataka ili fajla, otvaranje konekcije ka udaljenim serverima, te pojedine funkcije rade mnogo više od same obrade ulaznih parametara. Izvršavanje ovakvih funkcija može dovesti do pojave *side* efekata ili *exception*-a (izuzetaka) čime njihovo izvršavanje postaje nepredvidivo i one postaju *impure*. Funkcionalno programiranje ne može eliminisati *impure* funkcije, ali teži da njihovu upotrebu ograniči na ona mjesta gdje su zaista neophodne.

3.2 Immutability

U imperativnom programiranju, moguće je varijabli promijeniti prethodno dodjeljenu vrijednost (*mutable data*). Listing 3.2.1 predstavlja dozvoljenu operaciju u ovoj programskoj paradigmi.

```
var a = 3;
a = a + 1;
```

Listing 3.2.1 *Mutable* varijabla

Funkcionalno programiranje podržava *immutability* koncept koji zabranjuje izmjenu vrijednosti varijabli odnosno dozvoljava samo preuzimanje vrijednosti varijable. Samim tim koncept varijable ne postoji, ali se ipak naziv varijabla (promjenljiva) koristi iz istorijskih razloga. Funkcionalno programiranje čuva vrijednosti u konstantama. Ukoliko je potrebno izmijeniti vrijednost, kreira se nova konstanta sa novom vrijednošću. Ovim pristupom se piše jednostavniji i sigurniji kod, eliminišu se čak i slučajne izmjene vrijednosti.

3.3 Rekurzija

Immutability koncept onemogućava upotrebu *for*, *while* i *do while* petlji jer koriste brojačke promjeljive i promjeljive čijim bi se mijenjanjem vrijednosti postigli uslovi za izlazak iz petlje. Iz ovih razloga, koncept rekurzije (*recursion*) je veoma zastupljen u funkcionalnom programiranju. U svakom koraku rekruzije, funkcija vraća konkretnu vrijednost ili poziva samu sebe sa novim, izračunatim, vrijednostima argumenata poziva. Listing 3.3.1 predstavlja primjer rekurzivne funkcije za sabiranje niza cijelih brojeva.

```
function sum(first, last, currentSum) {
      if (first > last) {
            return currentSum;
      }
      return sum(first + 1, last,
      currentSum + first);
}
```

Listing 3.3.1 Primjer rekurzije

Osnovni elementi rekurzivne funkcije su:

- *Base case* – uslov za završetak rekurzije
- *Recursive call* – pozivanje funkcije u samoj sebi

3.4 Funkcije višeg reda

Za funkciju u funkcionalnom programiranju se kaže da je *first-class citizen* jer se funkcija tretira kao vrijednosti (slično kao *string* ili *int*). Ovo omogućava uvođenje koncepta funkcija višeg reda (*higher-order functions*) koje mogu da primaju funkciju kao argument ili vraćaju funkciju kao povratnu vrijednost. Takođe, definicija funkcije se može dodijeliti varijabli ili smjestiti u strukturu podataka. Funkcionalno programiranje podržava i *closure* princip koji povratnoj anonimnoj funkciji omogućava pristup svim lokalnim variablama funkcije koja ju je kreirala. Listing 3.4.1 predstavlja primjer anonimne funkcije kao povratne vrijednosti.

```
function makeAdd (increment) {
	return function (x) {
		return x + increment;
	};
}
```

Listing 3.4.1 *Anonimna funkcija kao povratna vrijednost*

3.5 *Curring*

Curring predstavlja razbijanje funkcije koja prima više parametara na kompoziciju više funkcija od kojih svaka prima samo jedan parametar.

Primjenom ovog principa, funkcija sa dva parametra se može predstaviti kao funkcija koja prima jedan parametar i vraća drugu funkciju koja prima drugi parametar. Izvršavanjem druge funkcije se dobija željeni rezultat. Jezici koji podržavaju ovaj koncept nude i sintaksu za lakše pisanje i čitanje koda.

Na listingu 3.5.1 se nalazi primjer funkcije napisan u *Javascript*-u upotrebom *arrow function* notacije.

```
var makeAdd = x => y => x + y;
var addFive = makeAdd(5);
var result = addFive(3);
```

Listing 3.5.1 Primjer *curring*-a

4. RAZLIKE U ODNOSU NA DRUGE PROGRAMSKE PARADIGME

4.1 Funkcionalno i objektno-orijentisano programiranje

Objektno-orijentisano programiranje (OOP) je programska paradigma, zasnovana na konceptu objekta kao gradivne jedinice aplikacije, koja omogućava definiciju klase kao šablona (*template*-a) za kreiranje i ponašanje objekata.

Klasa definiše podatke (atribute) koji predstavljaju stanje i metode (funkcije) koje predstavljaju ponašanje objekta kao i konstruktore koji kreiraju objekat date klase. Izvršavanje OOP aplikacije se svodi na kreiranje objekata i njihovu međusobnu interakciju odnosno mijenjanje stanja objekata. Tabela 4.1.1 predstavlja poređenje funkcionalnog i objektno-orijentisanog programiranja.

Tabela 4.1.1 *Poređenje funkcionalnog i objektno-orijentisanog programiranja*

Funkcionalno programiranje	**Objektno-orijentisano programiranje**
Koristi *immutable* podatke (zahtjeva više programske memorije)	Koristi *mutable* podatke (štedi memoriju)
Deklarativan pristup	Imperativan pristup
Izuzetno pogodno za paralelno izvršavanje	Nije pogodno za paralelno izvršavanje
Nema *side* efekte	Veoma prisutni *side* efekti
Redoslijed operacija nije toliko bitan	Redoslijed operacija je izuzetno bitan
Koristi rekurziju	Koristi petlje

4.2 Funkcionalno i proceduralno programiranje

Proceduralno programiranje (često se naziva i imperativno iako predstavlja njegovu podgrupu) posmatra program kao niz naredbi koji opisuje šta se treba uraditi. Zasnovano je na procedurama (*routines*) koje definišu instrukcije za pojedinačne korake čijim izvršavanjem se postiže željeni cilj. Prednost ovakvog pristupa je bliskost sa čovjekovim razmišljanjem pri rješavanju problema i sa arhitekturom računara što za posljedicu ima dobre performanse izvršavanja.

Mane proceduralnog programiranja u odnosu na funkcionalno su što je redoslijed izvršavanja procedura veoma značajan jer rezultat procedura direktno zavisi od trenutnog stanja programa. Veoma su prisutni *side* efekti i *mutable* podaci.

5. PROGRAMSKI JEZIK *F#*

F# [5] se pojavio u svijetu programskih jezika 2005. godine. Razvijen je od strane Majkrosofta [6] kao *strongly-typed* jezik koji podržava više programskih paradigmi. Primarno je funkcionalni, ali omogućava i imperativni i objektno-orijentisani stil programiranja. *F#* je *cross-platform* jezik te se može izvršavati na *Windows*-u, *Linux*-u ili *Mac OS*-u. Zasnovan je na *.NET* platformi tako da se lako može integrisati sa *C#*-om ili *Viusal Basic*-om i može koristiti sve *.NET* biblioteke.

Razvojna okruženja *Visual Studio* i *Xamarin* nude potpunu podršku za rad sa *F#*-om. Razvoj *F#* aplikacija moguć je još i u *Mono*, *MonoDeveloper* i *WebSharper* razvojnim okruženjima ili pomoću *Ionide* proširenja u *Atom*-u i *Visual Studio Code*-u.

F# nudi sintaksna pojednostavljenja u odnosu na *C#* i *Java* jezike. Izraze nije potrebno završavati sa tačka-zarezom (;). Kreiranje bloka izraza se ne postiže vitičastim zagradama, {}, nego indentacijom (uvlačenjem) izraza, te se izrazi u istom indentacionom nivou nalaze u istom bloku. Jednolinijski komentari počinju sa dvostrukim *slash* (/) karakterom, dok višelinijski komentari počinju sekvencom (*, a završavaju se sa *).

Podržava cjelobrojne tipove podataka, *signed* i *unsigned integer*, sa veličinom od 8, 16, 32 i 64 bita, kao i *bigInt* tip kojim se može predstaviti bilo koji cijeli broj. Od tipova podataka sa pokretnim zarezom podržani su 32-bitni i 64-bitni *float*, 128-bitni *decimal* tipovi. *Char* i *string* predstavljaju dostupne tekstualne tipove podataka, dok *bool* predstavlja logički tip. *F#* podržava osnovne binarne i unarne aritmetičke operatore primjenljive na *integer* i *float* tipove, zatim operatore poređenja, logičke i *bitwise* operatore.

Varijabla se definiše ključnom riječju *let* i = operatorom za dodjelu vrijednosti i podrazumijevano je da je *immutable*. Varijabla se može definisati i kao *mutable* sa *let mutable*, da bi joj se vrijednost mogla promijeniti korišćenjem ‹- operatora. Tip se može navesti nakon naziva praćenog dvotačkom (:). Ukoliko se tip ne navede,

F# kompajler će na osnovu dodjeljene vrijednosti dodijeliti tip varijabli.

F# nudi *if/elif/else* strukture za odlučivanje koje vraćaju vrijednost (suprotno *if*-u u *C#* i *Java* jeziku). Svaka grana vraća vrijednost svog posljednjeg izraza, te je neophodno da sve grane vraćaju vrijednosti istog tipa. Grana može da ne vrati vrijednost, odnosno da implicitno vrati *unit*, slično *void*-u u *C#*-u.

For...to i *for...downto* petlje omogućavaju eksplicitno navođenje broja iteracija. Pomoću *for...in* se može prolaziti kroz kolekcije. U *for* petljama nije moguće preskakati iteracije (*continue*) ili nasilno prekidati petlju (*break*). Podržana je još i *while...do* petlja.

Funkcija se definiše upotrebom ključne riječi *let* i navođenjem naziva i liste parametara bez tipa razdvojenih razmacima i tijela funkcije kao na listingu 5.1. Opciono, par naziv parametra : tip se može staviti u ( ) kako bi se specificirao tip parametra. Specificiranje tipa povratne vrijednosti (*return-type*) nakon dvotačke je opciono, jer kompajler može da odredi tip povratne vrijednosti na osnovu tipa vrijednost posljednjeg izraza u tijelu funkcije.

```
let function-name parameter-list [ :
return-type ] = function-body
```

Listing 5.1 Definicija funkcije

Rekurzivna funkcija se definiše korišćenjem *let rec* ključnih riječi. Operatori >> (*forward composition*) i << (*backward composition*) omogućavaju kompoziciju dvije funkcije u novu funkciju. Ulazni parametar naredne funkcije u kompoziciji mora po tipu odgovarati povratnoj vrijednosti prethodne funkcije. *Pipelining* (|> - *forward pipeline*, <| - *backward pipeline*) omogućava ulančavanje više funkcija kao uzastopnih operacija nad datom vrijednošću.

F# ima ugrađeni *option* tip za označavanje vrijednosti varijable koja može da postoji, a može i da ne postoji i ima dvije moguće vrijednosti *Some(x)* i *None*. *Option* nudi alternativu *null*-u u *C#-u* ili *Java* jeziku.

Dostupne strukture podataka su *tuple*, lista, niz, sekvenca, set, mapa, *record*.

F# podržava *discriminated* unije za definisanje skupa mogućih slučajeva u kojima varijabla može postojati. Korisne su za situaciju kada se jedan podatak može javiti u više oblika odnosno tipova. Slično ponašanje bi se u objektno-orijentisanom programiranju moglo postići definicijom bazne klase i više izvedenih što je znatno komplikovanije. Nedostatak *switch-case* strukture je nadoknađen *pattern matching*-om koji pored poređenja podataka omogućava vraćanje rezultata i dekompoziciju ulaznih podataka. Naročito je pogodan za korišćenje sa *discriminated* unijama.

F# u potpunosti omogućava objektno-orijentisano programiranje, te isti podržava klase, mehanizam nasljeđivanja i *interface*-e. Jezik još omogućava generičnost (*generics*), rad sa delegatima, mehanizme za rukovanje izuzecima, kao i ogranizaciju koda u module i *namespace*-eve.

F# je jezik opšte namjene. Može se koristiti za razvoj *Windows desktop* i *console*-nih aplikacija, za pisanje skript (.fsx fajlovi). Postoji nekoliko *framework*-a (*Suave, ASP .NET Core, Fable, WebSharper, Girrafe, Freya*) za razvoj veb aplikacija pomoću *F#*-a. Pogodan je za metematičke aplikacije jer se matamatičko razmišljanje može lako preslikati u *F#* kod.

6. ZAKLJUČAK

Funkcionalno programiranje predstavlja sasvim drugačiji pogled na organizaciju i izvršavanje programskog koda u odnosu na imperativni pristup.Većini programera imperativna odnosno objektno-orijentisana paradigma je osnovna ili jedina poznata prije svega zbog veće zastupljenosti OOP na računarskim i tehničkim fakultetima.

Razlog ovoga se može potražiti i u manjoj primjeni funkcionalne paradigme u razvoju modernih softverskih aplikacija prije svega zbog slabijih performansi funckionalnih jezika. Primjenom *mutability* koncepta i eliminacijom *side* efekata, funkcionalno programiranje omogućava pisanje koda sa manje *bug*-ova, koda u kojem se lakše i bezbjednije mogu praviti izmjene.

Takođe, testiranje i paralelizacija izvršavanja funkcionalnog koda je mnogo lakša. Zbog ovih prednosti, za očekivati je sve veću prisutnost funkcionalnog programiranja kako u formalnom računarskom obrazovanju tako i u razvoju komercijalnih softverskih rješenja.

7. LITERATURA

[1] Procedural programming, https://en.wikipedia.org/wiki/Procedural_programming

[2] Object-oriented programming, https://en.wikipedia.org/wiki/Object-oriented_programming

[3] Functional programming, https://en.wikipedia.org/wiki/Functional_programming

[4] Lambda calculus, https://en.wikipedia.org/wiki/Lambda_calculus

[5] FSharp, https://fsharp.org/

[6] Microsoft, https://www.microsoft.com

Kratka biografija:

Jovica Zarić rođen je 03.06.1992. godine u Bijeljini. Osnovnu školu „Jovan Dučić“ u Bijeljini završio je 2007. godine. U istom gradu je završio opšti smjer Gimnazije „Filip Višnjić“ 2011. godine. Zvanje diplomirani inženjer elektrotehnike i računarstva stekao je 2015. godine na Fakultetu tehničkih nauka u Novom Sadu.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38:620.9
DOI: https://doi.org/10.24867/01BE20Despotovic

ПРИМЕНА ЈЕДНОСМЕРНОГ НАПОНА ЗА НАПАЈАЊЕ ФРЕКВЕНТНО РЕГУЛИСАНИХ ЕЛЕКТРОМОТОРНИХ ПОГОНА

DC POWER SUPPLY OF FREQUENCY REGULATED ELECTRIC MOTOR DRIVES

Живадин Деспотовић, Веран Васић, Ђура Орос, *Факултет техничких наука, Нови Сад*

Област –ЕЛЕКТРОТЕХНИКА

Кратак садржај – *Избором децентрализованог концепта напајања фреквентно регулисаних вишепогонских транспортних трака, могуће је извршити централизовано исправљање напона, а уместо фреквентних претварача, непосредно код мотора, поставити инверторе. Пренос енергије од исправљача до инвертора се врши једносмерним напоном. Овакво напајање омогућава употребу каблова са проводницима мањег попречног пресека него при наизменичном напону. Анализа свих елемената напајања на примерима транспортних трака у рудницима указује на предности и оправданости оваквог напајања.*

Кључне речи: *Једносмерни напон, Фреквентни претварачи, Транспортне траке, "ETAP"*

Abstract – *By choosing a decentralized concept of power supply of frequency regulated multidrive conveyor belts, it is possible to perform centralized voltage rectifying, and instead of frequency converters install inverters directly on the motors or near them. The distribution of energy from the rectifier to the inverter is done by DC voltage. This type of power supply allows the use of conductors with the smaller cross section area than with AC voltage. The analysis of all elements of the power supply network in the examples of mining conveyor belts, points to the advantages and justifications of this type of power supply.*

Keywords: *DC voltage, Frequency converterc, Conveyor belts, ETAP*

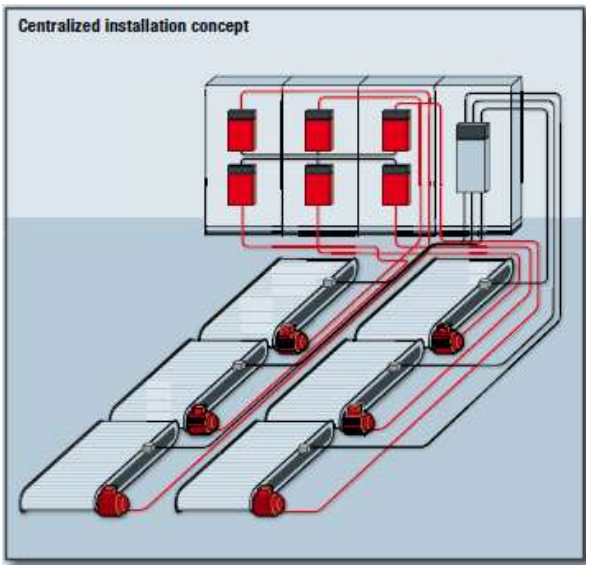
1. УВОД

У новије време се све више јавља идеја примене система напајања са једносмерним напоном на свим нивоима испоруке електричне енергије: преносу, дистрибуцији и потрошњи. Ово је подстакнуто појавом све већег броја потрошача једносмерног напона, повећањем броја произвођача електричне енергије из обновљивих извора, као и предвиђања везана за развој нових потрошача електричне енергије у будућности. Праћено је и развојем све јефтиније опреме за конвертовање напона – претварачи енергетске електронике [1].

Тема овог рада јесте анализа примене једносмерног напона у напајању индустријских фреквентно регулисаних транспортних трака. На једноставним примерима, указује се на постигнуте уштеде и упућује на сасвим извесну применљивост овог решења у напајању електромоторних погона транспортних трака.

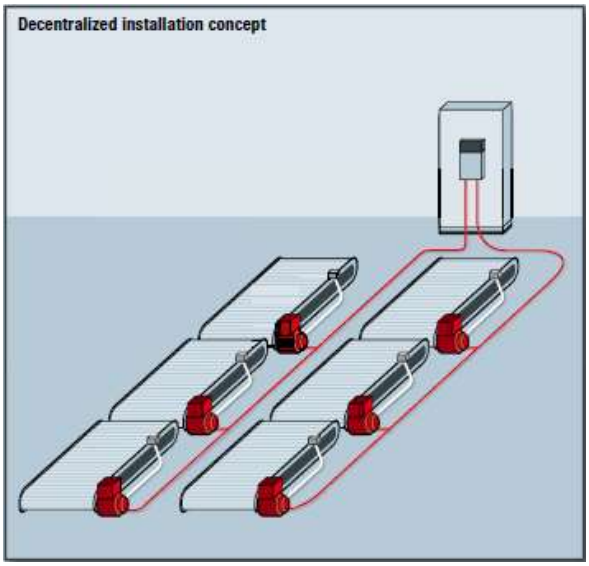
2. ТРАНСПОРТНЕ ТРАКЕ

Транспортне траке се налазе у готово свим гранама индустрије, папирној, аутомобилској, фармацеутској, прехрамбеној, рударској и др. Танспортне траке могу бити секционисане или несекционисане и свака, зависно од броја погона, може бити: једнопогонска или вишепогонска. Једнопогонске траке поседују један мотор на почетку или крају траке, док су код вишепогонских трака мотори распоређени дуж траке. У овом раду анализираће се вишепогонске фреквентно регулисане траке. Фреквентно регулисани мотори су рапоређени дуж траке и концепт њиховог напајња може бити централизован или децентрализован [2]. Централизовани концепт напајања подразумева да су сви фреквентни претварачи постављени на једно место. Код децентрализованог концепта, фреквентни претварачи су постављени непосредно поред мотора или су сједињени у компактну целину са мотором.



Слика 1. *Централизован концепт напајања*

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Ђура Орос, ванр. проф.



Слика 2. *Децентрализован концепт напајања*

Идеја за примену једносмерног напона за напајање вишепогонске фреквентно регулисане траке је следећа:
У децентрализованом концепту напајања фреквентно регулисане вишепогонске траке, потребно је: фреквентне претвараче заменити инверторима и повезати их на нисконапонску дистрибутивну мрежу једносмерног напона. Дистрибутивну мрежу формирати применом централног исправљача довољне снаге за снабдевање целокупог погона траке.

Применом оваквог концепта напајања очекује се смањење губитака у кабловима и елиминација реактивне снаге каблова. Структура губитака је промењена због промене природе извора напајања. Предпоставка је да смањени губици омогућују примену напојног кабла мањег попречног пресека.
Губитке у проводнику представља топлотна снага развијена услед протицања наизменичне струје. Ова снага је већа од снаге Џулових губитака који представљају производ отпорности проводника (при протицању једносмерне струје) и квадрата ефективне вредности струје.

Услед протицања наизменичне струје наелектрисања се, услед индукованог електричног поља, потискују ка рубу проводника и тиме се фактички смањује ефективна површина попречног пресека, те се и отпорност проводника повећава. Ефекат потискивања или површински ефекат зависи од фреквенције али и од површине попречног пресека. Поред наведеног, утицај на ефекат потискивања, у одређеној мери, имају и виши хармоници струје због своје увећане фреквенције.

Поред ефекта потискивања, јавља се и ефекат близине. Ефекат близине представља утицај другог проводника кроз који протиче струја. Услед протицања наизменичне струје успоставља се наизменично магнетно поље које својим присуством мења расподелу тока наелектрисања у првом проводнику.

Ова промена такође утиче на отпорност проводника јер смањује ефективну површину попречног пресека проводника. Поред фреквенције и површине попречног пресека проводника битан је и геометријски положај проводника. Меру утицаја, односно мегнетске спреге два проводника представља међуиндуктивност проводника. При индустријској фрквенцији, Џулови губици се често изједначавају са укупним губицима.

Претходно наведено наводи да се успостављањем једносмерне струје кроз проводник елиминишу губици услед ефекта потискивања и сузбијају губици услед ефекта близине. Излазни напон исправљача је већи од ефективне вредности улазног наизменичног напона. Коришћењем овог напона за пренос енергије смањила би се струја кроз проводник, а тиме би се смањили и Џулови губици.

Губици у фреквентним претварачима се могу груписати на губитке у исправљачком делу и на губитке у инверторском делу. Инверотор је сачињен од шест пуноуправљивих полупроводничких елемената (IGBT). Губици се јављају током прелазних режима комутације компоненти и губитака током провођења. Губици су највећи током прелазних режима. Пошто сваки погонски мотор транспортних трака задржава свој инвертор, овде се неће разматрати уштеда енергије на инвертору.

Предпоставка је да је снага губитака код централизованог исправљања приближно једнака укупним губицима код децентрализованог исправљања. На основу тога се може сматрати да се централизацијом исправљача неће смањити укупни губици исправљања напона.

Нелинеарни потрошачи, у коју групу спадају и исправљачи, генеришу више хармонике струје. Укупно изобличење струје зависи не само од снаге нелинеарних потрошача већ и од укупне снаге свих прикључених потрошача. Код децентрализованог исправљања напона услед напојних каблова и њихове дужине, део хармоника се филтрира у самим кабловима, јер се они у овом случају могу посматрати као RL филтери. Код централизованог исправљања, генерисање виших хармоника је такође централизовано и налази се непосредно код извора напајања (у овом случају трансформатор се посматра као извор напајања нисконапонске мреже), те се могу очекивати непогоднији услови за рад трансформатора, односно веће изобличење струје.

На основу претходног може се сматрати да се централизацијом исправљача очекује стварање погоршаних услова за рад трансформатора услед појаве виших хармоника струје.

У складу са претходним се спроводи анализа напајања транспортне траке у руднику, најпре са напајањем са наизменичним напоном а потом и са једносмерним.

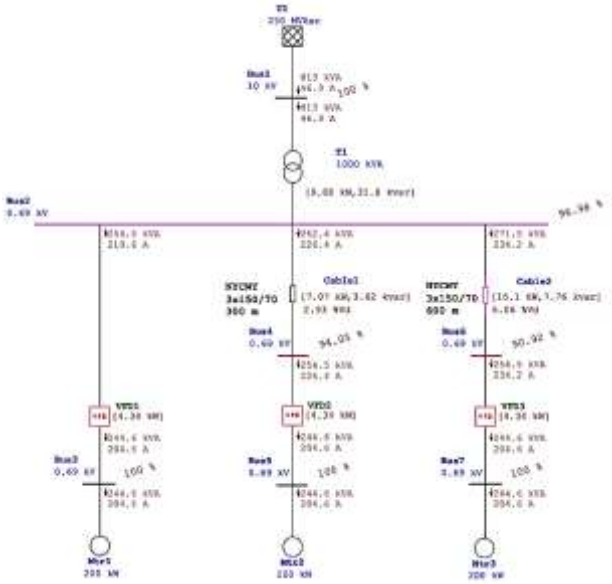
3. АНАЛИЗА НАПАЈАЊА ТРАНСПОРТНИХ ТРАКА У РУДНИЦИМА

За анализу електричног напајања погона траке, најпре је потребно одредити број погона, њихове снаге и позиције у односу на целу транспортну траку. У овом раду, за пример анализе, подаци о механици транспортне траке су преузети из литературе [3]. На тај начин је за реалан пример преузет маханички подсистем транспортне траке као и дефиниција броја и снаге електромоторних погона који га покрећу. Подаци о овој транспортној траци дати су у табели 1.

Табела 1. *Основне карактеристике транспортне траке*

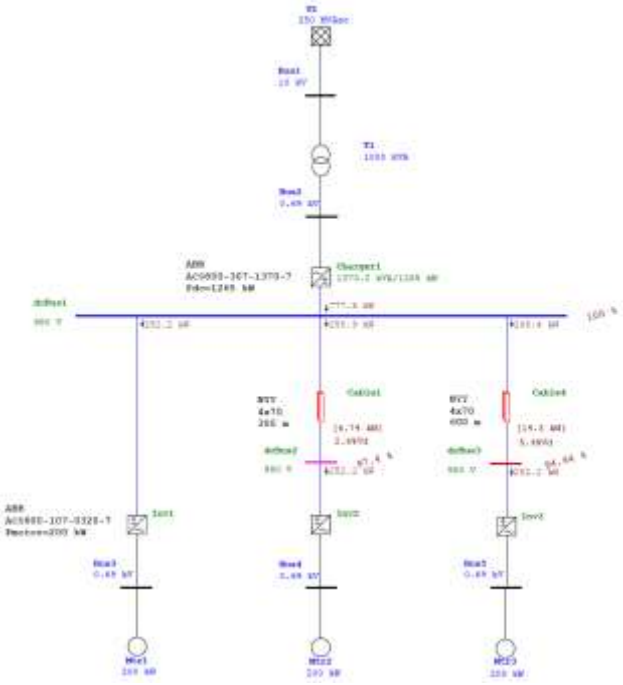
Дужина транспортне траке	600 m
Број погона	3
Позиција погона	Почетак, средина и крај траке
Појединачна снага погона	200 kW
Извор напајања	Централизован, на почетку траке

На основу овако дефинисане транспортне траке извршен је одабир свих елемената електричног напајања и спроведена детаљна анализа. За моделовање напајања траке при наизменичном напону напајања, одабрани су мотори, произвођача Siemens типа 1CV1315B. Фреквентни претварачи за контролу рада мотора су такође произвођача Siemens, серије Sinamics G150. Каблови за напајање мотора су типа NYCWY, пресека 3x150/70 mm^2, а примењени систем напајања је TN-C. Називни наизменични напон напајања је 690 V.



Слика 3. *Шема напајања траке при наизменичном напону са резултатима анализе токова снага*

За моделовање напајања траке при једносмерном напону, одабрани су исправљачи и инвертори произвођача ABB серије ACS800. Одабрани каблови за напајање једносмерним напоном су типа NYY, пресека 4x70 mm^2. Поново, систем напајања је TN-C као и код напајања са наизменичним напоном. Називни једносмерни напон напајања је 980 V. На слици 4 приказана је шема напајања траке, са резултатима анализе токова снага. Анализа је спроведена применом програмског пакета ETAP. Каблови су означени црвеном бојом, јер је струја оптерећења већа од дозвољене. Разлог овоме је што програм посматра оптерећење по проводнику, док је реално оптерећење подељено на два проводника. Стога, струја оптерећења другог извода износи приближно 264 A, док је оптерећење једног проводника 132 A. Трајно дозвољена струја кабла NYY 4x70 mm^2 са три оптерећена проводника износи 162 A.



Слика 4. *Шема напајања траке при једносмерном напону са резултатима анализе токова снага*

Сматра се да је кабел директно укопан без утицаја редукционих фактора. Називна струја оптерећења за четири оптерећена проводника мора бити мања него са три, због термичких напрезања. Дакле сматра се да је струја оптерећења од 132 A дозвољена за наведени кабел.

У табелама 2 и 3 приказани су резултати токова снага. Поређења ради, укупни губици привидне снаге износе 24,94 kVA, док су укупни губици при једносмерном напону 21,03 kW.

Табела 2. *Резултати анализе при наизменичном напону*

Кабел	Дужина [m]	Струја кабла [A]	Пад напона [%]	Губици [kW]	Реактивна снага кабла [kVAr]
Cable1	300 m	226,4	2,93	7,07	3,62
Cable4	600 m	234,2	6,06	15,13	7,76
Укупни губици и укупна реактивна снага каблова				**22,20**	**11,38**

Табела 3. *Резултати анализе при једносмерном напону*

Кабел	Дужина [m]	Оптерећење кабла [kW]	Пад напона [%]	Губици [kW]
Cable1	300 m	258,9	2,60	6,743
Cable4	600 m	266,4	5,36	14,283
Укупни губици				**21,02**

Одавде се закључује да су губици и падови напона приближно једнаки, стим што благу предност има једносмерни систем.
Значајна предност једносмерног система је у инвестиционој вредности каблова којима су обезбеђена напајања мотора. За реализацију напајања наизменичним напоном, примењен је кабел типа NYCWY 3x150/70 mm^2, док је за напајње једносмерним напоном примењен кабел NYY 4x70 mm^2. Да би се извршило приближно поређење цена каблова, довољно је одредити однос укупне запремине бакра. Запремина бакра у систему са наизменичним напоном износи:

$$V_{Cu-NYCWY} = 3 \cdot S_{150} \cdot l + S_{70} \cdot l = 0{,}468\ m^3 \qquad (1)$$

Док је запремина бакра у каблу у систему са једносмерним напоном:

$$V_{Cu-NYY} = 4 \cdot S_{70} \cdot l = 0{,}252\ m^3 \qquad (2)$$

На основу претходног закључује се да је запремина бакра у напојним кабловима једносмерног система мања за приближно 45%, што упућује и на знатно нижу цену кабла.
Да би се стекао комплетнији увид о уштедама приликом преласка на једносмерни напон извршена је анализа једне транспортне траке у руднику Бор. Основне карактеристике траке која спаја стари и нови коп су дате у табели 4, док је на слици 5 приказана натписна плочица једног мотора за погон траке.

Табела 4. *Основни подаци траке*

Капацитет	4700	t/h
Дужина	2650	m
Ширина	1600	mm
Брзина	3,4	m/s
Број погона	4	
Појединачна снага погона	1000	kW
Начин покретања мотора	Роторски упуштач	

Слика 5. *Натписна плочица мотора*

Као што се види на натписној плочици мотора, називна струја мотора износи 116,5 A. Мотор је високонапонски, 6 kV, са намотаним ротором. На основу спроведене анализе закључује се да је могућа употреба каблова за ред мањег површинског пресека. Проценат уштеде на инвестициониој вредности каблова би био приближно исти као у приказаном примеру. Да би прелазак на једносмерни напон био могућ, неопходна је употреба средњенапонских исправљача и инвертора.

4. ЗАКЉУЧАК

На примеру транспортне траке за пренос материјала у рудницима, анализирана је примена једносмерног напајања фреквентно регулисаних електромоторних погона. У наведеним примерима се показује да се може уштедети на инвестиционој вредности напојних каблова претварача мотора. Као резултат спроведене анализе добија се да је смањење количине бакра око 45% применом једносмерног напајања у односу на адекватно напајање наизменичном струјом.

На основу приказаних анализа, резултата и закључака могуће је констатовати и следеће: Прелазак на једносмерни напон, треба тражити на граничним појединачним снагама за дефинисани напон. Другим речима, приликом пројектовања напајања траке, када су снаге оптерећења на граници одлуке за прелазак на виши напонски ниво, али опет недовољне, треба размотрити примену једносмерног напона која би, на основу ове анализе, била сасвим оправдана.

5. ЛИТЕРАТУРА

[1] "LVDC: The better way", IEC - Електронска брошура - http://www.iec.ch/about/brochures/pdf/energy/iec_lvdc_the_better_way_en_lr.pdf
[2] "SAVING, Decentralized installation concepts with mechatronic drive systems", SEW Eurodrive - Електронска брошура - https://download.sew-eurodrive.com/download/pdf/19407211.pdf
[3] Z. Despodov, S. Mijalkovski, V. Adziski, Z. Panov, " Selection of Belt coneyors drive units number by Technical-Economical Analysis", Applied Mechanics and Materials Vol. 683 (2014), pp 189-195

Кратка биографија:

Живадин Деспотовић рођен је у Лозници 1989. год. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Енергетска електроника и електричне машине одбранио је 2018. год.
Контакт: zivadin.despotovic@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE22Savic

GRAF BAZE PODATAKA

GRAPH DATABASE

Ivan Savić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu su opisane razlike između graf baze podataka i ostalih tipova baza podataka. Rad je ispratila implementacija softvera u programskom jeziku Java. Zadatak softvera je da uspostavi konekciju sa Neo4j graf bazom podataka.*

Ključne reči: *Graf, baza podataka, Java, Neo4j, softver*.

Abstract – *The thesis describe differences between graph database and other types of database. With thesis, there is an implementation of software in Java programming language. Task of the software is to establish a connection with Neo4j graph database.*

Keywords: *Graph, database, Java, Neo4j, software*.

1. UVOD

Baze podataka oduvek su se smatrale jednim od najvažnijih delova sistema, odnosno aplikacija za različite namene. Osobina koja je ključna jeste trajno skladištenje podataka i dobavljanje istih kada je to potrebno.

Danas, najveći udeo na tržištu čine relacione baze podataka. Relaciona baza podataka je poseban tip baze podataka kod kojeg se organizacija podataka zasniva na relacionom modelu.

Podaci se u ovakvim bazama organizuju u skup relacija između kojih se definišu određene veze. Relacija se definiše kao skup n-torki sa istim atributima, definisanih nad istim domenima iz kojih mogu da uzimaju vrednosti. U relacionim bazama podataka, svaka relacija mora da ima definisan primarni ključ, koji predstavlja atribut pomoću kojeg se jedinstveno identifikuje svaka n-torka. Relacija opciono može da poseduje i spoljni ključ, preko kojeg se ostvaruje veza sa drugim relacijama.

Upravljanje ovakvim bazama podataka se realizuje preko sistema za upravljanje relacionim bazama podataka. O karakteristikama relacionih baza podataka, njihovim prednostima i manama, biće reći u narednim poglavljima ovog rada.

Pored pomenutih - relacionih baza podataka, svoje mesto na tržištu pronašle su i tzv. NoSQL (*Not only SQL*) baze podataka koje svojim osobinama i načinom upravljanja podacima nadoknađuju pojedine nedostatke relacionih baza podataka.
U ovom će radu naglasak biti na proučavanju osobina i načinu izvršavanja upita nad spomenutom kategorijom baza podataka, kao i na prednostima i nedostacima prilikom njihovog izvršavanja. Detaljnije će se analizirati načini izvršavanja upita, pre svega, u novije vreme sve popularniji, Cypher Query Language (u nastavku Cypher).
Spomenuti upitni jezici će se uporediti sa upitnim jezikom za relacione baze podataka (u natavku SQL). Tematika rada će biti usredsređena na graf baze podataka. Graf baze podataka predstavljaju poseban oblik NoSQL baza podataka.
U novije vreme ovaj tip baza podataka je sve korišćeniji, i kao takav nama postaje sve zanimljiviji za korišćenje i na nekin način testiranje.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Branko Milosavljević.

2. RELACIONE BAZE PODATAKA

Relaciona baza podataka se može definisati kao organizovana kolekcija podataka u kojoj su podaci organizovani u skup relacija [1]. Za uvođenje koncepta relacionih baza podataka najvećim delom je zaslužan Edward Codd, koji je ga je prvi spomenuo u svom radu 1970. god. U navedenom radu Edward Codd pojašnjava osobine i mogućnosti relacionog modela koji čini osnovu ove najčešće korišćene vrste baze podataka.

Kao jednu od važnijih osobina tog modela Edward Codd izdvaja činjenicu da su sve informacije u relacionim bazama podataka dostupne i predstavljene kao vrednosti u tabelama. Ova osobina omogućava programerima, ali i krajnim korisnicima, da pristupe podacima preko vrednosti podataka, a ne preko njihove pozicije u sistemu (kao što je bila ranija praksa u domenu baza podataka). Osnovna karakterisitka relacionih baza podataka jeste reprezentacija podataka u obliku dvodimenzionalnih nizova, tj. tabela određena sa n brojem redova i m brojem kolona.

Zbog načina njihovog predstavljanja u praksi mnogi koriste pojam tabela kao zamena za pojam relacija, kolona tabele kao zamena za atribut i red tabele kao zamenu za n-torka.

Dakle, relacione baze podataka svoju dugogodišnju popularnost duguju svojoj jednostavnosti i razumljivosti za osobe s različitim nivoima zanja i sposobnosti.

3. GRAF BAZE PODATAKA

S povećanjem složenosti i povezanosti podataka modeliranje podataka u obliku međusobno povezanih relacija više nije zahvalan zadatak. Zbog toga se danas kao prirodniji način modeliranja podataka sve više nameće modeliranje podataka u obliku grafova koji se sastoje od međusobno povezanih čvorova.

U tim grafovima čvorovi (*Nodes*) predstavljaju objekte u stvarnom svetu i okolini, dok veze (*Edges*) predstavljaju veze između tih objekata.

Tehnologija graf baza podataka omogućuje potpuno iskorišćavanje potencijala savremenih podataka u jednostavnijim, ali i složenijim situacijama. Poslednjih godina širok spektar mogućnosti, korisnost i performanse vodećih graf baza podataka su došle na zavidan nivo zrelosti, pa graf baze podataka već i danas zauzimaju značajan udeo na tržištu, a koji konstantno raste. Osim toga, graf baze podataka na važnosti dobijaju i zbog razvoja koncepta Interneta stvari (*Internet of Things*, u nastavku teksta IoT) jer on u svojoj u definiciji uključuje povezivanje uređaja, a time i podataka, što čini osnovu graf baza podataka.

Sistem za upravljanje graf bazama podataka je sistem sa Create, Read, Update i Delete, tzv. CRUD operacijama koje razotkrivaju i deluju nad graf modelom podataka [2]. Prema poslednjim podacima u avgustu 2018. godine sistemi za upravljanje graf bazama podataka zauzimaju 1.3%. Još uvek veliki udeo imaju relacioni DBMS sistemi (77%), pa se informacija o korišćenju graf baza podataka možda čini neznatnom.

Jedan od razloga velikog porasta popularnosti graf baza podataka na tržištu sigurno je činjenica da vrlo uspešno utiču na složene i dinamične veze između čvrsto povezanih podataka [2]. Dakle, graf baze podataka su sve popularnija kategorija NoSQL baza podataka koju je najbolje koristiti za reprezentaciju i pretraživanje povezanih podataka značajnih veličina.

3.1. Definicija grafa

Graf G je uređeni par (V, E). Elementi skupa V se zovu čvorovi (*Vertex*), a elementi skupa E grane (*Edge*) grafa G. Za dati graf G, skup čvorova se označava sa V(G), a skup grana sa E(G). Svaki čvor grafa ima jedinstveni identifikator (ekvivalent primarnom ključu kod relacionih baza podataka) i oznaku (eng. label) koja može biti svojstvena većem broju čvorova u grafu.
U praksi ovako jednostavno definisana struktura grafa omogućava modeliranje podataka iz bilo kojeg domena problema.
Grafovi se mogu iskoristiti za lakše razumevanje različitih oblasti poput nauke, poslovnih aktivnosti, vlade, društvenih mreža, medicine, itd.

3.2. Algebra grafova

Algebra grafova na kojoj se temelji graf model podataka predstavlja proširenje već spomenute relacione algebre na kojoj se temelji istoimeni model podataka. Međutim, postoje značajne razlike između navedenih algebri. Nekim operatorima relacione algebre pridružen je određeni operator u algebri grafova (s razlikama ili bez), ali su u algebri grafova definisani i novi operatori.

3.3. Graf model podataka sa atributima

Graf model podataka sa atributima (*Property graph data model*) čini osnovu graf baza podataka, a sadrži povezane entitete sa atributima navedenih u samoj definiciji grafa:
Čvorovi - predstavljaju entitete u stvarnom svetu, svaki ima svoj identifikator i atriute u obliku parova ključ-vrednost, dok više čvorova može imati istu oznaku (jednu ili više);
Veze - usmerene, imenovane i semantički relevantne veze između dva čvora (entiteta u stvarnosti), svaka ima identifikator, smer, tip, kao i početni i završni čvor (izvorište i odredište), a može imati i atribute (najčešće su kvantitativni poput težine, cene i sl.)

3.4. Obrada podataka u grafu

Procesiranje podataka u graf bazama podataka jedan je od važnijih faktora te tehnologije na koju treba obratiti pažnju. Naime, graf baze podataka koriste tzv. susednost bez indeksa (*Index-free adjacency*) za povezivanje čvorova u grafu.
To znači da je svaki čvor u grafu „fizički“ povezan sa svojim susednim čvorovima, odnosno u sebi sadrži vezu na druge čvorove.

3.5. Pronalaženje uzorka u grafu

Kao što je već spomenuto, popularnost graf baza podataka i algoritama zasnovanih na grafu u novije vreme sve više raste, posebno u naučnom okruženju. Zahvaljujući tome i visokom nivou ekspresivnosti u modeliranju složenih struktura podataka, broj mogućih primena za modeliranje podataka u obliku grafa raste.
Relativno brzo pretraživanje grafa moguće je zahvaljujući algoritmu pronalaženja uzoraka u grafu koji se koristi u izvođenju upita nad graf bazama podataka. Uzorkom u grafu smatra se izomorfna slika tog grafa, što predstavlja podgraf ciljnog grafa, pa se pronalaženje uzoraka u grafu (*graph pattern matching*) često naziva i tzv. problem izomorfizma podgrafa.

3.6. Poređenje graf baza podataka sa drugim bazama podataka

Graf baze podataka moguće je uporediti s relacionim, ali i drugim kategorijama NoSQL baza podataka s obzirom na različite karakteristike. Poređenjem relacionih i svih kategorija NoSQL baza podataka s obzirom na dubinu pretraživanja i veličinu podataka može se uočiti da su graf baze podataka jedini predstavnik NoSQL baza podataka koji može postići veću dubinu pretraživanja podataka (više nivoa) u odnosu na relacione baze podataka.

Međutim, kada se radi o sposobnosti skladištenja, skladišta tipa ključ-vrednost postižu bolje performanse i mogu skladištiti više podataka u odnosu na relacione i graf baze podataka.

4. SISTEMI ZA UPRAVLJANJE GRAF BAZAMA PODATAKA

Prilikom razvoja aplikacije korišćene su različite komponente sistema kako bi se poboljšale neke od funkcionalnosti, kao i omogućio brži i lakši razvoj. Sistemi za upravljanje graf bazama podataka imaju sličnu ulogu kao i relacioni DBMS sistemi opisani u prethodnim poglavljima.

Zastupljenost takvih sistema na tržištu u novije vreme sve više raste. Najčešće korišćeni graf DBMS sistem je Neo4j, čija popularnost konstantno raste. Slede ga OrientDB i Titan koji, zajedno sa Neo4j, ostvaruju najveći porast u korišćenju i zastupljenosti na tržištu. O Neo4j graf DBMS sistemu će biti više reči u nastavku. Osim njega, detaljnije će biti objašnjen i Rexster server korišćen u implementaciji aplikacije.

4.1. Neo4j

Neo4j je sistem otvorenog koda koji skladišti podatke u graf, čime ih je kasnije lakše dobavljati. Zasniva se na mrežno-orjentisanom modelu podataka sa osobinama u kojem su veze najvažniji objekti [4]. Implementiran je pomoću Java i Scala programskih jezika i pripada grupi starijih NoSQL baza podataka.

Neo4j pruža podršku za nativni pristup podacima, ali je podacima moguće pristupiti i putem Cypher i Gremlin upitnog jezika. Standardni jezik za upravljanje podacima u Neo4j bazi podataka je Cypher, pa je Cypher upite moguće definisati i izvršavati putem Neo4j desktop ili web aplikacije.

Zbog svoje robustnosti, skalabilnosti, opcione šeme i visokih performansi, Neo4j baze podataka moguće je koristiti i u velikim korporacijama [5].

4.2. Rexster

Rexster je server za graf baze podataka koji omogućava pristup graf bazi podataka putem HTTP/REST protokola i binarnog protokola RexPro koji omogućava slanje Gremlin skripti na udaljenu instancu Rexster servera [6].

Korišćenje HTTP protokola pruža podršku za svoje metode GET, POST, PUT i DELETE, fleksibilan model podataka koji je moguće proširiti brojnim ekstenzijama te procedure skladištene na serveru za brže izvršavanje Gremlin upita.

5. UPITNI JEZICI ZA GRAF BAZE PODATAKA

Povećanje mogućih oblasti primene graf baza podataka tokom nekoliko poslednjih decenija dovelo je do razvoja nekoliko različitih upitnih jezika za graf baze podataka. Ti upitni jezici su nastajali pod različitim uticajima: hipertekstualnih sistema u 80-im godinama prošlog veka, polustruktuiranih podataka i objektnih baza podataka u 90-im godinama prošlog veka, pa sve do najnovijeg uticaja semantičkog web-a i društvenih mreža [3].

Kao što je već spomenuto u prethodnim poglavljima, prilikom izvođenja upita koriste se različiti algoritmi obilaska grafa koji traže zadani uzorak.

Opšta šema obilaska u svakom od tih algoritama je da obilazak počinje u jednom čvoru iz kojeg se on nastavlja preko veza tog čvora prema ostalim čvorovima grafa. Putem posećenih veza obilazak se nastavlja prema posećenim čvorovima koji imaju neposećene veze i tako dugo dok se ne obiđu sve veze grafa.

5.1. Osnovni algoritmi grafa

Iako postoji puno algoritama obilazaka grafa, u ovom radu će se spomenuti i ukratko objasniti jedni od najpoznatijih i najkorišćenijih algoritama: Pretraga grafa po dubini (*Depth-first search*, u nastvku teksta DFS); Pretraga grafa po širini (*Breadth-first search*, u nastvku teksta BFS); Dijkstra algoritam

5.2. Upiti nad graf bazama podataka

Zbog širenja oblasti primene graf baza podataka tokom vremena, razvijeni su brojni upitni jezici za pretraživanje i upravljanje grafovima s različitim atributima i strukturama.

Povećana heterogenost i veličina podataka skladištenih u graf bazama podataka stvorila je potrebu za nativnim pristupom bazi podataka.

Savremeni koncept upita koji upravljaju grafom na globalnom nivou označava da se ograničenja i pravila nad čvorovima i vezama u grafu mogu definisati istovremeno nad skupom čvorova i veza koji čine traženi objekat, odnosno podgraf posmatranog grafa, za razliku od dosadašnjeg načina definisanja nad svakim pojedinačnim čvorom ili vezom u više iteracija.

6. ZAKLJUČAK

Graf baze podataka dobijaju sve veću pažnju kod korisnika, pre svega zahvaljujući svojim karakteristikama pojašnjenim u prethodnim poglavljima ovog rada.

Njihovim korišćenjem u različitim područjima moguće je skladištiti kompleksne podatke nad kojima se kasnije mogu izvršavati upiti na više nivoa, koji će vraćati rezultate u konačnom vremenu, što za relacione baze podataka nije uvek slučaj.

Njihovim poređenjem može se zaključiti kako je u slučaju povećanja složenosti i količine podataka u bazi podataka standardna devijacija vremena izvođenja upita nad tim podacima kod graf baza podataka mnogo manja u odnosu na relacione baze podataka.

Osim toga, često je skladištenje podataka u obliku grafa mnogo prirodnija nego njihovo skladištenje u redove tabela, jer je graf model podataka fleksibilniji. Neki autori smatraju graf baze podataka relacionim bazama podataka sledeće generacije, budući da su uspele preuzeti i proširiti neke prednosti relacionih baza poput matematičke algebre i razumljivosti modela podataka, ali i popraviti i zameniti njihove uočene nedostatke.

U radu su prikazani osnovni koncepti grafova koji čine model podataka graf baza podataka, ali i pojmovi poput pronalaženja uzoraka u grafu i obilaska grafa koje je potrebno razlikovati. Detaljnije su pojašnjeni najčešće korišćeni algoritmi xpronalaženja uzoraka i obilaska grafa. Ti su algoritmi samo jedan od faktora koji utiču na efikasno izvođenje upita na graf bazama podataka. Upiti nad graf bazama podataka mogu se izvoditi putem nekoliko upitnih jezika, od kojih su danas najzastupljeniji Cypher i Gremlin.

Poređenjem njihovih performansi u radu je pokazano kako Cypher ostvaruje bolje performanse u dobavljanju podataka, ali se ta razlika smanjuje s povećanjem složenosti podataka koje je potrebno dobaviti, jer ih Cypher nakon dobavljanja mora konvertovati u traženi oblik.

Osim teorijskog, ovaj radi sadrži i praktični dio u kojem je napravljena aplikacija za rad s Neo4j bazom podataka te izvođenje Cypher, upitima nad njom. Svrha aplikacije bila je pokazati da je moguće napraviti jednostavnu web aplikaciju, koja će, pomoću određenih bibliotekam, na elegantan način biti povezana sa Neo4j bazom podataka. Ovim je pokazano da nije potrebno mnogo truda oko učenja Cypher jezika i njegove sintakse, ali isto tako je bitno poznavanje njegovog izvršavanja.

Na osnovu analiziranih osobina graf baza podataka može se zaključiti kako će njihova zastupljenost na tržištu i u budućnosti nastaviti svoj rast zahvaljujući stalnom ulaganju u njhov razvoj, ali i povećanoj potrebi upravljanja složenim podacima.

7. LITERATURA

[1] Darwen H. (2012) An Introduction to Relational Database Theory. Peterlee, UK. Ventus Publishing ApS

[2] Robinson I., Webber J., Eifrem E. (2015) Graph Databases (2.izd) Neo Technology Inc. Sebastopol: O'Reilly Media Inc.

[3] Wood P.T. (2012) Query Languages for Graph Databases (1.izd) . London: Department of Computer Science and Information Systems. http:// users.dcc.uchile.cl/~pbarcelo/wood.pdf

[4] Angles R. (2012) A Comparison of Current Graph Database Models. http:// campuscurico.utalca.cl/~rangles/files/gdm2012.pdf

[5] Neo4j - Relational Databases vs Graph Databases: A Comparison - http:// neo4j.com/developer/graph-db-vs-rdbms/

[6] Rodriguez M.A. (2015) The Benefits of the Gremlin Graph Traversal Machine. http://www.datastax.com/dev/blog/the-benefits-of-the-gremlin-graph-traversal- machine

Kratka biografija

Ivan Savić je rođen 07.07.1994. godine u Novom Sadu. Osnovnu školu „Branko Radičević“ završio je 2009. godine u Šidu. Tehničku školu „Nikola Tesla“ u Šidu završio je 2013. godine kao Đak generacije. Iste godine upisao se na Fakultet tehničkih nauka u Novom Sadu, na odsek Računarstvo i automatika. Nakon uspešno završenih osnovnih akademskih studija na smeru Računarske nauke i informatika postaje diplomirani inženjer 2017. godine. Iste godine upisuje master akademske studije na istoimenom fakultetu i smeru i polaže sve ispite predviđene planom i programom 2018. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.5
DOI: https://doi.org/10.24867/01BE23Oljaca

INTERNACIONALIZACIJA I LOKALIZACIJA WEB APLIKACIJE RAZVIJENE KORIŠĆENJEM ASP.NET CORE MVC FRAMEWORK

INERNATIONALIZATION AND LOCALIZATION OF WEB APPLICATION DEVELOPED USING THE ASP.NET CORE MVC FRAMEWROK

Marko Oljača, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U okviru rada je izvršena analiza lokalizacije i internacionalizacije u okviru Microsoft tehnologija. Kreirana je web aplikacija korišćenjem ASP.NET Core MVC frameworka, a u procesu pravljenja iste prezentovani su pojmovi lokalizacije i internacionalizacije. Pošto je .NET Core sam po sebi relativno novi koncept u okviru Microsoft tehnologija početni deo rada se bavi samim razvojem te tehnologije. Zatim je objašnjeno kako u okviru .NET Core korišćenjem MVC frameworka možemo da kreiramo Web aplikaciju. Na kraju se u radu objašnjavaju tehnologije i alati koji se koriste za internacionalizaciju i lokalizaciju.*

Ključne reči: *.NET Core, MVC .NET Core, Internacionalizacija .NET Core web aplikacije, Globalizacija .NET Core aplikacije*

Abstract – *In this thesis theme was deep dive into localization and internationalization analysis within Microsoft technologies. Web application was created using ASP.NET Core MVC framework, and though the process of making it concepts of localization and internationalization were presented. Since .NET Core itself is a relatively new concept within Microsoft technology in the early part of the work is explained development of this technology. Then explained how to create Web application using the MVC framework in .NET Core. Finally part is dedicated to explanation of technologies and tools used for localization and internationalization.*

Keywords: *.NET Core, MVC .NET core, Internatiolization .NET Core Web application, Globalization .NET Core Web application*

1. UVOD

U savremenom svetu informacionih tehnologija omogućeno je da se lako povezujemo sa ljudima iz različitih kultura, koji koriste različite jezike i generalno komuniciraju na različite načine.

S tim na umu zadatak koji se nameće prilikom razvoja aplikacije jeste da ih pravimo tako da mogu sa lakoćom da menjaju svoj izgled na taj način da je mogu koristiti korisnici iz različitih kultura.

Zadatak rada jeste dublja analiza koncepata lokalizacije i internacionalizacije u okviru *Microsoft* tehnologija. Cilj je da se napravi web aplikacija korišćenjem ASP.NET Core MVC *frameworka* a u procesu pravljenja iste da se prezentuju pojmovi lokalizacije i internacionalizacije.

2. POJAM KULTURE U INFORMACIONIM TEHNOLOGIJAMA

Kultura je skup pravila i podataka koja su specifična za kombinaciju jezika i geografskog područja koje kultura jedinstveno reprezentuje.

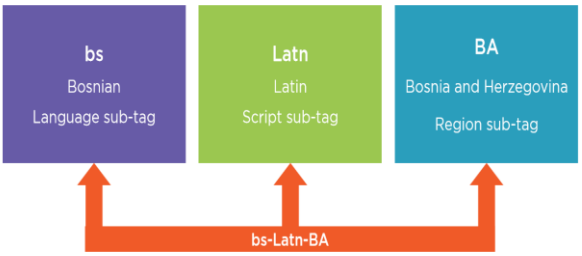
Locale = Culture u .NET okruženju local je sinonim za kulturu.

.Net Core ima podršku za različite kulture, tako da programeri ne moraju da se razumeju u pravila za specifične kulture.

Svaka kultura ima specifičan ***IETF Language Tag*** koji je jedinstveno obeležava. Primer jedno taga: **en-US**, kasnije će tag biti detaljno objašnjen.

IETF – Internet Egnineering Task Force su zaduženi za kreiranje i održavanje raznih internet standarda. Takođe su zaduženi za pojam internacionalizacije na internetu.

IETF Language Tag je identifikator jedne kulture koji je specificiran od strane **IETF.**



Slika 1. Struktura IETF *language tag-a*

2.1.POJAM KUTLURE u .NET

Skoro sve vezano za internacionalizaciju i18n u .NET Core se vrti oko klase **CultureInfo**. Ova klasa je postajala u .NET-u i portovana je na .NET Core i definisana je u System.Globalization modulu. Sve što nam je potrebno za

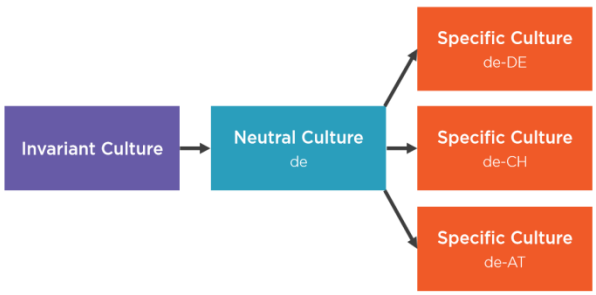
---

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Branko Milosavljević, red. prof.

korišćenje jedne kulture možemo da uradimo kreiranjem jedne instance pomenute klase.

CultureInfo ci = new CultureInfo(“en-US”);

Ono što je specifično za .NET Core je uvođenje dva pojma koji opisuju kulturu: **Culture** i **UI culture.** Oba pojma su intance CultureInfo klase, ali su odgovorni za različite aspekte internacionalizacije.
UI Culture je zadužena za
Prikazane tekstove
Culture je zadužena za
Format vremena i datuma
Format brojeva
Formatiranje vrednosti iskazanih u valutama
Redosled sortiranja
Velika i mala slova
Poređenje stringova
Ne postoji pravilo da **Culture** i **UI Culture** moraju imati istu vrednost, čak je ideja uvođenja dva različita pojma da bi se omogućila veća sloboda u razvoju. Odluka koje će **Culture** i **UI Culture** biti podržane i u kakvoj kombinaciji je procena koja se donosi u skladu sa biznis logikom.
Struktura samog taga ukazuje i na hijerarhiju kulture. Da bismo objasnili pojam hijerarhije kulture, moramo da uvedemo još par pojmova.
Neutralna kultura je kultura koja ima specificiran jezik, a nema specificiranu regiju ili pismo. Npr za Nemačku neutralna kultura je tag **de.** U samom C# postoji flag koji ukazuje na to da li je kultura neutralna. .IsNeutralCulture nam daje informaciju o tome da li je trenutna kultura neutralna kultura. Neutralna kultura nam omogućuje način da ipak podržimo veći broj kultura za specifičan jezik.
Invariant Cutlure je kultura bez taga, asocirana je za engleski jezik. Intanca invarijantne kulture se pravi na sledeći CutlureInfo.InvariantCulture. Ona nam može koristiti kada želimo da baratamo podacima koji nisu u relaciji sa bilo kojom specifičnom kulturom. Svaka kultura e hijerarhijski uvezana.



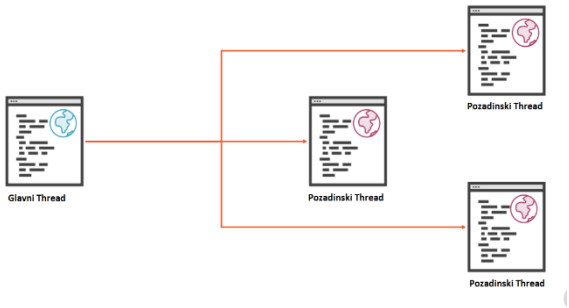
Slika 2. Hijerarhija kultura

2.2 KULTURE I MULTITHREADING

Svaka nit ima informaciju o kulturi pod kojom je ona pokrenuta, podaci o kulturi se mogu dobiti na sledeći način:

CultureInfo cultureInfo =
Thread.CurrentThread.CurrentCulture;

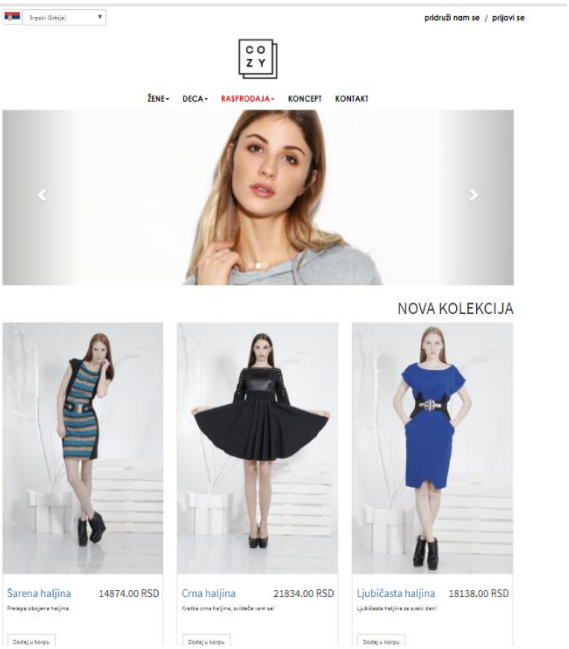
U slučaju kada se iz glavne niti kreira pozadinska nit, pozadinska nit nasleđuje kulturu od glavne niti. Svaka dalja promena kulture glavnog niti ili pozadinske niti neće uticati na kulturu niti kojoj nije promenjena kultura. Odnosno obe niti su posle kreiranja autonomne.
Nasleđivanje kulture prilikom kreiranja niti možemo zaobići postavljanjem default kulture za nit pre nego što je kreirana iz glavne niti. Property koji to omogućuje je DefaulThreadCurrentCutlure.



Slika 3. Kulture u nitima

3. SPECIFIKACIJA SISTEMA

Sistem koji se razvija u okviru ovog rada je web aplikacija za *on line* modnu prodavnicu. Nazvaćemo je *Master Fashion Store*. Ono što je pogodno za primer lokalizacije je to što u okviru rada sa prodavnicom je da moramo da obraćamo pažnju na rad sa različitim jezicima, apoenima i datumima. Na taj način se može slikovito prokazati svi aspekti tehnologije koje su potrebni za ovaj rad.



Slika 3. Izgled početne strane

4. IMPLEMENTACIJA SISTEMA

U ovom poglavlju ćemo se baviti implementacijom lokalizacije u referentnom projektu čime ćemo pokazati na koji način se implementira lokalizacija na projektima koji su implementirani korišćenjem MVC .Net Core-a *framework-a.*

Lokalizacija je u MVCu-u, kao i većina stvari implementirana upotrebom *middlewera*. Zato moramo da uvedemo pojam *middlewera.*

Šta je middleware?

Middleware je software koji se dodaje u tok aplikacije tako da je odgovoran za rad sa *requests* i *response*. Svaka komponenta ima dve opcije:

- Bira da li *request* da prosledi sledećoj komponenti u redu
- Može da uradi neki posao pre nego što se pozove sledeća komponenta u toku

U referentnom projektu smo koristili *Request Localization Middleware*. Izdvojićemo najvažnije karakteristike ove implementacije:

- Vezana je za životni vek jednog *HTTP requesta*. Kao što naziv middelwarea ukazuje *request* lokalizaijca je process koji se dešava u toku životnog veka jednog *HTTP requesta*.
- Preferencije klijenta diktiraju koja se kultura koristi
- Separacija *thredova*. Svaki HTTP *request* se izvršava u različitom *thredu,* tako da je moguće da paralelno dva korisnika gledaju sadržaj, a da je taj sadržaj prikazan u različitim kulturama.
- Nova instance lokalitazatora se kreira po *requestu.*

Prvo nam je potrebno da uključimo potrebne NuGet pakete, a zatim da uključimo sam *middelwhare* u aplikaciju. Od koda nam je potrebno samo ovo:

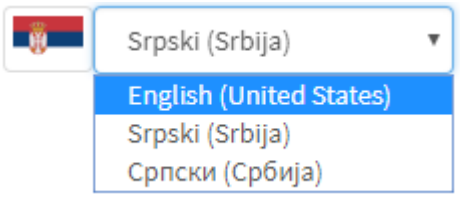
```
dependencies":
{
  ...,
  "Microsoft.AspNetCore.Localization":"1.1.1",
  ...
}
app.UseRequestLocalization(...);
```

Pošto smo dodali potrebne pakete, dodali smo i *middelwhare* koji nam je potreban za lokalizaciju. Sada nam ostaje da konfigurišemo njegovo ponašanje.

```
services.Configure<RequestLocalizationOptions>
(options =>
      {
  options.SupportedUICultures = new
List<CultureInfo>()
      {
          new CultureInfo("en-US"),
           new CultureInfo("sr-Latn-RS"),
           new CultureInfo("sr-Cyrl-RS")
      };
options.DefaultRequestCulture =
new RequestCulture("en-US");
});
```

Preko *Configure* metode i prosleđenog delegate možemo da konfigurišemo naš servis. Ovom prilikom smo definisali podržane kulture, kao i *default* kulturu za našu web aplikaciju.

Sa ovim koracima smo obezbedili da naša aplikacija podržava različite kulture i naveli smo koje kulture. Sada ćemo da vidimo na koji tačno način ćemo čuvati vrednost kulture i kako ćemo obezbediti da korisnik menja kulturu.



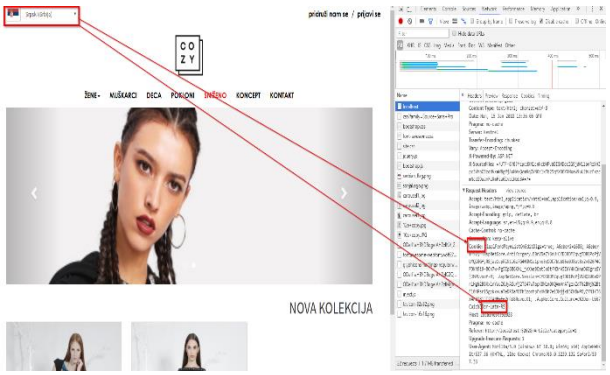
Slika 4. *Partial view* zadužen za izbor kutlure

Na slici 4. se vidi komponenta koja je zadužena za promenu kulture. Sa klijentske strane je to *Combobox* koji prikazuje podržane kulture. Ono što je bitno je da bi korisnicima bila jasnije koje kulture biraju imena država su napisana jezikom kulture koje biraju. Takođe sama zastava je lokalizovana. Sada ćemo da vidimo kontorler koji ima samo jednu metodu koja smešta aktivnu kulturu u *cookie* i tako nam obezbeđuje da u toku trajanja sesije korisnik koristi izabranu kulturu.

```
[HttpPost]
public IActionResult Set(string uiCulture, string
returnUrl, string returnParam)
                    {
          IRequestCultureFeature feature
HttpContext.Features.Get<IRequestCultureFeature>();

RequestCulture requestCulture =
 new RequestCulture(feature.RequestCulture.Culture,
   new CultureInfo(uiCulture));



string cookieValue =
          CookieRequestCultureProvider.MakeCookieVal
ue(requestCulture);
string cookieName =
          CookieRequestCultureProvider.DefaultCookieN
ame;
```

```
Response.Cookies.Append(cookieName, cookieValue);
    if (!String.IsNullOrEmpty(returnParam))
        {
            returnUrl = returnUrl + returnParam;
        }

        Return LocalRedirect(returnUrl);
}
```

Ovaj kod nam pokazuje kako se uzima izabrana kontrola od strane korisnika i smeštanje tog izbora u *cookie*.



Slika 5. Izbor kulture se smešta u cookie

4. ZAKLJUČAK

Master Fashion Store je studija slučaja na osnovu koje je prikazan proces lokalizacije i globalizacije. Osim same implementacije proces dizajna aplikacije koja u startu ima globalni pristup je ono što je bitno istaći. Internacionalizacija je jako ozbiljan proces kome se ponekad ne daje dovoljno pažnje u ranim fazama razvoja aplikacije, što dovodi do niza problema u kasnijim fazama. Ako smo svesni da je naša aplikacija internacionalnog karaktera, internacionalizacija mora biti uključena od samog starta razvoja.

Pošto je pristup arhitekturi razrađen uvidom u implementaciju je prikazano na koji način je u sklopu MVC .Net Core omogućeno realizovati internacionalizaciju. Videli smo alate i načine implementacije internacionalizacije. Bitan je i način projektovanja baze podataka koji u startu ima u vidu koji pojmovi se trebaju lokalizovati i kako.

Na osnovu iskustva u toku implementacije izdvojio bih par preporuka.

Preporuke za UI

- Zastava nije kultura

Jako je bitno da ne brkamo zastavu i kulturu, preporuka je da se naziv zemlje piše korišćenjem upravo pisma koje je specificiranu za tu kulturu. Nikako ne smemo pisati na engleskom nazive država.

- Autodetekcija jezika korisnika

Da bismo olakšali korisniku treba da vršimo predikciju jezika koji on koristi. U referentnoj aplikaciji se čita iz cookija, a ako nije podešeno setuje se u cookie-u.

- Jasan način da je korisnik svestan koji deo aplikacije je za promenu jezika

U referentnoj aplikaciji sve vreme je korisniku dostupna promena jezika, a zastava se koristi upravo da se korisniku pojasni čemu služi taj deo funkcionalnosti.

Prezentovan je način razmišljanja i alati koji su potrebni da se napravi aplikacija koja je spremna za svetsko tržište.

5. LITERATURA

[1] Microsoft dokumentacija, **https://docs.microsoft.com/en-us/globalization/software-internationalization**

[2] Gill Cleeren, Buidling your first ASP.NET Core Web Application, Pluralsight**,** **https://app.pluralsight.com/library/courses/aspdotnetcore-web-application-building/table-of-contents**

[3] Roland Gujit, Understanding ASP.NET Core Security, Pluralsight, **https://app.pluralsight.com/library/courses/asp-dot-net-core-security-understanding/table-of-contents**

[4] Admir Tuzović, ASP.NET Core Internationalization Deep Dive, Pluralsight**,**

https://app.pluralsight.com/library/courses/aspdotnetcore-internationalization-deep-dive/table-of-contents

Kratka biografija:

Marko Oljača rođen je 29.07.1983. godine u Novom Sadu. Osnovnu školu „Prva Vojvođanska Brigada“ završio je 1998. godine. Gimnaziju „Jovan Jovanović Zmaj“ u Novom Sadu završio je 2002. godine. Iste godine upisao se na Fakultet tehničkih nauka, odsek Računarstvo i automatika. Školske 2002/2003. godine upisao se na smer Računarske nauke i informatika. Osnovne studije završio 2012 godine godine. Zaposlen u *Schneider Eletric DMS NS* 2012 – 2017. Od 2017 do danas zaposlen u *Enjoying, Belgrade*, Serbia. Položio je sve ispite predviđene planom i programom Master studija na smeru Računarske nauke i informatika.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.3
DOI: https://doi.org/10.24867/01BE27Pejic

FUNKCIONALNO TESTIRANJE WIFI HALOW™ MAC SOFTVER STEKA KORISTEĆI VIRTUAL SYSTEM PLATFORM

FUNCTIONAL TESTING OF WIFI HALOW™ MAC SOFTWARE STACK IN SYSTEM CONTEXT USING VIRTUAL SYSTEM PLATFORM

Vladislav Pejić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Cilj rada jeste da predstavi okruženje i testove korišćene za funkcionalno testiranje Wi-Fi HaLow™ softver steka koristeći alat pod nazivom Virtual System Platform (VSP).*

Ključne reči: *WiFi HaLow™, IEEE 802.11ah, VSP, MAC*

Abstract – *Goal of this paper is to introduce environment and tests used for functional testing of Wi-Fi HaLow™ software stack using Virtual System Platform (VSP).*

Keywords: *WiFi HaLow™, IEEE 802.11ah, VSP, MAC*

1. UVOD

IEEE 802.11 WLAN (Wireless Local Area Network) je jedna od najpopularnijih bežičnih tehnologija. Ona koristi 2.4GHz i 5GHz frekvencijske opsege i ima dobru brzinu prenosa podataka, lako se postavlja i jeftina je. Međutim, korišćenje visoke frekvencije ograničava domet bežične mreže. Pored ograničenog dometa, prekomerna upotreba je dovela do zasićenja. Povećanjem broja uređaja koji koriste iste frekvencije dovešće do još većeg zasićenja mreže [1].

Naša želja za automatizacijom zahtevaće dosta uređaja koji će trebati da budu povezani. Ovi pametni uređaji (senzori, roboti, kontroleri), povezani međusobno, čine mrežu zvanu Internet of Things (IoT). Najlakši i najjeftiniji način povezivanja toliko uređaja je pomoću neke od bežičnih tehnologija. Kako bi se ovi uređaji međusobno povezali, neophodno je da mreža poseduje veliki domet, dobru brzinu prenosa podataka, može da poveže veliki broj stanica, ali i da ima malu potrošnju energije zbog baterijski napajanih uređaja. IEEE 802.11ah, poznat kao i Wi-Fi HaLow™, poseduje sve ove karakteristike.

Ovaj rad opisuje okruženje i testove korišćene za testiranje Wi-Fi HaLow™ softer steka pomoću Virtual System Platform alata kompanije Cadence®.

2. VIRTUAL SYSTEM PLATFORM

Virtual System Platform je deo Cadence® System Development Suite-a. VSP omogućava razvoj softvera, funkcionalnu verifikaciju, analizu sistema i njegovu optimizaciju pre nego što su hardverski moduli u potpunosti razvijeni. Umesto potpuno razvijenih hardverskih modula koriste se virtuelni prototipovi [2].

VSP omogućava upotrebu hardverskih modela, ali i brz i automatizovan način za kreiranje virtuelnih hardverskih modela. Automatizovan generator koda čita IP-XACT ili tekstualni ulaz i kreira TLM (Transaction-Level Modeling) 2.0 templejt. U VSP-u, biblioteka TLM IP modela je dostupna za korišćenje u virtuelnim prototipovima. Kreirani TLM model je SystemC model i lako ga je modifikovati i prilagoditi sopstvenim potrebama.

VSP poseduje grafičko okruženje koji omogućava softversko i hardversko debagovanje. Okruženje sadrži breakpoint-ove, simulaciju korak po korak, iščitavanje memorije kao i grafičko predstavljanje softverskih i hardverskih signala. Jedna od glavnih osobina VSP alata je istovremeno debagovanje softvera i hardvera.

Prednosti korišćenja virtuelnih hardverskih prototipova je ta da je moguće početi sa razvojem softvera pre završetka RTL ili FPGA modula. Takođe omogućava lakše debagovanje kompleksnih modela koji sadrže hardverske i softverske delove.

Dizajn koji može da se koristi kao ulaz u VSP može biti:

- SystemC TLM 1.0 ili TLM 2.0,
- Legacy RTL: Verilog®, VHDL, ili SystemVerilog,
- C/C++/asembler.

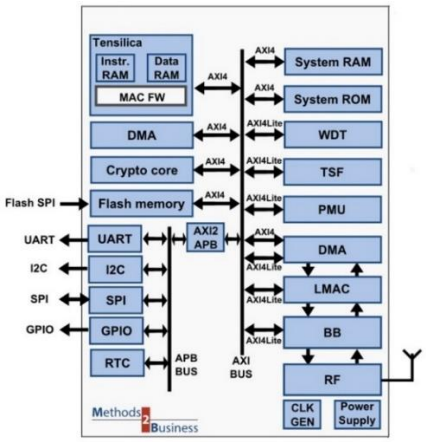
3. STRUKTURA SISTEMA

Slika 1. prikazuje celo Wi-FI HaLow™ SoC (System-On-Chip) rešenje. SoC se sastoji od Wi-FI HaLow™ MAC sistema, fizičkog sloja i RF radia. Fizički sloj i RF radio nisu bili deo sistema koji se testirao, ali će biti deo finalnog rešenja. Hardverska platforma je ista i za stanicu i za Access Point. Wi-FI HaLow™ MAC sistem kombinuje hardversko i softversko rešenje. Vremenski kritične funkcije su implementirane u hardveru, a viši protokoli su implementirani u softveru. Hardverska platforma se sastoji od:

- Tensilica® Fusion DSP,
- DMA (Direct Memory Access),
- LMAC (Lower Medium Access Control),

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji je mentor bio dr Rastislav Struharik.

- PMU (The Power Management Unit) i
- Kripto jezgro.



Slika 1. *Wi-FI HaLow™ SoC*

Tensilica® Fusion DSP se koristi za pokretanje Wi-Fi HaLow™ MAC softver steka. DSP je dizajniran da optimizuje potrošnju energije i procesorsku moć dok nudi mnogo veću fleksibilnost nego tipični procesor kao što je ARM jezgro.

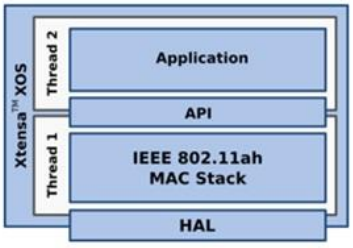
DMA je modul koji se koristi za brzi transfer podataka od i do memorije, bez potrebe za intervencijom procesora.

LMAC predstavlja donji nivo MAC steka. Ovaj modul je implementiran u hardveru i sastoji se od nekoliko manjih modula (kontrolna jedinica, modul za pristup kanalu, modul za agregaciju/deagregaciju, CRC generator). Projektovan je da izvršava funkcionalnosti nižeg nivoa kao što su: kontrola pristupa kanalu, filtriranje paketa, agregacija... Više o ovim funkcijama može se pronaći u [3] i [4].

PMU modul vodi računa o potrošnji energije celog sistema. Ovaj modul uključuje i isključuje delove sistema kako bi se smanjila potrošnja energije. U vreme testiranja, ovaj modul je bio u procesu dizajniranja.

Kripto modul je zadužen za enkripciju i dekripciju paketa. U vreme testiranja, modul nije bio dostupan i Tensilica® DSP je bio zadužen za enkripciju/dekripciju paketa.

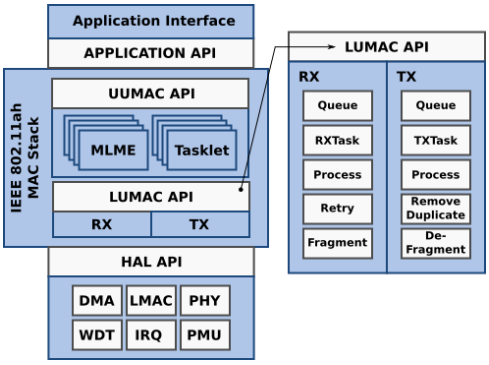
Viši protokoli Wi-Fi HaLow™ MAC steka su implementirani u softveru. Xtensa™ XOS je korišćen kao operativni sistem a softver stek je podeljen u dva treda: aplikativni i MAC stek tred (slika 2.). Komunikacija između tredova se obavlja pomoću API funkcija.



Slika 2. *Arhitektura Wi-Fi HaLow™ MAC softver platforme*

Stanica i Access Point imaju istu arhitekturu softver steka, ali je implementacija funkcionalnosti drugačija. Koristeći makro direktive, vrši se izbor za čega će se softver stek koristiti: za stanicu ili Access Point.

MAC softver stek je konstruisan pomoću modularne softverske arhitekture koju je lako proširiti i dodati nove funkcionalnosti. Struktura MAC softver steka je prikazana na slici 3.



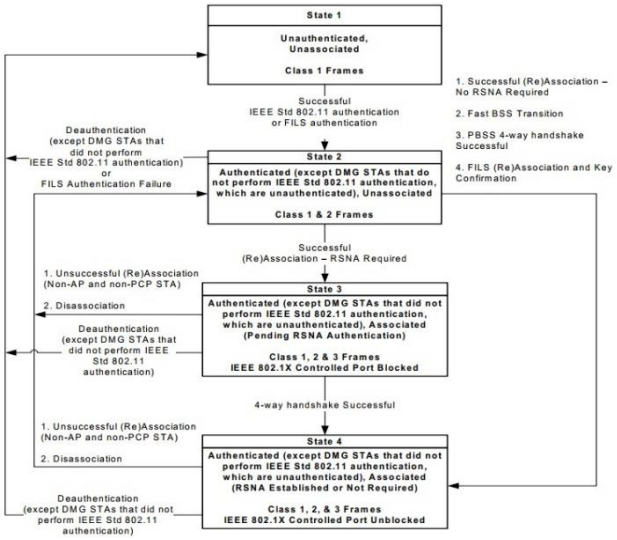
Slika 3. *Struktura Wi-Fi HaLow™ MAC softver steka*

Wi-Fi HaLow™ MAC softver stek je kompleksan deo softvera. Viši protokoli koji komuniciraju sa višim nivoima u mreži su implementirani u UUMAC (Upper Upper MAC) dok su funkcije koje komuniciraju sa hardverskim delom sistema u nižim nivoima MAC-a (LMAC), implementirane u LUMAC (Lower Upper MAC).

4. FUNKCIONALNOSTI KOJE SU TESTIRANE

4.1. Konekcija

Da bi stanica mogla da šalje podatke unutar bežične mreže, neophodno je da prvo prođe proces konekcije. Slika 4. prikazuje odnos između stanja stanice tokom konekcije i usluga koje su joj dostupne.



Slika 4. *Odnos stanja stanice i dostupnih usluga [3]*

Stanica prolazi kroz 4 stanja tokom konekcije:

- Stanje 1 - ovo je inicijalno stanje stanice. Stanica nije ni autentikovana niti asocirana.
- Stanje 2 - stanica je uspešno autentikovana, ali još nije asocirana.

- Stanje 3 - stanica je uspešno autentikovana i asocirana i čeka na 4-way handshake ako Access Point zahteva upotrebu enkripcije. Ako ne zahteva, stanica je uspešno konektovana i prelazi u stanje 4.
- Stanje 4. - stanica je uspešno konektovana.

U svakom od stanja, stanica može da šalje samo odgovarajuće pakete. Postoje 3 klase paketa: klasa 1, klasa 2 i klasa 3. U stanju 1. stanica može da šalje samo pakete klase 1, u stanju 2. pakete klase 1 i 2 i u stanjima 3 i 4 pakete svih klasa.

4.2. Diskonekcija

Ako postoji potreba da se stanica diskonektuje sa Access Point-a, stanica će poslati Deauthentication paket. Posle slanja ovog paketa, stanica se vraća u stanje 1. Access Point može u bilo kom trenutku da diskonektuje konektovanu stanicu na isti način, slanjem Deauthentication paketa.

4.3. Prenos podataka

Prenos podataka unutar bežične mreže je jedna od najvažnijih funkcionalnosti koje će biti testirane. Wi-Fi HaLow™ stanica ili Access Point, mogu da šalju podatke koristeći PV0 ili PV1 QoS Data pakete. Postoje dva tipa PV1 paketa: tip 0 i tip 3. Razlika je u formatu adresa.

4. Intenzivan prenos podataka

Intenzivno slanje podataka je slično regularnom slanju. Podaci se mogu slati pomoću istih paketa kao i kod regularnog slanja. Jedina razlika je što se kod intenzivnog slanja, podaci šalju jedan za drugim, najbrže moguće.

4.5. RTS/CTS mehanizam

RTS (Request to Send)/CTS (Clear to Send) mehanizam se koristi za zaštitu Data paketa namenjenih za slanje. Kada se mehanizam koristi, stanica će poslati RTS paket i čeka odgovor od primaoca RTS i Data paketa. Očekivani odgovor je NDP CTS i ako je primljen stanica će poslati Data paket. Ako NDP CTS nije primljen, stanica će ponovo pokrenuti mehanizam. Da bi se odredilo koji Data paketi treba da budu zaštićeni ovim mehanizmom, koristi se do11RTSThreshold parametar. Ako je dužina Data paketa veća od vrednosti ovog parametra, koristitiće se RTS/CTS mehanizam. Ako je vrednost ovog parametra jednaka 0, svaki Data paket će biti zaštićen.

4.6. RSNA konekcija

RSNA konekcija je sloj iznad regularne konekcije. Ovaj tip konekcije se koristi kada je uključena enkripcija unutar bežične mreže. RSNA konekciju inicira Access Point nakon uspešne asocijacije.

4.7. Fragmentacija

Fragmentacija je procedura tokom koje se veliki paket deli na više manjih. Koristi kada je potrebno povećati pouzdanost prenosa podataka. Manji paketi imaju veću šansu uspešnog slanja. Defragmentacija je obrnuti proces. Manji paketi se spajaju i originalni paket se rekonstruiše. Da bi se odredilo koji paket treba da se fragmentiše, koristi se dot11FragmentationThreshold. Ako je dužina korisnih informacija koje paket nosi veća od ovog parametra, paket će biti fragmentisan.

4.8. Agregacija

Prilikom svake transmisije u IEEE 802.11 bežičnim mrežama samo se deo transmisionog vremena koristi za prenos korisnih informacija. Ostatak vremena čini prenos raznih zaglavlja, razmak izmedju paketa i potvrda o uspešnom prijemu paketa. Kako bi se povećala efikasnost, koristi se agregacija. Više paketa se spajaju u jedan i tako se šalju. Na prijemnoj strani se takav paket deli na originalne pakete, koji se dalje obrađuju pojedinačno.

4.9. Detekcija duplikata

U bežičnim mrežama javlja se dosta grešaka prilikom slanja i prijema paketa. Zbog ovoga je važno da postoji način da se filtriraju paketi i odbace duplikati. U IEEE 802.11 bežičnim mrežama detekcija duplikata se obavlja na osnovu rednog broja (Sequence Number) i Retry polja (samo za PV0 pakete). Svaki paket sadrži redni broj koji mu dodeljuje pošaljilac. Primalac čuva najmanje redni broj poslednjeg paketa koji je primio. Kada je paket retransmisija, Retry polje je postavljeno na 1. Na strani primaoca, ako je Retry polje postavljeno na 1, poredi se redni broj primljenog paketa sa rednim brojem poslednjeg primljenog paketa. Ako su jednaki paket se odbacuje kao duplikat. Za PV1 pakete, pošto nemaju Retry polje, redni broj svakog paketa se poredi.

4.10. Kontrola prenosa podataka (Flow control)

Kontrola prenosa podataka (Flow control) je način na koji jedna stanica može da naredi drugoj da joj ne šalje podatke određeni vremenski period, naznačen u naredbi.

4.11. Provera Duration polja

Duration polje je polje unutar MAC zaglavlja PV0 paketa. Ovo polje služi za zaštitu paketa koji slede nakon paketa sa Duration poljem (paketi koji su odgovor na taj paket ili su predviđeni za slanje u istom TXOP-u). Vrednost ovog polja predstavlja vreme nakon prijema paketa sa Duration poljem, tokom kojeg je bežični medijum zauzet.

4.12. PV1 paketi

Jedna od novih karakteristika u Wi-Fi HaLow™ mrežama je nova grupa paketa pod nazivom PV1 paketi. Ovi paketi imaju drugačije MAC zaglavlje nego PV0 paketi koji su do sada korišćeni. Da bi se povećala efikasnost prilikom slanja, MAC zaglavlje ovih paketa je kraće od zaglavlja PV0 paketa. PV1 paketi nemaju Duration polje, format adresa je drugačiji, a takođe nemaju ni Retry polje. Za PV1 pakete ne koristi se RTS/CTS mehanizam.

4.13. Senzorski mod stanice

Stanica u Wi-Fi HaLow™ može da radi u dva moda, senzorski i nesenzorski mod. Senzorska stanica je stanica koja ima sledeće karakteristike:

- Ograničena količina saobraćaja (TXOP = 0, senzorske stanice mogu da pošalju samo jedan paket u TXOP).
- Uređaj sa baterijskim napajanjem.
- Duration polje u PV0 paketima jednak je 0 za senzorske stanice.

4.14. Skeniranje

Stanica koristi skeniranje da proveri da li postoje Access Point-ovi u njenoj blizini. Postoje dva tipa skeniranja:

- Pasivno skeniranje - stanica osluškuje kanal za S1G Beacon pakete, koje Access Point periodično šalje.
- Aktivno skeniranje - stanica šalje Probe Request i čeka Probe Response od Access Point-a.

5. STRUKTURA TESTOVA I OKRUŽENJA

Platforma koja je korišćena za testiranje Wi-Fi HaLow™ MAC softver steka je predstavljena na slici 5. Platforma se sastoji od dva Wi-Fi HaLow™ MAC steka (sadrže model fizičkog sloja) i modela bežičnog medijuma.

Slika 5. *VSP platforma korišćena za testiranje*

Model bežičnog medijuma ima mogućnost da simulira smetnje u kanalu. Može biti konfigurisan da ošteti određene sekvence paketa, određene pakete, određene sekvence bajtova unutar paketa ili nasumično odabrane pakete. Konfiguracija kanala se vrši pre početka simulacije pomoću TCL skripte, postavljanjem određenih ulaznih signala.

Jedan od zadataka ovog rada je bio razvoj okruženja za automatizovano regresiono testiranje i generisanje izveštaja. Ovo je postignuto pomoću shell skripte i Makefile. Shell skripta je dizajnirana da kreira izveštaj, odredi SVN revizije određenih delova sistema i pozove Makefile. Skripta će generisati dva izveštaja: u jednom će se nalaziti svi pokrenuti testovi, dok u drugom samo oni testovi koji nisu uspešno završeni. Imena regresionih izveštaja će biti sledećeg formata: regression_report_log_yyyymmdd_serial_number i regression_report_log_yyyymmdd_serial_number_failed_tests. Serijski broj predstavlja redni broj izveštaja za taj datum. Nakon kreiranja izveštaja poziva se Makefile i počinje simulacija.

Makefile ima sledeće ulazne parametre: lista testova koje treba izvršiti, regresioni tajmaut i ime regresionog izveštaja. Makefile će rekompajlirati softver pre svakog testa sa odgovarajućim aplikacijama i ulaznim parametrima. Ime testa i rezultati kompajliranja i izvršavanja testa biće upisani u izveštaj nakon završetka testa. Primer izveštaja je prikazan na slici 6.

```
**********SOFTWARE REGRESSION REPORT LOG**********
Regressions started on: 2017.03.29
Regressions started at: 16:51:27
trunk SVN revision: 5396
UM Library SVN revision: 133
Firmware SVN revision: 5396
LMAC SVN revision: 5396
DMA SVN revision: 5396
PHY Adapter SVN revision: 5396
**********

001_connect_1:
- Compiling firmware: OK
- Running test: Passed

001_connect_2:
- Compiling firmware: OK
- Running test: Passed

001_connect_3:
- Compiling firmware: OK
- Running test: Passed

001_connect_5:
- Compiling firmware: OK
- Running test: Passed
```

Slika 6. *Primer regresionog izveštaja*

Za testiranje Wi-Fi HaLow™ MAC softver steka korišćeni su testovi bazirani na događajima (event-ovima). Aplikacija je korišćena i kao drajver i čeker. Okidači događaja su postavljeni unutar MAC softver steka. Nihov broj i mesto unutar softvera birano je tako da ne menjaju ni vreme izvršavanja ni funkcionalnost MAC softver steka. Testovi su podeljeni u grupe prema funkcionalnostima koje testiraju. Unutar svake grupe, testovi su posebno numerisani. Svaki test ima svoju fasciklu u kojoj se nalaze: aplikacije za stanicu i Access Point, TCL skripta za podešavalje kanala i dokument sa ulaznim parametrima.

6. ZAKLJUČAK

Primarni zadatak ovog rada je bio testiranje Wi-Fi HaLow™ MAC softver steka koristeći VSP. Bilo je potrebno razviti okruženje za automatizovano regresiono pokretanje testova i generisanje izveštaja a zatim i testove za testiranje određenih funkcionalnosti. Svi zadaci su uspešno realizovani. Preko 200 testova je razvijeno i preko 150 grešaka i problema je pronađeno.

Ovi testovi su jedan deo testiranja Wi-Fi HaLow™ MAC softver steka. Razvijeni su da testiraju određene situacije u idealnim uslovima. Sledeći korak će biti prilagođavanje ovih testova za FPGA testiranje u kojem se prijem paketa ne može predvideti niti garantovati. U planu je i da se razvije automatizovano okruženje za regresiono testiranje na FPGA pločama.

7. LITERATURA

[1] W. Sun, M. Choi, and S. Choi, “IEEE 802.11ah: A Long Range 802.11 WLAN at Sub 1 GHz,” Journal of ICT Standardization, vol. 1, no. 1, pp. 83–108, 2013.

[2] https://www.cadence.com/content/cadence-www/global/en_US/home/tools/system-design-and-verification/software-driven-verification (pristupljeno u septembru 2018)

[3] IEEE Standard for Information technology - Telecommunications and information exchange between systems Local and metropolitan area networks - Specific requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11™-2016, IEEE Standards Activities Department, IEEE, 2016

[4] IEEE Standard for Information technology - Telecommunications and information exchange between systems Local and metropolitan area networks - Specific requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Amendment 2: Sub 1 GHz License Exempt Operation, IEEE Std 802.11ah™-2016, IEEE Standards Activities Department, IEEE, 2016

Kratka biografija:

Vladislav Pejić rođen je 9.5.1991. u Sremskoj Mitrovici. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu, na Katedri za elektroniku, u novembru 2015. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.317
DOI: https://doi.org/10.24867/01BE28Mirkovic

PROJEKTOVANJE MERNO-INFORMACIONOG SISTEMA ZA UDALJENA MERENJA ZASNOVANOG NA MIKROC I LABVIEW OKRUŽENJU

DESIGN OF MEASUREMENT-AND-INFORMATION SYSTEM FOR REMOTE MEASUREMENTS BASED ON MICROC AND LABVIEW ENVIRONMENT

Stefan Mirković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu predstavljen je koncept merno-avizicionog sistema za udaljena merenja i akviziciju parametara kvaliteta životne sredine i uslova radne sredine, kao i prateći softver za daljinski nadzor i upravljanje, snimanje podataka i analizu rezultata. Sistem je baziran na mikrokontroleru iz PIC32 familije i senzorskih modula koji služe za merenje parametara okoline. Sistem je zamišljen tako da se lako može proširiti broj senzora koji komuniciraju sa centralnom jedinicom. Za kontrolu i prikupljanje informacija sa udaljene lokacije putem TCP/IP protokola razvijena je LabVIEW PC aplikacija (virtualni instrument).*

Ključne reči: *Merenje; Senzori; LabVIEW; mikroC; TCP; Životna sredina*

Abstract – *In this paper concept of measurement and acquisition system for remote measurement and acquisition of environmental parameters is presented, as well as accompanying software for remote monitoring and controle, recording and analysis of results. The system is based on PIC32 family microcontroller and sensor modues that serves for environmental measurements. The system is designed so that is easy to expand the numbers of sensors that communicate with central unit. LabVIEW application (virtual instrument) was developed to control and collect information from a remote location, based on TCP/IP protocol.*

Keywords: *Measurements; Sensors; LabVIEW; mikroC; TCP; Environment*

1. UVOD

Praćenje kvaliteta životne sredine predstavlja sistematsko merenje i ispitivanje parametara kao i ocenjivanje indikatora stanja i zagađenja životne sredine. Na osnovu dostupnih podataka sa mernih mesta o stanju životne sredine dobija se jasan uvid u promene kvaliteta i kvantiteta životne sredine, emisije zagađujućih materija i korišćenje prirodnih resursa.

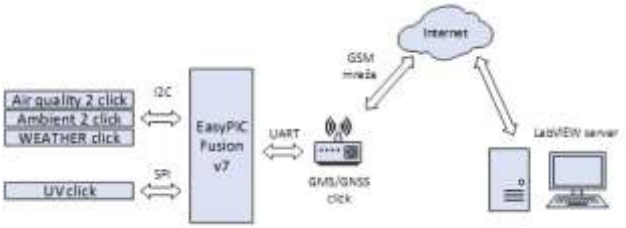
U cilju razvoja sistema za merenje i akviziciju parametara kvaliteta životne sredine i uslova radne sredine, razvijen je koncept merno-avizicionog sistema, pa i prateći softver za daljinski nadzor i upravljanje, snimanje podataka i analizu rezultata. Sistem je baziran na mikrokontroleru iz PIC32 familije, senzorskih modula, GSM modula za povezivanje na internet i LabVIEW aplikaciji koja služi kao TCP server, jer je odabran TCP protokol za razmenu podataka. Osnovne karakteristike GSM mreže su globalna rasprostranjenost i dostupnost, ali i značajne mogućnosti kada je u pitanju prenos mernih podataka. GPRS (General Pocket Radio Service) je trenutno dostupan u skoro svim GSM mrežama. GPRS predstavlja prenos podataka zasnovan na Internet protokolu. Zahvaljujući porastu brzine prenosa sve veći značaj ima plaćanje prema količini prenetih podataka u odnosu na vreme provedeno na mreži. Mikroprocesorski bazirani senzori i merni uređaji opremljeni sa GPRS predajnikom na taj način mogu da budu stalno povezani na mrežu a podatke da šalju po potrebi.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Platon Sovilj, vanr. prof.

2. OPIS SISTEMA

Sistem čine nekoliko nezavisnih celina: senzorski deo, mikrokontroler, deo za komunikaciju i softver. Senzorski moduli su povezani sa razvojnim sistemom EasyPIC Fusion v7, na kom se nalazi i mikrokontroler. Pored toga, na razvojni sistem povezan je i GSM modem za bežičnu komunikaciju putem mobilnog interneta. Kada se uspostavi TCP konekcija između GSM modema i LabVIEW softvera, gde je GSM modem klijent, mikrokontroler u određenim vremenskim intervalima koje diktira LabVIEW server šalje podatke o izmerenim veličinama ka serveru. LabVIEW te podatke prihvata, procesuira, prikazuje na ekranu računara i smešta u memoriju računara. Pauze između dva merenja određuje server, odnosno korisnik koji unosti vrednost pauze u sekundama.

Pri realizaciji ovog merno-akvizicionog sistema korišćeni su senzorski moduli koji podržavaju I^2C ili SPI komunikaciju. Mikrokontroler komunicira sa svim modulima, prikuplja informacije o merenim veličinama, formatira u određeni oblik, beleži u jedan niz karaktera i u određenim vremenskim trenucima putem TCP/IP protokola pomoću GSM modula šalje na udaljenu lokaciju ka LabVIEW serveru. Svi ovi moduli integrisani su pomoću razvojnog sistema EasyPIC v7 Fusion. Implementirani mikrokontroler je PIC32MZ2048ECH144 model sa 144 pina, integrisan na pločici koju je razvila MikroElektronika. Senzorski deo se sastoji od četiri senzorska modula: Air quality 2 click, Ambient 2 click, WEATHER click, UV click.



Slika 1. *Blok šema sistema*

3. HARDVER

Air quality 2 click sastoji se od IAQ-CORE C (Ams) senzorskog modula i dodatne elektronike za prilagođenje naponskih nivoa komunikacionih pinova. IAQ-CORE C senzorski modul koristi se za estimaciju CO_2 i TVOC (ukupna isparljiva organska jedinjenja) ekvivalenata. Baziran je na MEMS metal-oksid tehnologoiji. Opseg osetljivosti ovog modula je 450 ppm do 2000 ppm CO_2 ekvivalenta i 125 ppb do 600 ppb TVOC ekvivalenta. Komunicira sa mikrokontrolerom putem I2C protokola.

Ambient 2 click sastoji se od OPT3001 (Texas Instruments) senzorskog modula. Ovaj senzor se koristi za merenje intenziteta svetlosti vidljivog spektra (iluminaciju). Njegov spektralni odziv približno odgovara spektralnom odzivu ljudskog oka (približno normalna raspodela), što je i pogodno u slučaju praćenja životne sredine i meteoroloških parametara. Radi na principu pretvaranja foto-struje sa fotoosetljivog elementa u napon pomoću internog pojačavača i prosleđivanjem tog napona na interni AD konvertor. Opseg osetljivosti ovog senzora je 0.01 lx do 83 000 lx. Komunicira sa mikrokontrolerom putem I2C protokola.

WEATHER click sastoji se od BME280 (BOSCH) senzorskog modula. Ovaj modul ima mogućnost merenja ambijentalne temperature, atmosferskog pritiska i relativne vlažnosti vazduha. Unutar memorije ovog modula nalaze se fabričke kalibracione konstante koje služe za kompenzaciju očitanih vrednosti. Nakon očitavanja temperature, vrši se korekcija te vrednosti prema tim konstantama, kako bi se u sledećem koraku mogao odrediti uticaj izmerene temperature na očitan pritisak i vlažnost i takođe izvršiti korekcija. Operativni opseg je -40 °C do +85 °C, 300 hPa do 1100 hPa i 0 % RH do 100 % RH. Komunicira sa mikrokontrolerom putem I_2C protokola.

UV click sastoji se od ML8511 (Lapis Semiconductor) senzorskog modula i MCP3201 12-bitnog (SAR) AD konvertora (Microchip). Koristi se za merenje intenziteta UV zračenja. Glavni deo ML8511 modula je fotoosetljivi element čija se foto-struja pretvara u napon pomoću internog pojačavača, koji se zatim prosleđuje na ulaz MCP3201. Spektralni odziv ML8511 modula je najbolji kod talasnih dužina u okolini λ=365 nm, i izlazni napon ovog modula je približno lineran u odnosu na promenu intenziteta UV zračenja. Komunikacija MCP3201 AD konvertora sa mikrokontrolerom je ostvarena preko SPI protokola. Svi moduli se napajaju sa +3.3 V.

U ovom projektu korišćen je *GSM/GNSS click* modul sa MC60 (Quectel) GSM/GNSS modemom koji sa mikrokontrolerom komunicira preko UART protokola. MC60 je multi-funkcionalni modul koji integriše GNSS uslugu i quad-band GSM/GPRS uslugu preko mobilne mreže. Quad-band GSM/GPRS usluga može da radi na frekvencijama GSM 850 MHz, EGSM 900 MHz, DCS 1800 MHz, PCS 1900 MHz. Modul podržava i sisteme pozicioniranja i navigacije GPS, GLONASS, SBAS. Internet protokoli koje podržava ovaj modul su TCP, UDP, PPP, HTTP,FTP, NTP i PING. Nominalan napon napajanja kreće se u opsegu od 3.3 V do 4.6 V. Nominalna radna temperatura je od -35 ºC do +75 ºC. Ovaj modul ima slot za micro SIM karticu (podržane i 1.8 V i 3.0 V kartice), Bluetooth antenu (MC60 podržava Bluetooth 3.0) kao i dodatne pinove za povezivanje zvučnika i mikrofona.

Slika 2. *GSM/GNSS click modul*

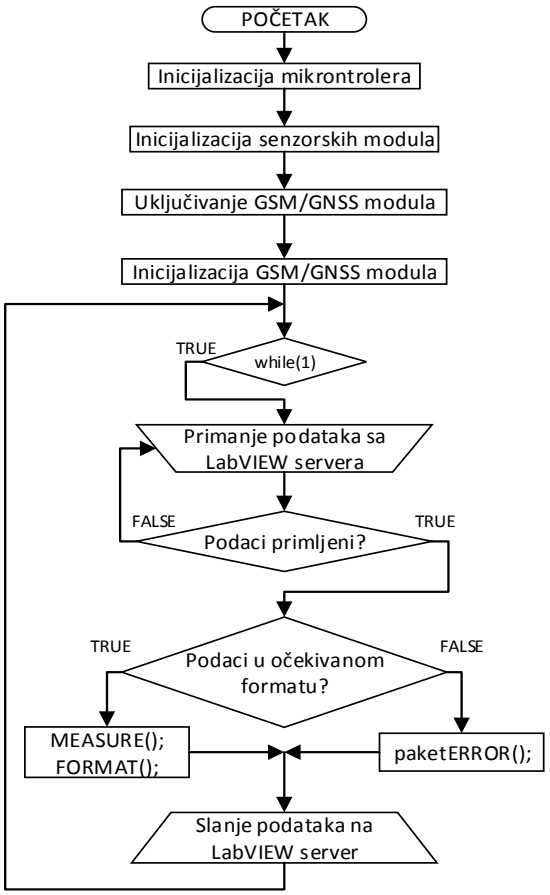
Za potrebe ovog projekta bitno je objasniti jednu od mogućnosti koju pruža ovaj PIC32 mikrokontroler a to je Peripheral Pin Select (PPS). PPS konfiguracija omogućava hardversko povezivanje određenog pina sa određenim modulom unutar mikrokontrolera. Drugim rečima, omogućeno je da se za određeni interni modul bira sa kojim će pinom biti povezan. Ovo naravno ne važi za sve module i za sve pinove, ali postoje tablice sa ponuđenim dozvoljenim kombinacijama prema kojima je moguće izvršiti takozvano mapiranje. Ovo važi kako za ulazne pinove, tako i za izlazne pinove.
Potreba za PPS konfiguracijom bila je kako kod UART, tako i kod SPI komunikacije. Ovde je RD2 pin konfigurisan kao MISO (SPI), a RD10 pin kao MOSI(SPI). SPI clock (SCK) ima svoj definisan pin i ne može da se "premosti" na neki drugi pin, i u ovom slučaju to je RD1 pin. Kod I^2C komunikacije je nemoguće mapirati pinove, odnosno definisano je koji je pin SCL (clock) a koji SDA (I/O). Ovde je izabran I2C modul koji je povezan sa RA14 (SCL) i RA15 (SDA) pinovima.

4. FIRMVER

Firmver koji izvršava mikrokontroler napisan je u programskom okruženju mikroC PRO for PIC32, razvijen od kompanije Mikroelektronika. mikroC PRO for PIC32 je programsko okruženje namenjeno za programiranje PIC32 serije mikrokontrolera u programskom jeziku C. Prednost ovog okruženja je veliki broj napisanih i testiranih biblioteka kao što su SPI, I2C, UART, CAN, ETHERNET, ADC, FFT, PWM itd. koje mnogo olakšavaju izradu firmvera. Ovo okruženje ima mnoge funkcije koje olakšavaju rad i sa TFT ekranima. Takođe podržava i hardversko debagovanje čime se olakšava nadgledanje rada celokupnog firmvera. Pored ovoga, implementirana je podrška i za FreeRTOS, gde može da se primeni i FreeRTOS operativni sistem pri programiranju mikrokontrolera.

Firmver mikrokotrolera počinje inicijalizacijom samog mikrokontrolera. Ovde se vrše sledeće operacije: definisanje svih pinova mikrokontrolera kao digitalne I/O, inicijalizacija dva interna UART1 modula, inicijalizacija

I2C1 modula mikrokontrolera i inicijalizacija SPI1 modula mikrokontrolera. Sledeći korak je inicijalizacija senzorskih modula. Pre merenja, od korišćenih modula potrebno je postaviti inicijalne parametre kod Ambient 2 click i WEATHER click modula.



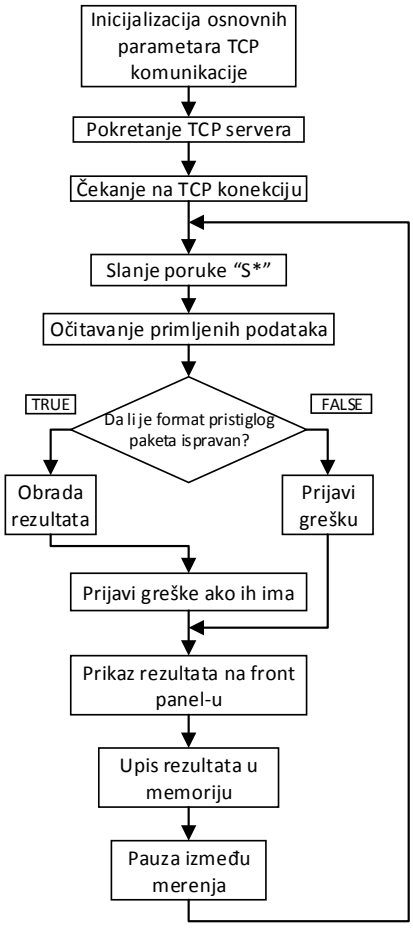
Slika 3. *Algoritam firmvera*

Sledeći korak jeste inicijalizacija senzorskih modula. Ova podešavanje se odnose na odabir modova u kojima će moduli raditi, podešavanje mernih opsega, očitavanje kalibracionih konstanti itd. Nakon toga sledi uključivanje GSM/GNSS modula slanjem naponskog impulsa u trajanju 3 sekunde na jedan od njegovih pinova. Kada se GSM/GNSS click modul uključi, pristupa se postavljanju njegovih osnovnih parametara slanjem određenih tzv. AT komandi. Pored ostalih, tu se unosi i javna IP adresa računara na kojem se nalazi LabVIEW server, kao i broj porta (5000). Ako su sve ove komande uspešno prihvaćene od strane GSM/GNSS click modula, od tog momenta modul je uspostavio TCP konekciju sa udaljenim LabVIEW serverom i spreman je za razmenu podataka. Server je isprogramiran tako da čim se uspostavi konekcija sa njim, da nazad vrati određeni odgovor.

Naredni korak algoritma jeste ulazak u beskonačnu while petlju. Prvi korak unutar petlje je čekanje odgovora od servera. Dokle god server ne odgovori, algoritam se ne izvršava dalje. U trenutku kada server odgovori porukom "S*" (skraćeno od START), nastavlja se dalje sa izvršavanjem. Ako se nakon tumačenja primljenih podataka ispostavi da su podaci koje je poslao server primljeni na ispravan način, kreće se sa merenjem fizičkih veličina pomoću senzorskih modula pozivanjem funkcije MEASURE(). Kada se završi sa merenjem, odnosno prikupljanjem i obradom rezultata, funkcija MEASURE() se zatvara, i zatim poziva funkcija FORMAT(). Ova funkcija ima zadatak da prikupljene izmerene veličine formatira u niz karaktera. Pre upisivanja svake vrednosti u niz vrši se provera da li je merenje proteklo bez prijavljivanja greške. Ako je kojim slučajem došlo do neke greške na bilo kom senzoru, ili neki od senzora nije prošao ID check test, na mesta u nizu koja su rezervisana za taj senzor se upisuju karakteri ‘E’ koji signaliziraju grešku (error). Nakon formatiranja, taj formatiran niz karaktera se šalje putem TCP protokola na LabVIEW server, i ovde se završava jedna iteracija. Nakon ovoga, mikrokotroler se ponovo vraća na mesto gde očekuje poruku od servera. Dakle, pauze između dva merenja definišu pauze između dva slanja poruke "S*" od strane servera.

5. LabVIEW APLIKACIJA

Aplikacija je napravljena u LabVIEW programskom okruženju. Glavni zadatak aplikacije jeste da prikupi podatke o merenim veličina, obradi, prikaže na front panel-u i sačuva na računaru.

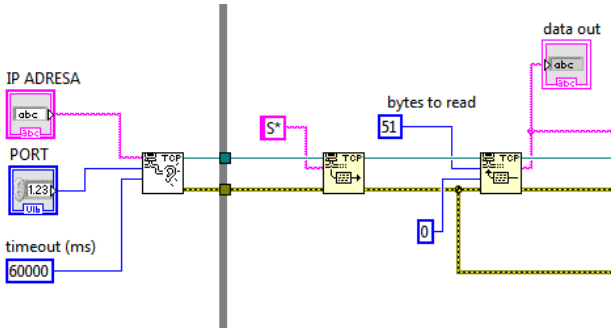


Slika 4. *Algoritam LabVIEW programa*

Na samom početku izvršavanja PC aplikacije, vrši se inicijalizacija komunikacije gde se podešava “slušanje” TCP komunikacije na portu 5000. Nakon pokretanja aplikacije, pokreće se TCP server na portu 5000. Od tog momenta, server čeka TCP konekciju, i tu program stoji dokle god se ne pojavi zahtev za konekciju od strane GSM/GNSS click modula.

Kada se uspostavi TCP konekcija, server ka GSM modulu šalje poruku "S*" što znači da da je dozvolu modulu da izmeri i da pošalje rezultate nazad. Za to vreme, server čeka paket koji treba da stigne od GSM modula sa podacim o izmerenim veličinama. Kada se podaci prime, proveravaju se da li su stigli prema određenoj konvenciji i ako jesu, dalje se obrađuju. U suprotnom prijavljuje se greška. Vrednosti koje su prošle sve kontrole se pretvaraju u određenu fizičku jedinicu i prikazuju na ekranu. Ako postoje polja koja sadrže karaktere 'E', to znači da je došlo do nekog problema pri merenju te veličine, gde je najčešći uzročnik sam senzror, i te greške se prijavljuju. Nakon prikaza na front panelu, izmereni podaci se čuvaju u memoriju.

Poslednji korak jeste generisanje vremenske pauze koju određuje korisnik. Kada pauza prođe, program ponovo šalje "S*" poruku kojom signalizira mikrokontroleru pa ponovi proces merenja i tako u krug.



Slika 5. *Deo LabVIEW koda vezan za TCP*

3. ZAKLJUČAK

Zahvaljujući LabVIEW programskom paketu, korisnici veoma jednostavno mogu da razmenjuju podatke preko Interneta zahvaljujući gotovim funkcijama koje se dobijaju uz ovaj paket. Front panel virtualnih instrumenata vrlo lako je distribuirati preko Web servisa na više različitih načina. Pored ovoga, velika većina korisnika zahteva interaktivnu komunikaciju sa udaljenim aplikacijama preko Web servisa, što je omogućeno u LabVIEW programu, čak i za korisnike koji nemaju visok nivo znanja iz programiranja.

Slika 6. *Front panel aplikacije*

Ovaj merno-informacioni sistem se pokazao stabilno u radu i testiran je u sobnim uslovima. Dalji rad na ovom sistemu može da ide pre svega u hardverskom usavršavanju, realizaciji PCB pločice i integraciji modula. LabVIEW nudi *remote panel* uslugu, koja omogućava da se front panel koji se nalazi na serverskoj mašini podeli sa dodatnim klijentima. To znači da se uz pomoć LabVIEW programa i remote panel usluge mogu pratiti rezultati merenja, gde god se klijenti nalazili. Realizacijom mobilnih aplikacija koje bi komunicirale sa serverom, mobilni uređaj postaje udaljeni virtualni merni instrument.

4. LITERATURA

[1] J. Tomić, M. Kušljević, V. Vujičić, M. Živanov, M. Slankamenac, "Realizacija virtualne laboratorije iz električnih merenja u LabVIEW programskom paketu," Tehničko rešenje (M85), Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad, 2013.

[2] B. Trump, The Signal, Texas Instruments, Dallas, Texas, 2017.

[3] C. S. Raghavendra, Krishna M. Sivalingam, Prof. Taieb Znat, Wireless Sensor Networks, Springer, New York, USA, 2006

[4] https://www.mikroe.com (20.09.2018.)

[5] Sovilj P. M., Milovančev S. S., Vujičić V.: Digital Stochastic Measurement of a Nonstationary Signal With an Example of EEG Signal Measurement, Instrumentation and Measurement IEEE Transactions on, 2011, Vol. 60 - issue 9, pp. 3230-3232, ISSN 0018-9456, DOI: 10.1109/TIM.2011.2128670

[6] Radonjic, A. ; Sovilj, P. ; Vujicic, V..: Stochastic Measurement of Power Grid Frequency Using a Two-Bit A/D Converter , Instrumentation and Measurement IEEE Transactions on, 2014, Vol. 63 - issue 1, pp. 56-62, DOI: 10.1109/TIM.2013.2277515, ISSN 0018-9456

[7] M. Urekar, P. Sovilj, „EEG dynamic noise floor measurement with stochastic flash A/D converter", Biomedical Signal Processing and Control, Vol. 38, pp. 337-345, Elsevier B. V, 2017, ISSN 1746-8094

[8] P. Sovilj, M. Milovanović, D. Pejić, M. Urekar, Z. Mitrović, Influence of Wilbraham-Gibbs Phenomenon on Digital Stochastic Measurement of EEG Signal over an Interval, pp. 270-278, Measurement Science Review, Vol. 14, No. 5, 2014, ISSN 1335 - 8871

[9] P. Sovilj, B. Vujičić, M. Sokola, D. Pejić, Ž. Beljić, Z. Mitrović, „Stochastic Measurement of Noise True RMS using 2-bit Flash A/D converters",Technical Gazette, Vol.24 No.5 October 2017, pp. 1315-1322, ISSN 1330-3651 (Print), ISSN 1848-6339 (Online), DOI 10.17559/TV-20151124100705

[10] Ž. Beljić, V. Vujičić, D. Pejić, M. Sokola, Z. Mitrović, P. Sovilj, „Grid Fundamental Harmonic Measurement in Presence of Gaussian Frequency Deviation Using 2-bit Flash A/D Converter", Technical Gazette, Vol.24 No.2 April 2017, pp. 481-488, ISSN 1330-3651 (Print), ISSN 1848-6339 (Online), DOI 10.17559/TV-20151109231714

Kratka biografija:

Stefan Mirković rođen je u Novom Sadu 1993. god. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu, na Katedri za električna merenja. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Merno-informacioni sistemi odbranio je 2018. god.
kontakt: mirkovicst@uns.ac.rs

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/01BE29Laketa

RAZVOJ MIKROFLUIDNOG ČIPA ZA DETEKCIJU PSIHOAKTIVNIH SUPSTANCI

DEVELOPMENT OF A MICROFLUIDIC CHIP FOR PSYCHOACTIVE SUBSTANCES

Jelena Laketa, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je razmatran problem detekcije droge primenom mikrofluidnih čipova. Za potrebe ovog rada izradili smo tri vrste mikrofluidnog čipa. Svaki čip se sastoji od 7 PVC slojeva i 2 zlatne elektrode, a različitog geometrijskog oblika mikrofluidnih kanala. Nakon laminacije, testiranje čipa se prvo vršilo ručno, a potom korišćenjem špric pumpe. Kada tečnost prođe kroz mikrofluidni kanal, dolazi do promene električnih parametara kondenzatorske strukture. Ova promena se merila korišćenjem analizatora impedanse. Na osnovu dobijenih rezultata zaključuje se da se korišćenjem ove metode mogu detektovati različite koncentracije trodona u rastvoru kao što je veštačka pljuvačka.*

Ključne reči: *mikrofluidika, medicinska elektronika, psihoaktivne supstance, analizator impedanse*

Abstract – *The paper studies the problem of drug detection using microfluidic chips. For the purpose of this thesis we have developed three types of microfluidic chips. Each chip is composed of 7 layers of PVC and 2 gold electrodes, and the different geometry of the microfluidic channel. After lamination, the first testing of chip was performed manually, and then using a syringe pump.When the liquid passes through a microfluidic channel, there is a change of the electrical parameters of the capacitor structure.This change is measured using impedance analyzer. Based on these results it is concluded that the using this method we can detect various concentrations of Tramadol (Trodon) in a solution such as an artificial saliva.*

Keywords: *microfluidics, medical electronics, psychoactive substances, impedance analyzer*

1. UVOD

Najnovija svetska istraživanja pokazuju da je danas na svetu oko 25 miliona ljudi zavisno od neke vrste ilegalne droge ili psihoaktivne supstance. Zbog velike pristupačnosti, broj se povećava iz dana u dan.

Prednost mikrofluidnih uređaja za detekciju psihoaktivnih supstanicu su: mali protok fluida, brze analize, sigurna i jeftina izrada. Zbog ovih prednosti u poređenju sa ostalim tehnikama, koje se uveliko već primenjuju, kao što su masena spektrometrija, UV spektroskopija ili NMR spektroskopija njihova primena postaje sve rasprostranjenija. U proteklom periodu sve je češća primena mikrofluidnih uređaja u različitim istraživanjima vezanim za drogu kao i u forenzičkoj analizi droge.

Primena mikroelektroda za detekciju ilegalnih droga je takođe ispitivana. Različite tehnologije i materijali mogu se koristiti za izradu mikrofluidnih uređaja: polidimetilsiloksan (PDMS), staklo kao i drugi polimeri ili papir. Kombinacija tehnika detekcije impedanse zasnovane na mikrofluidnim sistemima je već istražena. Vrednost impedanse zavisi od geometrije elektroda. Sistem interdigitalnih elektroda sa kanalom između njih prilikom prolaska tečnosti omogućuje promenu električnih karakteristika.

Ovaj rad opisuje novi dizajn mikrofluidnog čipa u kombinaciji interdigitalnih struktura elektroda sa mikofluidnim kanalima radi detekcije različite koncentracije opijata kroz promene električnih parametara kao što su impedansa, fazni ugao, kapacitivnost itd. Kompaktnost mikrofluidnog uređaja je postignuta primenom xurografik tehnike, zasnovane na laminaciji PVC folija. Nakon dizajna i fabrikacije mikrofluidnog uređaja u kratkom vremenskom periodu, zbog dužine mikrofluidnog kanala (malog rastojanja između inlet-a i outlet-a) mogu se dobiti rezultati varijacija električnih parametara za različite koncentracije opijata merenjem analizatorom impedanse.

Na ovaj način razvili smo manje subjektivan metod za detekciju nelegalnih droga u odnosu na dugotrajne laboratorijske analize i metod zasnovan na promeni boja.

2. MIKROFLUIDIKA

Mikrofluidika je mlada grana inžinjerstva, tehnologije i nauke koja se bavi proučavanjem i primenom fluida u protočnim sistemima submilimetarskih dimenzija.

Veličina mikrofluidnih sistema su veoma male, mikrofludini kanali su dimenzija od nekoliko milimetara pa do mikrometa. Na slici 1 je prikazana tabela koja pokazuje odnos zapremine fluida koja može da teče kroz određene dimenzije mikrofluidnih kanala.

Zapremina fluida		Dužina mikrofluidnog kanala
$1nL$	$10^{-3}mm^3$	$100\mu m$
$1\mu L$	$1mm^3$	$1mm$
$1mL$	$1cm^3$	$1cm$

Slika 1. *Tabelarni prikaz dimenzija mikrofluidnih sistema [1]*

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Goran Stojanović, red. prof.

Mikrofluidni čipovi se na osnovu materijala od kog su izrađeni dele na meke i čvrste.
Danas se sve više izrađuju čipovi od mekih materijala kao što su PVC folije kako zbog cene tako i zbog brže izrade (Slika 2).

Slika 2. *Izgled mikrofluidnog čipa izrađen od mekog materijala [2]*

Prednosti mikrofludnih sistema sa ekonomskog aspekta: manja potrošnja reagenata, manja potreba uzoraka, smanjena potrošnja energije kao i velika efikasnost.

Mikrofluidika je mlada i još uvek neistražena oblast inženjerstva, koja nam omogućava da kontrolišemo tečnosti na mikro i nano skali, omogućavajući naukama kao što su medicina, hemija i biologija da zakorače van poznatih granica. Zamislite, na primer, rastvor u kom se nalazi milion sićušnih ćelija koje treba prebrojati i analizirati kako bi se detektovale anomalije ili bolesti.

Uraditi ovo ručno veoma je teško, dugotrajno i nedovoljno precizno. Mikrofluidika nam omogućava da ovu vrstu analiza sprovedemo veoma brzo i precizno, analizirajući ćelije jednu po jednu.

3. RAZLIČITE STRUKTURE MIKROFLUIDNIH ČIPOVA

Za mikrofluidni sistem detekcija pomoću pljuvačke predstavlja veliki izazov zbog svoje kompleksnosti. Pogodna je za analizu malih molekula kao što je kokain, ali zbog svog sastava lako može doći do začepljenja kanala na čipu. Dokazano je da se sastav pljuvačke menja u toku dana. Mikrofluidni sistemi uglavnom se sastoje od fluidnih kanala kroz koje se propušta tečnost, pumpnih delova i elektronike za prikupljanje podataka i kontrolu. Da ne bi došlo do začepljenja prilikom protoka fluida posebno se mora voditi računa o materijalu od kojeg se izrađuju mikrofluidni kanali [3].

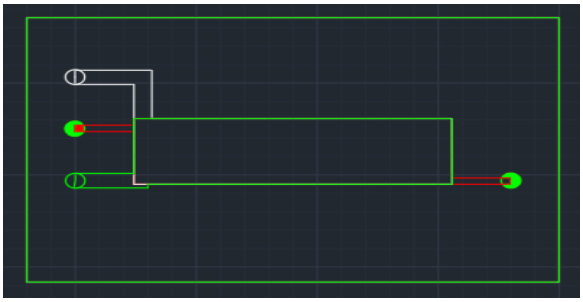
Za detekciju kokaina u ljudskoj pljuvački postoji nekoliko različitih mikrofluidnih čipova za koje je dokazano da uspešno mogu detektovati psihoaktivnu supstancu.

4. POSTUPAK IZRADE MIKROFLUIDNOG ČIPA

Svaki mikrofluidni čip sastoji se od 7 slojeva. Prvi sloj je PVC folija koja služi kao podloga- debljina PVC folije koju smo koristili za sve PVC slojeve iznosi 80um. Drugi sloj predstavlja elektrodu izrađenu od zlata - elektroda, kao i ostali slojevi sečeni su korišćenjem Cutter plotting mašine.

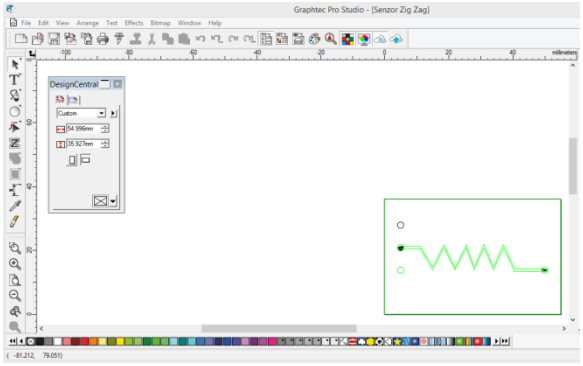
Treći sloj predstavlja PVC foliju na kojoj je izrađen mikrofluidni kanal za protok fluida - radi poboljšanja protoka fluida kroz kanal, četvrti i peti sloj su isti kao i treći. Šesti sloj predstavlja drugu elektrodu, takođe izrađenu od zlata - koja zajedno sa prvom elektrodom čini željenu strukturu.Dok je sedmi sloj PVC folija koja sadrži ulaz i izlaz pomoću kojih vršimo testiranje čipa.

Za izradu dizajna čipa koristili smo AutoCad koji je jedan od najpoznatijih softverskih programa za računarsko projektovanje (Slika 3). AutoCAD je specifičan po izuzetnoj preciznosti (koja može ići i ispod milimikrona) za merenje sistema i dimenzioniranje sistema sa automatskim računanjem koji zadovoljava najstrožije tehničke standarde.



Slika 3. *Prikaz krajnjeg izgleda čipa sa širokim elektrodama*

Ovaj dizajn se potom uvuče u GraptecPro Studio 2.10 softverski program. Korišćenjem ovog programa vrše se dodatna podešavanja kao što su promena dimenzija, pozicioniranje čipa, rotiranje, promena boja linija po kojima cutter vrši isecanje kao i podešavanje sile i brzine kojom će se izvršiti, slika 4.



Slika 4. *Izgled softverskog programa koji je povezan sa cutter-om*

Kada se podese svi parametri možemo početi isecanje označene strukture na foliji. Folija je prethodno zalepljena na kapton.

Postupak lepljenja folije obavlja se na sledeći način. Isečemo kapton odgovarajućih dimenzija (dimenzije se određuju na osnovu površine koju cutter uređaj može da pređe). Zatim na karton pažljivo zalepimo lepljivu traku, vodeći računa da se ne pojave nabori (ukoliko je neophodno može se preći valjkom kako bi se traka izravnala). Preko lepljive trake stavljamo PVC foliju debljine 80 μm. Folija ima dve strane sjajnu i mat (lepljivu), prilikom postavke folije mat strana se stavlja na lepljivu.

Ukoliko se radi sa folijom zlata postupak je malo drugačiji. Preko PVC folije pređe se lepkom. U ovom radu korišćen je Mixition lepak koji obezbeđuje dobru adheziju zlatnim listićima. Treba ostaviti da se lepak delimično osuši pre nanošenja zlata. Na tako pripremljenu podlogu postavi se tanak zlatan list na kome se nalazi zlato, a potom se list odvoji.

Isecanje struktura na foliji vrši se pomoću cutter plotter mašine (slika 5). Pre početka isecanja potrebno je proveriti da li je postavka podloge (na kojoj se nalazi materijal za isecanje) dobro postavljena kako ne bi došlo do kačenja ili preskakanje igle koja vrši isecanje.

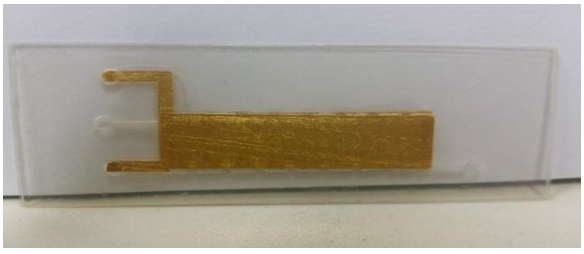
Slika 5. *Izgled cutter uređaja sa postavkom za sečenje*

Nakon isecanja vrši se odvajanje strukture sa folije. Laminaciju (spajanje) izdvojenih struktura obavljamo korišćenjem termalnog laminatora. Prilikom spajanja treba voditi računa da lepljiva strana folije (mat) ide na lepljivu stranu, takođe treba voditi računa o zaštiti slojeva prilikom ubacivanja u laminator.

Laminator ima mogućnost podešavanja brzine i temperature kojom se vrši laminacija. U zavisnosti od izbora materijala zavisi i izbor ovih parametara. Za izradu naših čipova podesili smo brzinu 2 i temperaturu od 120°C (slika 6).

Slika 6. *Radi zaštite prilikom laminiranja, slojevi se postave prvo u kapton pa se potom ubace u laminator*

Nakon laminacije dobijamo krajnji izgleda čipa na kojem vršimo testiranje (slika 7).



Slika 7. *Prikaz krajnjeg izgleda čipa sa meandriranim kanalom*

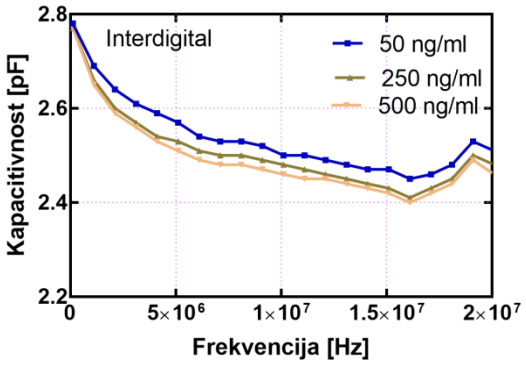
Za merenje željenih električnih parametara koristili smo Impedance Analyzer HP4194A. Pomoću ovog instrumenta želeli smo da ispitamo promenu kapacitivnosti u zavisnosti od frekvencije za različite vrednosti koncentracije fluida u kanalu (slika 8).

Slika 8. *Postavka instrumenta za merenje različitih koncentracija uzorka*

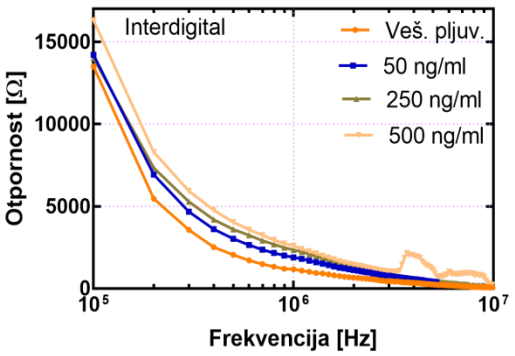
5. REZULTATI MERENJA

Merenja su se vršila za različite vrednosti koncentracije trodona (50mg/ml, 250mg/ml i 500mg/ml) rastvorenog u veštačkoj pljuvački korišćenjem analizator impedanse. Tečnost se ubrizgavala ručno korišćenjem šprica u inlet mikrofluidnog čipa.

U zavisnosti od različite vrednosti koncentracije uzoraka kao i različitog geometrijskog oblika mikrofluidnog kana i elektroda dobijali smo različite vrednosti električnih parametara. Na slikama 9 i 10 prikazani su rezultati dobijeni za mikrofluidni čip sa interdigitalnim kanalom.

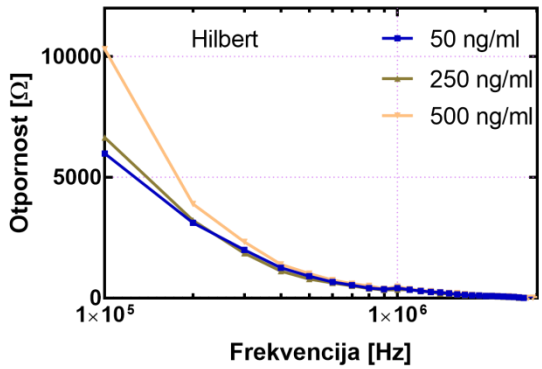


Slika 9. *Prikaz zavisnosti kapacitivnosti od frekvencije za različite koncntracije trodona*

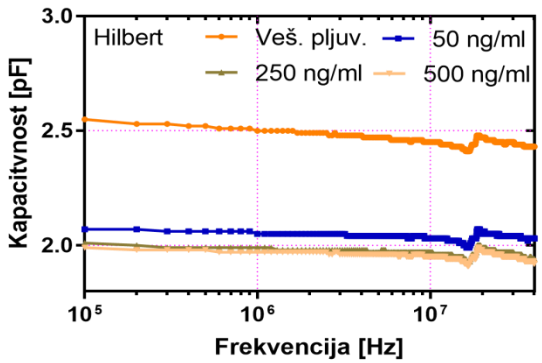


Slika 10. *Prikaz zavisnosti otpornosti od frekvencije za različite koncentracije trodona*

Na slikama 11 i 12 prikazani su rezultati za mikrofluidni čip sa hilbertovim kanalom.



Slika 11. *Prikaz zavisnosti otpornosti od frekvencije za različite koncentracije trodona*



Slika 12. *Prikaz zavisnosti kapacitivnosti od frekvencije za različite koncentracije trodona*

Dobijeni rezultati pokazuju uspešno detektovanje varijacije koncentracije trodona u uzorku prilikom prolaska tečnosti kroz mikrofluini kanal koji se nalazi između dve elektrode.

6. ZAKLJUČAK

Ovaj rad ima cilj da objasni dizajn i fabrikaciju mikrofluidnih čipova kao i njegovu primenu u detektovanju droga i prihoaktivnih supstanci u vodenim i biološkim sredinama kao što je pljuvačka. Ovakav princip detektovanja doprinosi uštedi novca, zbog pristupačnosti materijala koji se koristi za izradu kao i lakoj prenosivosti uređaja. Pored toga predstavlja neinvanzivan metod uzorkovanja kao i brzu analizu.

7. LITERATURA

[1] H. Bruus, "TheoreticalMicrofluidics", Oxford University Press, 2008.
[2] N. Blow, "Microfluidics: in search of a killer application", Nature Methods, 4:665 – 670, 2007.
[3] A. Manz and J.C.T. Eijkel, "Miniaturization and chip technology. what can we expect?", Pure and Applied Chemistry, 73(10):1555–1561, 2001.

Kratka biografija:

Jelena Laketa rođena je u Kikindi 1993. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva- Medicinska elektronika, odbranila je 2017. god. na temu "Određivanje električnih parametara u uzorku krvi na papiru".

Goran Stojanović doktorirao je na Fakultetu tehničkih nauka 2005. godine, a od 2015. godine je u zvanju redovnog profesora za užu naučnu oblast Elektronika. U svojoj dosadašnjoj karijeri je publikovao 70 naučnih radova u časopisima sa SCI liste, sa impakt faktorom. Koordinator je velikog broja međunarodnih projekata.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4:621.395
DOI: https://doi.org/10.24867/01BE30Cizmar

PRIMENA IBEACON PROTOKOLA NA IOS PLATFORMI

APPLICATION OF IBEACON PROTOCOL ON IOS PLATFORM

Dorian Čizmar, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *U radu je opisan način rada bikona kao i primena i implementacija iBeacon protokola na iOS platformi. Takođe su opisani problemi kod razvoja aplikacija koje koriste bikone. Postoji implementirano rešenje u Swift programskom jeziku.*

Ključne reči: *Bikon, iBeacon protokol, iOS platforma*

Abstract – *The thesis describes working of beacons and application and implementation of iBeacon protocol on iOS platform. Also there is description of problems that can be faced during development of application that use beacons. There is an implemented software solution in Swift programming language.*

Keywords: *Beacon, iBeacon protocol, iOS platform*

1. UVOD

Mobilni telefoni su se pojavili i aktivno se koristili još krajem prošlog veka. Vremenom je tehnologija napredovala i došlo je do enormnih izmena kod mobilnih telefona, te se oni telefoni i telefoni koji se danas koriste jako razlikuju.

Današnji tzv. pametni telefoni se sastoje od procesora, memorije, operativnog sistema. Sve to omogućuje razvoj aplikacija na pametnim telefonima. Aplikacije korisniku omogućavaju interakciju sa drugim sistemima i uređajima. Jedni od tih uređaja su bikoni. Bikoni su mali uređaji koji emituju radio signal koji može biti detektovan od strane pametnog telefona.

Ovaj rad prati softversko rešenje u vidu iOS aplikacije za inventarisanje robe. Aplikacija korisniku omogućuje da pravi listu stavki za inventar i kasnije da proverava da li su sve stavke iz liste prisutne.

2. iOS

iOS je operativni sistem za mobilne i tablične uređaje, razvijen od strane *Apple*-a. Ovaj operativni sistem može da se pokreće isključivo na *iPhone* i *iPad* uređajima.

2.1 Slojevi iOS-a

Postoji 4 sloja iOS operativnog sistema [1]. Prvi sloj pod nazivom *CoreOS* je najniži sloj i odnosi se na servise vezane za hardver, npr. rad sa fajl sistemom, *bluetooth* servisi, servisi za pristup *Keychain*-u itd [1]. *Core Services* je drugi sloj i nudi servise za mrežu, lokaciju, lokalnu bazu podataka itd [1]. Treći sloj pod nazivom *Media Core* predstavlja sloj za obradu slike, zvuka i video sadržaja [1]. Najviši sloj iOS operativnog sistema jeste *CocoaTouch* i zadužen je za korisnički interfejs. Tu se nalaze dugmadi, tekstualna polja, razne kontrole itd [1].

2.2 Radno okruženje

Radno okruženje koje se uglavnom koristi za programiranje iOS aplikacija jeste Xcode. Xcode je razvijen od strane *Apple*-a.

2.3 Jezici

Postoje dva jezika za razvoj iOS aplikacija. To su Objective-C i Swift.

2.4 Swift programski jezik

Swift je najnoviji programski jezik za razvoj *iOS*, *macOS*, *watchOS* i *tvOS* aplikacija. Nudi primarne tipove kao što su *Int*, *Double*, *Float*, *Bool* i *String* [2]. Takođe postoje tri tipa kolekcija a to su *Array*, *Set* i *Dictionary* [2].

Varijable se mogu koristiti na dva načina. Kada definišemo varijablu sa ključnom reči *var* tada se ona može menjati bilo kada. Ukoliko definišemo varijablu sa *let* to znači da vrednost te varijable više ne možemo da menjamo [2].

2.5 Storyboard

XCode razvojno okruženje nudi jednostavan način generisanja grafičkog korisničkog interfejsa, preko *storyboard*-a [3]. Pomoću *storyboard*-a se može definisati korisnički interfejs za sve ekrane u aplikaciji i sve veze između njih. Postoji skup predefinisanih komponenti koje se prevlače na ekran za koji pravimo korisnički interfejs.

2.6 Softverski obrazac MVC

U današnje vreme postoje različiti pristupi kako organizovati kod i dodeliti uloge određenim komponentama sistema. Tako postoje različite softverske arhitekture za razvoj *iOS* aplikacija a to su *MVC*, *MVVM*, *MVI*, *MVP*, *VIPER* i druge. Ne postoji striktno propisana softverska arhitektura koja se mora koristiti ali *Apple* preporučuje korišćenje *MVC* softverske arhitekture. *Model* predstavlja sloj koji treba da opisuje entitete i sadrži poslovnu logiku aplikacije. *View* sloj služi za prikazivanje podataka, dok *Controller* treba da poveže prethodna dva modula.

3. BIKONI

Bikon je mali *Bluetooth Low Energy* uređaj koji emituje radio signal. Taj signal može da bude detektovan od strane mobilnih uređaja.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Stevan Gostojić, vanr. prof.

3.1 Hardverska strana i fizika bikona

Bikon je podeljen u 3 celine: *ARM* računar, Bluetooth modul i baterija [4]. *ARM* računar predstavlja centralnu procesorsku jedinicu (eng. *Central Processing Unit - CPU*) koja pokreće softver niskog nivoa programiran tako da brine o ponašanju bikona. *CPU* modul takođe sadrži i antenu koja emituje radio signal [4]. *Bluetooth* modul je zadužen za prenos podataka *bluetooth* tehnologijom. Bikoni uglavnom koriste tzv. pametni *Bluetooth* standard (eng. *Smart Bluetooth standard*). Maksimalna veličina jednog paketa prilikom prenosa podataka je 257 bajta [4]. Ova količina jeste dovoljna za slanje podataka vezanih za sam bikon. Bikon ne šalje signal konstantno, već to radi tako što šalje signal u intervalima. Princip rada bikona je jednostavan. Centralna procesorska jedinica zna koje podatke treba da pošalje na osnovu softvera koji pokreće. Antenom se šalje *Bluetooth* paket sa podacima iz ARM računara, a električna energija se snabdeva baterijom.

3.2 Vrste tehnologija bikona

Tehnologija bikona podrazumeva skup mogućnosti bikona koji mogu da se konfigurišu pre upotrebe bikona. Postoji *Apple*-ova tehnologija pod nazivom *iBeacon* i *Google*-ova koja se zove *Eddystone*.

3.3 Konfiguracija bikona

Konfiguracija bikona predstavlja podešavanje određenih parametara koji utiču na način slanja signala.

3.3.1 Konfiguracija *iBeacons*

Parametri za konfigurisanje ove tehnologije su [6]:

1. UUID – Univerzalni jedinični identifikator
2. Major – Broj od 1 do 65535
3. Minor – Broj od 1 do 65535
4. Vremenski interval – interval u kom se šalje signal
5. Jačina slanja signala (u decibelima)

3.3.2 Konfiguracija *Eddystone*

Kod konfiguracije *Eddystone* tehnologije mogu se konfigurisati 3 različita paketa [7]:

1. Eddystone-UID – definiše se *Namespace* (statički identifikator dužine 10 bajta) i *Instance* (6-bajtni tekst)
2. Eddystone-URL – definiše se URL koji će se slati uz ovaj paket
3. Eddystone-TLM – Ovaj paket se ne šalje često, a kad se šalje onda se šalje uz UID ili URL pakete, sa dodatnim informacijama o bikonu (temperatura, zdravlje baterije, verzija itd.)

3.4 Načini skeniranja

Postoje dve vrste skeniranja bikona a to su *monitoring* i *ranging* [5]. Kod monitoringa se aplikacija obaveštava kada je uređaj ušao u region koji se skenira odn. primio signal od bikona. Informacije o bikonu koji je poslao signal se ne šalju kod ove vrste skeniranja. Primera radi, kada se ulazi u sportsku prodavnicu, može se koristiti ovaj režim skeniranja da se obavesti korisnik o potencijalnim popustima na određene artikle [5]. U ovom slučaju, aplikaciji nije bitno koji tačno bikon šalje signal, već u kom se regionu nalazi uređaj. *Ranging* skeniranje pruža više detalja o bikonu koji šalje signal. Tako se mogu poslati sledeće informacije [5]:

1. UUID, minor i major
2. Tačnost udaljenosti u metrima
3. Relativnu udaljenost (enumeracija sa vrednostima: blisko, daleko, neposredna blizina i nepoznato)
4. Jačina signala (u decibelima)

4. SPECIFIKACIJA ZAHTEVA

Uz ovaj rad je implementirana iOS aplikacija čija je namena da omogući inventarisanje pokretne imovine. Ideja je da svaki element koji treba da bude inventarisan poseduje svoj bikon i da na taj način aplikacija dobija informacije o elementu koji treba da se nađe u inventaru.

4.1 Funkcionalni zahtevi

Ovi zahtevi su prikazani pomoću dijagrama slučajeva korišćenja (eng. *Use-case diagram*) (slika 1).

Slika 1. *Dijagram slučajeva korišćenja*

Najbitniji slučajevi korišćenja su:

1. Prijava u aplikaciju
2. Odjava iz aplikacije
3. Dodavanje novog inventara
4. Odabir željenog inventara
5. Dodavanje stavke u inventar
6. Brisanje stavke iz inventara
7. Izmena stavke iz inventara
8. Početak inventarisanja
9. Završetak inventarisanja
10. Pregled stavki inventara na mapi
11. Generisanje izveštaja za inventar
12. Slanje izveštaja elektronskom poštom
13. Konfigurisanje korisničkog profila

4.2 Nefunkcionalni zahtevi

Neki od nefunkcionalnih zahteva su:

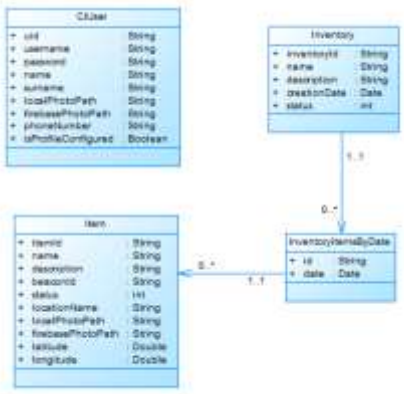
1. Instalacija aplikacije – instalacija treba da se vrši preko *XCode*-a.
2. *Offline mode* – Aplikacija treba da prikazuje postojeće inventare i bez Internet konekcije.

3. Softverski zahtevi – korisnik treba da uređaj minimalno sa iOS 11 operativnim sistemom.
4. Hardverski zahtevi – Korisnik treba da poseduje *iPhone* ili *iPad* uređaj i *Estimote* bikone.
5. Performanse – Korisnik ne sme da čeka duže od 5-6 sekundi na skeniranje bikona.
6. Sigurnost – Kredencijali korisnika treba da se čuvaju u *Keychain*-u.
7. Održivost – Aplikacija treba da je razvijena po nekoj od gore navedenih softverskih arhitektura.
8. Dodavanje korisnika u sistem – treba da se vrši isključivo na web portalu.

5. SPECIFIKACIJA DIZAJNA

Dizajn sistema je opisan sa dva tipa dijagrama. Jedni opisuju strukturu aplikacije (dijagram klasa i dijagram komponenti), dok drugi opisuje ponašanje aplikacije (dijagrami sekvenci).

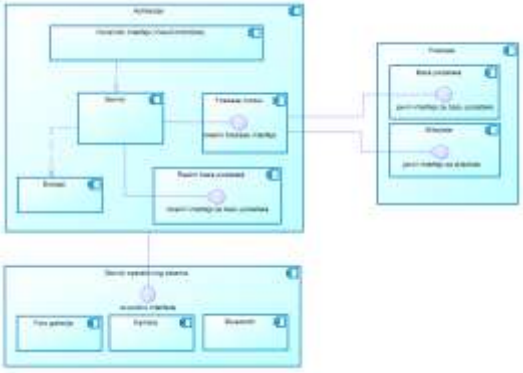
5.1 Dijagram klasa



Slika 2. *Dijagram klasa*

Dijagram klasa (slika 2.) sadrži 4 klase. Prva klasa jeste *CIUser* koja predstavlja model korisnika sa njegovim osnovnim podacima. Druga klasa modeluje inventar. To je klasa pod nazivom *Inventory*. Inventar može da se vrši više puta, te je neophodno modelovati datum inventara. Datum inventara je opisan klasom *InventoryItemsByDate*. Inventar za svaki datum može da ima listu stavki. Stavka je modelovana klasom *Item*.

5.2 Dijagram komponenti



Slika 3. *Dijagram komponenti*

Ovaj dijagram (slika 3) opisuje strukturu čitavog sistema. Postoje dve celine a to su server i mobilna aplikacija. Aplikacija koristi servise operativnog sistema kao što je bluetooth servis, kamera i foto galerija. Server je predstavljen *Google*-ovim servisom pod nazivom *Firebase*. Aplikacija se obraća *Firebase*-u preko *Firebase* modula. Taj modul parsira podatke i šalje ih servis modulu. Servis modul koristeći *Realm Database* modul pamti podatke lokalno i prikazuje ih na ekranu.

6. IMPLEMENTACIJA

U ovom odeljku je opisana implementacija bitnih delova aplikacije za inventarisanje nepokretne imovine.

6.1 Struktura projekta

Projekat je urađen po ugledu na MVC softversku arhitekturu te stoga postoje Model, View i Controller.U *Model*-u se nalazi opis entiteta, komunikacija i parsiranje zahteva i odgovora servera kao i sva poslovna logika aplikacije. U *View* sloju se prikazuju podaci pripremljeni u modelu, a ViewController spaja Model i View slojeve tako što uzme pripremljene podatke od modela i prezentuje ih u View sloju.

6.2 iBeacons u Swift programskom jeziku

Napravljena je bazna klasa *BeaconScanner* koja sadrži logiku za konfigurisanje regiona za skeniranje i generalne stvari koje su potrebne za skeniranje bikona. U toj klasi se instancira klasa koja brine o skeniranju bikona. To je klasa *ESTBeaconManager*.

6.3 Dodavanje stavke u inventar

Kada se dodaje stavka u inventar, korisnik unosi podatke preko View sloja aplikacije. Ti podaci se obrađuju u servis sloju a nakon toga se vrši dodavanje tih podataka na *Firebase*. Za dodavanje podataka na *Firebase* zadužena je klasa *FirebaseInventoryEngine* koja u sebi instancira *FirebaseDatabase* objekat koji je zadužen za komunikaciju sa serverom.

6.4 Algoritmi za skeniranje bikona

Algoritmi se nalaze u klasama *NearestBeaconScanner* i *MonitoringBeaconScanner*. Obe klase nasleđuju baznu klasu BeaconScanner i implementiraju logiku za različite potrebe skeniranja.

6.4.1 Struktura podataka za skladištenje skeniranih bikona

Struktura predstavlja modifikovan red (eng. *queue*), koji ima ograničenje za broj elemenata u redu. Ukoliko se doda novi element a red je pun, tada se iz reda izbacuje prvi dodati element.

6.4.2 Algoritam za skeniranje najbližeg bikona

Ovaj algoritam koristi *ranging* način skeniranja. Svake sekunde se dobija lista bikona i proverava se da li je bikon u neposrednoj blizini na 0,5m sa jačinom signala većom

od -75db. Ukoliko su zadovoljeni ti uslovi, bikon se dodaje u red. Ukoliko postoji više od 3 istih bikona u redu taj bikon se uzima kao skenirani bikon za stavku.

6.4.3 Algoritam za skeniranje bikona na osnovu liste identifikatora

Pre početka algoritma se definiše lista bikona koji treba da budu skenirani. Takođe se koristi *ranging* način skeniranja. Svake sekunde se dobija lista bikona. Algoritam gleda da li postoji neki bikon iz dobijene liste bikona u predefinisanoj listi bikona. Ukoliko postoji i ukoliko je udaljenost bikona između 0 i 1 tada se dodaje u red. Ukoliko postoji više od 3 istih bikona u redu može se smatrati da je bikon pronađen.

7. DEMONSTRACIJA

Na slici 4 je prikazan ekran sa listom stavki u jednom inventaru. Na tom ekranu se može vršiti dodavanje, izmena i brisanje stavke. Takođe klikom na „Start inventory“ se započinje novi inventar. Nakon završetka inventara moguće je generisati izveštaj u PDF formatu.

Slika 4. *Stavke u inventaru*

8. ZAKLJUČAK

U ovom radu je opisan princip rada bikona na iOS platformi. Dve najbitnije stvari kod razvoja aplikacije koja treba da radi sa bikonima su da se bikoni konfigurišu u skladu sa potrebama i da se osmisli odgovarajući algoritam za skeniranje bikona.

Aplikacija koja prati ovaj rad ne podržava rad više korisnika na jednom inventaru, što bi u stvarnosti trebalo da bude omogućeno, te se to može smatrati kao mana i polje gde ova aplikacija može da se unapredi. Takođe aplikacija je ograničena na rad sa *Estimote* bikonima, stoga se ovo takođe može smatrati kao mana i oblast za unapređenje.

9. LITERATURA

[1] Stanford kurs za *iOS 11* programiranje - https://docs.microsoft.com/en-us/xamarin/cross-platform/get-started/introduction-to-mobile-development (pristupljeno u septembru 2018.)

[2] Swift dokumentacija - https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html (pristupljeno u septembru 2018.)

[3] Storyboard - https://developer.apple.com/library/archive/documentation/General/ Conceptual/Devpedia-CocoaApp/Storyboard.html (pristupljeno u septembru 2018.)

[4] Estimote blog o hardveru bikona - https://blog.estimote.com/post/106913675010/how-do-beacons-work-the-physics-of-beacon-tech (pristupljeno u septembru 2018.)

[5] Estimote blog o monitoringu i rangiranju - https://community.estimote.com/hc/en-us/articles/203356607-What-are-region-Monitoring-and-Ranging- (pristupljeno u septembru 2018.)

[6] Apple dokumentacija o iBeacon tehnologiji - https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf (pristupljeno u septembru 2018.)

[7] Google dokumentacija o Eddystone format - https://developers.google.com/beacons/eddystone (pristupljeno u septembru 2018.)

Kratka biografija:

Dorian Čizmar rođen je u Vrbasu 1993. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranio je 2016. god.

kontakt: 064/0456-945
mail:theory93rk@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE31Andjelic

PROCEDURALNO GENERISANJE SCENE RAČUNARSKE IGRE ZASNOVANO NA GRAMATICI

PROCEDURAL GENERATION OF GRAMMAR-BASED COMPUTER GAMES

Lazar Anđelić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *Opis jednog načina primene L-sistema zajedno sa standardnom gramatikom za generisanje 3D scene računarske igre.*

Ključne reči: *L-sistem, Gramatika, Proceduralno generisanje, Video igra*

Abstract – *Description of a method of using L-systems combined with a formal grammar to generate a 3D scene usable in video game.*

Keywords: *L-system, Grammar, Procedural generation, Video game*

1. UVOD

Uz napredak tehnologije i povećanje mogućnosti računara, raste i želja korisnika za što detaljnijem i realinijem prikazu. Posebno industrija video igara ima težak zadatak da zadovolji očekivanja za najveće projekte, i troškovi izrade takvih projekata naglo rastu.

Potencijalno rešenje za problem stvaranja sadžaja koji zadovoljava korisnička očekivanja je primena metoda proceduralnog generisanja. Ove metode se već dugo primenjuju u oblasti računarske grafike, i tu imaju širok spektar primene. Samo neki od primera primene su generisanje zemljišta, vodenih površina i biljaka. Kompleksne scene, koje sadrže veliki broj različitih modela, mogu da zahtevaju više meseci da se ručno naprave, ali primenom metoda proceduralnog generisanja one mogu biti generisane i za manje od sat vremena.

Metode proceduralnog generisanja je moguće primeniti i za generisanje entiteta koji su nastali uticajem ljudi, kao što su gradovi. Gradovi su vizuelno kompleksni i rezultat su razvoja pod uticajem mnogobrojnih faktora, koji može da traje i više od stotinu godina. Veliki je izazov napraviti realan model za proceduralno generisanje toliko kompleksnog sistema.

2. FORMALNE GRAMATIKE

Formalna gramatika je deo formalne teorije jezika. Formalnu teoriju jezika je opisao N. Čomski 1956. godine. U pitanju su velika pojednostavljenja i apstrakcije nad empirijskim domenom prirodnih jezika.

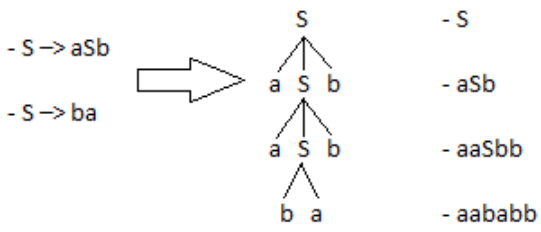
NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivetić, red. prof.

Jedna od ključnih osobina formalne teorije jezika jeste to da je značenje „reči“ koje čine neki jezik u potpunosti ignorisano. Apstrakcije u formalnoj teoriji jezika su veoma uspešno odabrane, što pokazuje široki obuhvat oblasti u kojima je do sada nađena primena iste.

Formalni jezik je neograničen skup nizova karaktera, gde su nizovi karaktera neodređene ali konačne dužine. Nizovi tih karaktera predstavljaju izraze u formalnoj teoriji jezika. Formalna teorija jezika je skup matematičkih alata koji opisuju načine definisanja i obrade formalnih jezika. U formalnoj teoriji jezika se istražuju matematičke osobine formalnih jezika, i načini da se ti jezici opišu na konačan način, čak i ako je neki jezik beskonačan. Uobičajeni način da se takav opis postigne je pomoću formalne gramatike. Takođe, unutar teorije jezka, jezici su hijerarhijski poređani po kompleksnosti. To je *hijerarhija čomskog*. Četiri grupe koje čine hijerarhiju su: računski prebrojivi jezici (tip 0), kontekstno osetljivi jezici (tip 1), kontekstno slobodni jezici (tip 2), i regularni jezici (tip 3).

Formalna gramatika se izražava pomoću matematičkih pravila i logike. Formalna gramatika se sastoji od četiri elementa. To su konačan skup simbola (*terminala*), koji su elementi jezika, konačan skup simbola (*neterminala*), *početni simbol* iz skupa *neterminala*, i konačan skup pravila (produkciona pravila).

Produkciona pravila su oblika $\alpha \rightarrow \beta$, što znači „α se može zameniti sa β“, i gde su α i β nizovi sastavljeni od terminala i neterminala. U slučaju da je na niz karaktera „xαy“ primenjeno pravilo $\alpha \rightarrow \beta$, dobija se niz karaktera „xβy“. Primer upotrebe formalne gramatike je na Slici 1.



Slika 1. *Primena pravila formalne gramatike*

2.1. Lindenmajerovi sistemi

Lindenmajerovi sistemi, često skraćeno na L-sistemi, su nastali kao matematički model za predstavljanje razvoja biljaka. Fokusiraju se na topologiju biljaka, odnosno odnose susednih ćelija ili većih gradivnih jedinica u sklopu biljke. L-sistemi su gramatika koja funkcioniše na principu zamene, odnosno iterativno zamenjuju elemente

trenutnog stanja objekta pomoću predefinisanih *pravila zamene*. Pravila zamene su u stvari *produkciona pravila*. L-sistemi se definišu kao uređeni par G=(*V*, *ω*, *P*), gde je *V* skup simbola koji sadrži *konstante* (terminale) i *promenljive* (neterminale), *ω* aksiom, odnosno početno stanje sistema, a *P* skup produkcionih pravila, odnosno pravila zamene.

Pravila zamene se primenjuju iterativno, počevši od početnog stanja. Primenjuje se svako pravilo koje je moguće primeniti u datoj iteraciji. To je glavna razlika u odnosu na ostale formalne gramatike, koje primenjuju samo jedno pravilo pri svakoj iteraciji.

Pravila zamene takođe određuju da li je L-sistem deterministički. Deterministički L-sistem je takav da za svaki simbol koji se nalazi sa leve strane produkcionog pravila postoji tačno jedno produkciono pravilo. Ako L-sistem nije deterministički, onda je stohastički.

L-sisteme je moguće primeniti za generisanje dvodimenzione i trodimenzione grafike. Ova funkcionalnost se nadograđuje na generisanje nizova karatera, tako što se proces generisanja tih nizova modifikuje tako da se u nizovima nađu određeni kontrolni simboli. Kontrolni simboli služe za upravljanje interpreterom generisanih nizova karaktera.

Interpreter se definiše kao uređena trojka *(x, y, α)*. Simboli *x* i *y* predstavljaju trenutne koordinate na kojima se nalazi kursor interpretera. Simbol *α* predstavlja ugao rotacije kursora, i tako je određen pravac u kom će se on kretati. Pored ta tri parametra, potrebno je definisati dužinu koraka *d* i povećanje ugla *δ*. Kada su svi pomenuti parametri definisani, u osnovnom, dvodimenzionom, obliku važe sledeće oznake: *F* kao kretanje napred za korak *d* i crtanje linije između prethodne i nove pozicije, *f* kao pomeraj napred za korak *d*, + i - kao povećanje ili smanjenje ugla *α* za *δ*, i | kao obrtanje smera kretanja. Osim osnovnih funkcionalnosti interpretera, moguće je uvesti još dve. Njihovi simboli su [i], a značenje im je sledeće: [kao čuvanje trenutnog stanja kursora na *LIFO* („pushdown“) steku, i] kao uzimanje stanja sa steka i korišćenje istog kao trenutno stanje kursora. Iako se pri uzimanju stanja sa steka kursor pomera, linija se ne crta.

3. PROCEDURALNO GENERISANJE

Proceduralno generisanje sadržaja je automatsko stvaranje digitalnih objekata za video igre, simulacije ili filmove, zasnovano na predefinisanim algoritmima i šablonima koji zahtevaju minimalnu korisničku interakciju [1]. Može smanjiti pritisak na dizajnere tako što automatizuje deo dizajnerskog procesa. Isto tako, može biti i previše vremenski zahtevno za implementirati i konfigurisati neki kompleksan algoritam koji bi zadovoljio potrebe video igre.

Postoji mnogo kategorija proceduralnog generisanja, i skoro sve imaju isti problem: nedostatak kreativnosti i izvesnosti. Teško je sa sigurnošću potvrditi da je sadržaj stvoren nekim algoritmom novitet, ili da je koristan i visokog kvaliteta. Ipak, ta neizvesnost je takođe i dobra strana proceduralnog generisanja. Pomoću proceduralnog generisanja moguće je dobiti neočekivano kvalitetan sadržaj, kao i neupotrebljivo loš sadržaj. Zato treba biti veoma pažljiv pri razvoju metoda proceduralnog generisanja i napraviti algoritam koji je na granici između fiksnog, potpuno predvidivog sistema, i haotičnog, nepredvidivog sistema.

3.1. Neizvesnost u video igrama

Neizvestan rezultat procesa je potpuno nepoznat, dok je izvestan rezultat predodređen. Na granici između ta dva, nalazi se rizik, koji ima rezultate sa poznatom verovatnoćom dobijanja istih. Na osnovu toga, Salen i Cimerman su definisali neizvestan sistem kao sistem u kojem akcije uključuju neki nivo rizika [2]. Osim toga, identifikovali su nekoliko osobina takvih sistema koje su relevantne za proceduralno generisanje u video igrama. Prva je da, čak i bez upotrebe metoda proceduralnog generisanja, video igra može imati „osećaj“ nasumičnosti. Druga je da čak i igre koje su u potpunosti zasnovane na sreći mogu ponuditi smisao za nastavak igre, dok god igrači imaju šansu da iskoriste neke smislene prilike tokom igre. Treća osobina neizvesnih sistema je identifikovana kao prikrivanje rezultata koje bi igrači inače mogli da predvide. Sa takvim osobinama, igrač mora, barem donekle, rizikovati u svojim akcijama da bi došao do uspeha.

Nedostatak izvesnosti je neophodan deo u dizajnu video igre. Iz tog razloga je dobro što je proceduralno generisanje neizvesno. Neizvesnost se često pojavljuje kao jedan od ključnih elemenata igre. To igračima daje osećaj svrhe za nastavak igranja. Ako igrači u potpunosti poznaju logiku iza igre, i znaju šta se tačno dešava u istoj, gube osećaj neizvesnosti i interesovanje i prestaju da troše vreme na istu. Igračima treba dozvoliti da donesu smislene odluke, koje imaju neizvesne posledice. Metode proceduralnog generisanja uvode neizvesnost u video igre na jedinstven način. Na primer, nivoi koji se generišu pseudo nasumično, ili dijalozi i ponašanja likova koje kontroliše kompjuter, svi mogu da zavise od odluka koje igrač donese, ali da ne postoji jedan siguran rezultat za svaku odluku igrača.

Neizvesnost u video igrama ne mora da se ispoljava u obliku neizvesnosti događaja tokom trajanja igre. Mnoge video igre arkadnog tipa, kao „*Space Invaders*“ (1978), uvek imaju isti završetak, a to je da igrač izgubi. Ipak, igra poseduje neizvesnost u obliku rezultata koji igrač ostvari pre nego što se igra završi. Igrač ne može biti siguran koliki će mu biti maksimalan rezultat u partiji koju igra, niti zna da li će oboriti svoj prethodni maksimalan rezultat.

4. GENERISANJE SCENE POMOĆU GRAMATIKA

Potrebno je generisati trodimenzionu scenu upotrebljivu u video igri. Scena treba da sadrži urbanu i/ili prirodnu sredinu, koja treba biti raznolika. Scena se sklapa od predefinisanih trodimenzionih oblika koji su određeni kao elementarni oblici. Ti oblici mogu, uz odgovarajuće teksture, predstavljati deo bilo čega što se nađe na sceni. Generisana scena mora biti smislena predstava njoj odgovarajuće sredine u realnom svetu. Obavezno je upotrebiti gramatiku u toku implementacije rešenja. Za primere ranijih rešenja sličnih problema, pogledati reference [3, 4].

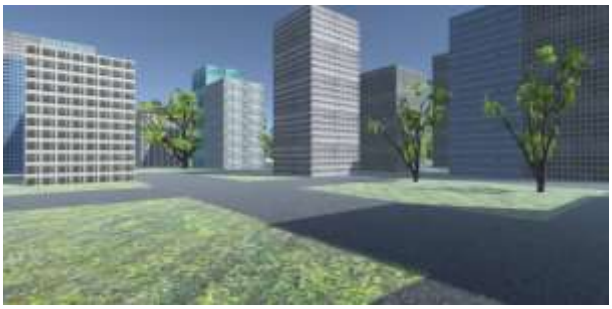
Scena se generiše u više koraka, i u pitanju je urbana sredina. Prvi korak je generisanje saobraćajnica. Zatim se

na osnovu saobraćajnica formira matrica koja predstavlja scenu. Kada je matrica formirana, kroz nju se iterira da bi se popunio ostatak scene. Tokom interpretacije, kursor se uvek kreće u istoj ravni.

Saobraćajnice se generišu pomoću L-sistema. L-sistem je implementiran sa osnovnim funkcionalnostima koje su opisane u Poglavlju 2.1. Aksiom, pravila zamene i broj iteracija su ulazne vrednosti koje korisnik može da specifikuje, i smeju sadržati osnovne karaktere opisane u Poglavlju 2.1. L-sistem generiše teksutalnu predstavu saobraćajnice, koja se zatim interpretira i crta na sceni. Tokom crtanja se obavlja i teksturisanje saobraćajnice.

Sledeći korak je napraviti dvodimenzioni niz, odnosno matricu, objekata koji sadrže informacije o koordinatama polja scene na kojima se generišu njeni delovi, kao i informacije o tipu tih polja na osnovu kojih se zna šta je generisano na tom polju. Tip polja može biti prazno polje, saobraćajnica, građevina, zelena podloga i slično.

Kada je opisana matrica napravljena, iteriranjem kroz nju se dolazi do polja koja još ne sadrže deo scene. Zatim se na njih, sa određenom šansom postavljaju objekti ili odgovarajuće podloge (travnjak i sl.). Pošto objekti mogu da se nalaze samo uz saobraćajnicu, šansa da na takvo polje bude postavljen objekat je veća nego šansa da se postavi podloga. Objekti i podloge se, kao i saobraćajnice, teksturišu prilikom generisanja. Za objekte se odabira jedna od petnaest ponuđenih tekstura, a za podloge se bira tekstura u odnosu na tip podloge. U slučaju da je generisana podloga definisana tako da može da podrži rast vegetacije, uz određenu šansu se postavljaju drva na istu. Primeri generisane scene iz perspektive igrača („*first person*“) su na Slici 2.



Slika 2. *Generisana scena iz perspektive igrača*

Ako su spoljni faktori kao što su rezolucija i broj tekstura izuzeti, brzina rada algoritma zavisi od tri faktora, i to su broj iteracija u radu L-sistema, sadržaj aksioma i broj promenljivih i njima odgovarajućih pravila zamene.

Ni jedan od ovih faktora nije značajniji od druga dva, već su svi međusobno zavisni. Dovoljno visok broj iteracija će uvek loše uticati na performanse algoritma, ali u slučajevima kada se zada kompleksan aksiom i/ili kompleksna pravila zamene, čak i mali broj iteracija (manje od 10) može znatno da utiče na performanse. Iz tog razloga nije moguće tvrditi da će za svaku situaciju manji broj iteracija obezbediti bolje performanse algoritma.

U Tabeli 1 predstavljene su performanse za jednu promenljivu „F“, aksiom „FFF-FFF-FFF-FFF“, i jedno pravilo zamene „F=FFF-FFF+FFF-FFF-FFF“.

Tabela 1. *Performanse algoritma kroz različit broj iteracija*

Broj iteracija	Prosečno trajanje [s]
1	0,05
2	0,75
3	65,3

Iz tabele je moguće videti da za dati primer aksioma i pravila zamene performanse naglo opadaju sa povećanjem broja iteracija. Ipak, za ovaj primer je dovoljno samo dve iteracije da bi se generisala scena dovoljna za neke video igre. Primer rezultata generisanja sa dve iteracije je na Slici 2. Trenutna implementacija ostavlja dosta opcija za unapređenja, među kojima su: upotreba stanja kurosra interpretera, upotreba stohastičkog L-sistema, upotreba L-sistema za generisanje objekata i vegetacije, generisanje tekstura i vodenih površina, kretanje kursora u prostoru i uvođenje sgementacije scene u obliku octree stabla radi jednostavne ručne izmene delova scene ili dodele semantike istima.

5. ZAKLJUČAK

U ovom radu je prikazan jedan pristup proceduralnom generisanju urbane okoline za video igru. Prvi deo algoritma se sastoji od L-sistema koji generiše mrežu saobraćajnica, a drugi deo od jednostavne gramatike koja definiše pravila generisanja ostatka scene. U zavisnosti od parametara L-sistema, algoritam može biti primenjiv u realnom vremenu.

Da bi sa trenutnom implementacijom rezultati generisanja bili raznovrsniji, potrebno je menjati ulazne parametre između svaka dva generisanja. Ulazni parametri su broj iteracija, aksiom i pravila zamene. Dovoljno je promeniti jedan od njih da bi se dobila različita mreža saobraćakmoca, ali potrebno je paziti da se ne zadaju parametri koji će onemogućiti rad u realnom vremenu, u slučaju da je to jedan od uslova. Potencijalna unapređenja algoritma su navedena u Poglavlju 4.

6. LITERATURA

[1] J. Freiknecht, W. Effelsberg, „*A Survey on the Procedural Generation of Virtual Worlds“*, 2017.

[2] K. Salen, E. Zimmerman, „*Rules of Play: Game Design Fundamentals*, 2004“.

[3] S. Ince, “*Automatic Dynamic Content Generation for Computer Games”*, 1999.

[4] Y. I. H. Parish, P. Mueller, “*Procedural Modeling of Cities”*, 2001.

Kratka biografija:

Lazar Anđelić rođen je u Novom Sadu 1993. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranio je 2018.god.

kontakt: cyber.laki@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE32Skulec

iOS AR APLIKACIJA ZA ASISTENCIJU KORISNIKA SA DALTONIZMOM

iOS AR APPLICATION FOR ASSISTING USERS WITH DALTONISM

Isidora Škulec, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIČKO I RAČUNARSTVO

Kratak sadržaj – *Predstavljanje aplikacije za asistenciju daltnonistima u realnom vremenu. Svrha aplikacije je brza i jednostavna identifikacija predmeta određene boje gledajući kroz kameru mobilnog telefona. Razmatrano je nekoliko načina sa namerom da bude jednostavno za korišćenje i da efikasno koristi resurse uređaja. Problem je rešen šrafiranjem regiona odgovarajuće boje.*

Ključne reči: *Augmentnovana realnost, daltonizam, aplikacija za mobilni telefon, iOS*

Abstract – *Presenting a real time application to assist daltonist users. Application purpose is fast and easy identification of certain coloured objects looking though mobile phone camera. Several ways were discussed, with intention being ease of use and efficient phone resource usage. Problem was solved by marking regions of certain colour.*

Keywords: *Augmented reality, daltonism, mobile app, iOS*

1. UVOD

Prema definiciji ljudska bića imaju trihromatsku viziju. Ovo znači da se percepcija boja svodi na tri tipa receptora, od kojih je svaki zadužen za određeni opseg talasnih dužina spektra svetlosti. Kombinacijom nadražaja na ove receptore ljudi su u stanju da prepoznaju boje. Međutim, određeni procenat stanovništva ima oslabljenu ili potpuno odsutnu funkcionalnost bar jednog od ova tri tipa receptora, što uzrokuje smanjen opseg boja kojima raspolažu i teže raspoznavanje određenih objekata u okruženju. Crveno-zeleni tip daltonizma javlja se kod 8% muške populacije.

Iako nije velik hendikep, može otežati ljudima neke svakodnevne stvari poput odabira odeće, razlikovanja zrelog od zelenog voća, gledanja na semafor, gledanja sportova, pronalaženja crvenih predmeta u travi i sl.

Neke stvari poput semafora imaju ovaj problem rešen delimično korišćenjem drugog oblika ili redosleda svetala (zeleno će uvek biti iznad crvenog, na pešačkom prelazu će crveno će imati oblik čoveka koji stoji). Međutim, zrelo i nezrelo voće neće imati drugi oblik, privezak za ključeve koji padne u travu biće previše sitan da se raspozna po obliku, prikaz pogođenih i promašenih penala u toku fudbalske utakmice koristiće krugove obojene u crveno i zeleno.

U ovakvim slučajevima posledice za grešku većinom neće biti velike ali bez obzira izazivaju frustraciju.

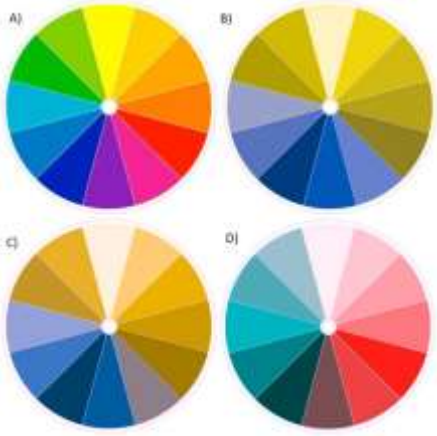
Ovaj rad proučava tipove oslabljene kolor vizije, zatim postojeća rešenja kojima je pokušano da se pomogne osobama sa oslabljenom kolor vizijom. Prikazaće se rešenje problema kroz aplikaciju koja pomoću augmentovane realnosti pokušava da nadomesti oslabljenu kolor viziju, kao i tehničke detalje kreiranja aplikacije sa augmentovanom realnošću na telefonu sa video kamerom.

2. KOLOR VIZIJA

Ljudsko oko sadrži dve vrste ćelija koje omogućavaju vid. Cilindrične ćelije osetljive su na jačinu svetla i zadužene su za monohromatsku viziju, dok su kupaste ćelije zadužene za percepciju boja. Kao što je već pomenuto, ljudska vizija je trihromatska. Ovo znači da ljudsko prepoznavanje boja zavisi od tri tipa kupastih ćelija: S, M i L ćelije zadužene su za, redom: short, middle, long odnosno kratke, srednje i duge talasne dužine svetlosti.

Oslabljena kolor vizija klasifikuje se kao: anomalična trihromatska vizija (gde je osetljivost jednog od tipova kupastih ćelija izmenjena), dihromatska vizija (gde je osetljivost u jednom tipu kupastih ćelija potpuno odsutna) i monohromatska vizija (kada je izgubljena osetljivost u dva ili sva tri tipa kupastih ćelija).

Nedostaci u L, M i S ćelijama redom označavaju se kao protan, deutan i tritan tipovi. Dakle, kod anomalične trihromatske vizije javljaju se protanomalija, deutanomalija i tritanomalija, dok se kod dihromatske javljaju protanopija, deutanopija i tritanopija, kao vrste kolor deficitne vizije (Slika 1.1) [1].



Slika 2.1 *Simulacija daltonizma*
A) normalan prikaz B) protanopija
C) deuteranopija D) tritanopija

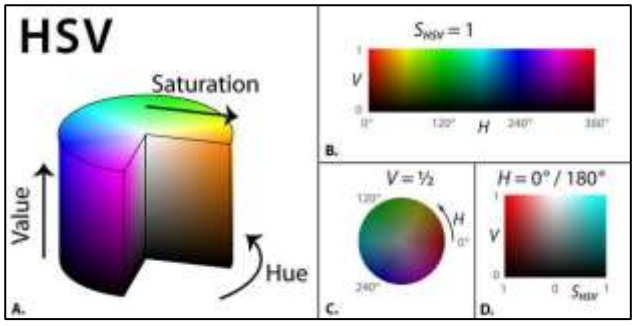
NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivetić, red. prof.

2. AR I SEGMENTACIJA BOJE

Augmentovana realnost predstavlja audio/vizuelni prikaz realnosti iz ugla korisinka kome je dodat računarski generisani prikaz tako da gledacu daje dodatne informacije o relanosti koje on prirodno ne bi percepcirao, a njemu su prezentovani na prirodni način. Mobilni telefoni obezbeđuju niz podataka koji se mogu iskoristiti, od GPS antene, akcelerometra i žiroskopa koji daju informacije o položaju, kretanju i brzini kretanja uređaja i njegovog korisnika, do audiovizuelnog sistema koji omogućava prikaz trodimenzionalnih modela u realnom svetu, zahvaljujući sve jačoj procesorskoj moći mobilnih uređaja. Neke od primena AR jesu mape i navigacija (Google Maps) i zabava poput AR igara (Ingress, Pokemon GO) i interaktivnih filtera za fotografije (Snapchat) [2].

Za potrebe AR, površine određene boje mogu se posmatrati kao markeri na koje se postavlja 2D sadržaj koji dopunjava kontekst korisnim informacijama. Zadatak je izolovati regione slike koji su zadate boje. *Color spaces*, odnosno modeli boja, predstavljaju način zapisa vrednosti boja. Potrebno je razmotriti koji modeli boja su dobri kandidati za vršenje segmentacije i odabrati najpogodniiji.

HSV model boja (*hue, saturation, value* - nijansa, zasićenost, intenzitet) i njemu sličan HSI su modeli koji se često koriste za analizu digitalne slike zbog jednostavnosti prikaza boje i intuitivnosti modela. H komponenta približno preslikava spektar vidljive svetlosti (slika 2.1-B), čime je veoma olakšano definisanje svake boje (sa slike 2.1-D može se videti da se nijanse tirkizne nalaze u okolini 180^o). S komponenta opisuje zasićenost, odnosno raspon od belog do pune boje, dok V komponenta daje intenzitet boje [3].



Slika 2.1 *HSV model boja*

3. POSTOJEĆA REŠENJA

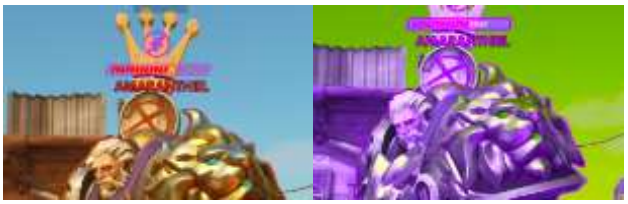
EnChroma je firma koja proizvodi naočare za ljude sa protanomalijom i deuteranomalijom, ali ne za ljude sa čistom dihromatskom vizijom. Njihove naočare filtriraju spektar svetlosti kako bi izolovali čisto crvene i zelene nijanse i na taj način pojačale kontrast između boja. Međutim, iako pojačavaju saturaciju svetla koje dolazi do oka, ne mogu da nadomeste nedostatke u receptorima, te korisnici i dalje ne mogu da prođu Išihara test koji se koristi za dijagnostikovanje nedostatka kolor vizije, već im boje izgledaju "življe". Nude dva tipa filter naočara – za monitore i za dnevnu svetlost.

Video igre imaju *color blind* modove kao deo prilagođavanja sadržaja korisnicima, ali često umesto da dizajniraju sa hendikepima u vidu i definišu ključne delove UI-ja u bojama pogodnim za daltoniste (poput narandžaste i plave, Slika 3.1), neki vid *color shift* filtera se samo doda na kraju razvoja igre (Slika 3.2).

Real time aplikacije za mobilne telefone obrađuju sadržaj koji snima kamera i često takođe koriste filtere za izmenu celog spektra. Bez dodavanja mašinskog učenja i prepoznavanja pojedinačnih predmeta teško je odrediti na koji način bi se promenila boja samo određenih delova slike tako da je najjednostavnije izmeniti kompletnu sliku.

Softverska rešenja se uglavnom bave izmenom kompletnog spektra slike u zavisnosti od tipa daltonizma, vodeći se idejom da isključenje problematičnih boja iz slike i zamena tih boja nekim koje su vidljive korisniku olakšava razlikovanje boja na slici. Problem se javlja jer se tada stvara konfuzija kod boja koje su ranije bile vidljive korisniku jer se menjaju i boje koje ne predstavljaju problem.

Slika 3.1 *Primer deuteranopija filtera u igri League of Legends*



Slika 3.2: *Primer tritanopija filtera u igri Overwatch*

3. REŠENJE DATE APLIKACIJE

Rešenje koje ovaj rad daje u domenu *real time* aplikacija tvrdi da je promena celog spektra boja nepotrebna jer s veštačkim kreiranjem kontrasta između jednog para delova spektra posledično kreira smanjenje kontrasta između ostalih boja. Potrebno je naći alternatini način za isticanje samo jednog dela spektra, takav koji ne obuhvata manipulisanje bojama. Po uzoru na neke bolje dizajnirane video igre i aplikacije, odlučeno je da se problematični regioni slike obeleže šrafiranjem. Na ovaj način ne menja se prikaz na koji je korisnik navikao, osim u delu koji predstavlja obeležavanje boje koju korisnik ne razlikuje.

3.1. Implementacija

Koristeći se alatima koje obezbeđuju *Apple* i *OpenCV*, napravljena je *real time* AR aplikacija za ajfon koja šrafira površine problematičnih boja za daltoniste. Svaki dostupan frejm prebacivan je iz RGB u YCrCb format radi jednostavne definicije opsega jedne boje.

Eksperimentisanjem sa različitim vrstama osvetljenja i materijala od kojih mogu biti obojeni predmeti definisani su sledeći opsezi: za crvenu boju videti (1), za zelenu (2).

$$(Y, Cr, Cb) \in [(100, 0, 182), (255, 138, 255)] \cup [(0, 70, 170), (110, 130, 255)] \quad (1)$$

$$(Y, Cr, Cb) \in [(0, 0, 0), (174, 116, 130)] \cup [(108, 0, 0), (255, 99, 136)] \quad (2)$$

Sama segmentacija boje vršena je koristeći YCrCb model boja. Orignalna sllika je čuvana, a kopija je prebacivana u ovaj model boja i segmentirana u zavisnosti od odabranog filtera. Odabirani su pikseli koji su se nalazili u zadatom opsezima.

Da bi se pokrile varijacije u nijansama korišćena su po dva opsega, a rezultujuće binarne slike su kombinovane u jednu. Nakon dobijanja segmentirane slike, učitavan je uzorak paterna koji se koristio za šrafiranje.

On se multiplicirao dovoljno puta da prekrije celu sliku i zatim bi se odsecao višak.

Ta maska je sabirana sa segmentiranom slikom i time je dobijana binarna slika gde je šrafiran samo deo slike koji nas zanima. Ovo je tada sabirano sa sačuvanom originalnom slikom i tako prikazivano na ekranu.

3.2. Rezultati

Aplikacija je isprobana na dnevnom svetlu – u senci (Slika 3.1) i direktno na suncu (Slika 3.2). Crvene su detektovane bez problema i bez mešanja sa ružičastim bojama. Što se tiče zelenih, u senci su neki od listova prelazili u plave nijanse, a na suncu u žute te nisu u potpunosti detektovani.

Sledeće je testirano kako aplikacija reaguje na slike sa monitora (Slika 3.3).

Testirano je na fotografiji utakmice gde su timovi nosili crvene i zelene dresove te su za daltoniste bili teško razlikovani. Iako je osvetljenje monitora bilo spušteno, slika je i dalje bila svetla na telefonu.

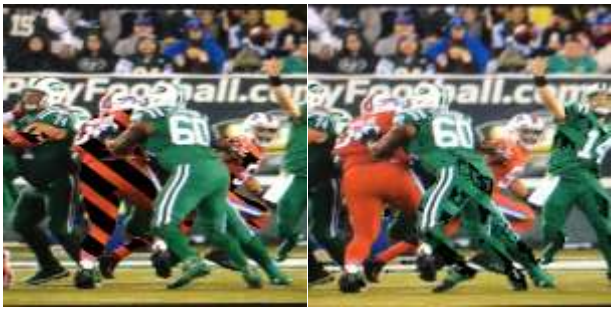
Crvena je detektovana bez problema, ali je zelena samo delimično detektovana. Ovo je delimično zbog osvetljenja, a delimično jer nijanse takođe počinju da prelaze u plavo ili sivo.

Poslednji test izvršen je na veštačkom (žutom) svetlu (Slika 3.4) gde su crvena i dve zelene olovke postavljene na crveni stolnjak.

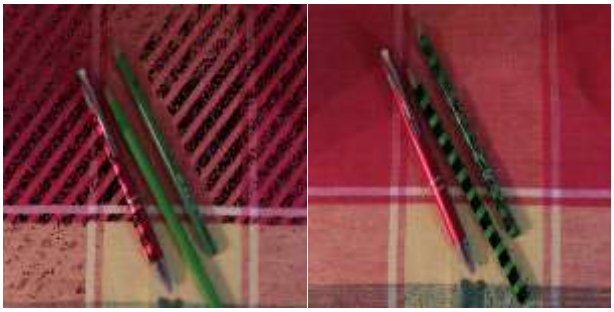
Pod veštačkim svetlom crvene nijanse postale su bliže ružičastim te su teže detektovane, dok sa zelenim nije bilo problema.

Slika 3.1 *Protan i deutan filteri u senci*

Slika 3.2 *Protan i deutan filteri na suncu*



Slika 3.3 *Protan i deutan filter na slici sa monitora*



Slika 3.4 *Protan i deutan filteri na veštačnom svetlu*

3. ZAKLJUČAK

Kao motivacija za kreiranje ove aplikacije predstavljene su situacije u kojima zavisnost od informacija koje se prenose bojom može otežati svakodnevne situacije ljudima sa daltonizmom. Pokazano je kako funkcioniše kolor vizija i koji tipovi daltonizma mogu postojati. Zatim je predstavljen koncept augmentovane realnosti i načini njene primene u svakodnevnom životu.

Objašnjeno je kako se kolor fotografija formira pomoću digitalne kamere.

Zatim su predstavljena najčešća softverska rešenja koja pokušavaju da prilagode prikaz korisnicima sa daltonizmom i objašnjeno je zašto je odlučeno da se koristi šrafiranje za obeležavanje boja na slici. Ukratko je pokazana tehnologija razvoja iOS aplikacije koja koristi kameru i oradu slike u realnom vremenu. Na kraju je data sama implementacija aplikacije i rezultati upotrebe filtera u nekoliko situacija.

Prvi problem se javio u pogledu razvoja aplikacije za iOS. Postoje obilni resursi i smernice kako jedna aplikacija izgleda. Međutim, kako *Apple* drži monopol nad alatima za razvijanje (*Xcode* i *macOS*), oni nisu na nivou konkurentnih (na primer Guglov *Android Studio* koji je *cross platform* i jednostavan za korišćenje, na nivou sa vodećim razvojnim okruženjima).

S te strane, potrebno je uložiti neko vreme da bi se sve podesilo i da bi se savladao programski jezik koji takođe samo oni koriste. S druge strane, OpenCV koji nudi alate

za samu obradu slike odlično uklapa svoju biblioteku u potpunosti sa intefejsom koji mu pruža *AVFoundation framework* te je vreme potrebno za transfer znanja minimalno.

Drugi problem predstavljala je efikasnost operacija koje su izvršavane nad svakim frejmom slike. U pogledu konkretnog rešenja koje je ovim radom predstavljeno, minimizovan je skup operacija potreban da se slika obradi, te su prikazane zadovoljavajuće performanse za prikaz u realnom vremenu. Kako je ideja bila da se predstavi alat koji će se koristiti za brzu identifikaciju objekata neke boje u odnosu sa pozadinu, šrafiranje se pokazalo kao dovoljno. Pored toga, sam uređaj pruža automatsko korigovanje ekspozicije i fokusa pa je prikaz prilično ujednačen.

Najvažniji problem se javlja pri zadavanju opsega samih boja. Za tačnu identifikaciju boja potrebno je analizirati i kontekst u kom se objekat nalazi, a ne samo vrednost piksela. Promena osvetljenja drastično utiče na boju, što ljudski mozak može da interpretira u većini slučajeva. Za tačniju segmentaciju bilo bi potrebno pratiti i pozadinu na kojoj se objekat nalazi kao i ivice objekta čime bi se dala mogućnost za odstupanje od opsega ukoliko se nalazi u okruženju piksela koju mu odgovaraju. Ovakve analize međutim zahtevale bi napredniju i dužu obradu pojedinačnih frejmova čime bi se smanjila odzivnost prikaza i povećala bi se frustracija korisnika. Na kraju su odabrani rasponi nijansi u kojima je boja čista i jasna, dok su varijacije u osvetljenju i boji svetla ignorisane, imajući u vidu činjenicu da je korisniku kroz aplikaciju dostupan i blic, odnosno lampa telefona koja se može koristiti da se osvetljenje normalizuje.

Pronalazak boljeg rešenja za varirajuće osvetljenje je najbitniji naredni korak. To bi uključivalo zahtevniju obradu frejmova koja bi mogla kao posledicu imati pad u performansama što bi dovelo do frustracije korisnika.

Osim toga, postoje unapređenja koja bi mogla biti dodata da daju korisniku veću moć nad prikazom: zumiranje slike kako bi aplikacija mogla da se koristi i kao lupa, ručno podešavajne osvetljenosti slike, čuvanje filtrirane fotografije u album sa slikama, zadavanje same šare za šrafiranje, izmena regiona problematične boje kao i zadavanje boje kojom bi problematična boja mogla da se zameni i identifikator boja koji bi na ekran ispisivao ime boje koja je u fokusu.

4. LITERATURA

[1] *Opsin genes, cone photopigments, color vision, and color blindness*, Lindsay T. Sharpe, Andrew Sockman, Herbert Jägle, Jeremy Nathans (1999) http://www.staff.uni-giessen.de/~g61088/colbook/sharpe.pdf

[2] *Handheld Augmented Reality*, D.Wagner, dissertation, *Graz University of Technology, Institute for Computer Graphics and Vision* (Oktobar 2007) http://studierstube.icg.tugraz.at/thesis/Wagner_PhDthesis_final.pdf

[3] *Digital Image Processing, 2nd Edition*, Rafael Gonzales, Richard Wods, Pearson Education (2002) http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/Digital_Image_Processing_2ndEd.pdf

Kratka biografija:

Isidora Škulec rođena je u Novom Sadu 1993. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranila je 2018.god.
kontakt: isika93@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE33Urosevic

NOVI STANDARDI PROGRAMSKOG JEZIKA JAVA

THE NEW STANDARD IN PROGRAMMING JAVA LANGUAGE

Dejan Urošević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *Tema rada jeste predstavljanje novina i poboljšanja novih verzija programskog jezika Java. U okviru rada predstavljene su novine u verzijama 8, 9 i 10. Sadržaj rada pored opisa novina sadrži i jednostavne primere u kojima se iste te novine i poboljšanja koriste.*

Ključne reči: *Java, lambda izrazi, Stream, Asinhroni pozivi, API, JavaScript, garbage collector, bytecode, modularni sistem, var, sertifikat*

Abstract – *The aim of this paper is to present the new add-ons and improvement in the newest version of the Java programming language. The content of this paper is an improvement in versions 8, 9 and 10. Beside the description this paper contains examples in which, the new add-ons and improvement are used.*

Keywords: *Java, lambda expression, Stream, asynchronous call, API, JavaScript, garbage collector, bytecode, modular system, var, certificates*

1. UVOD

Java [1] predstavlja programski jezik razvijen početkom 1990-ih godina od grupe ljudi iz *Sun Microsytem* firme koju je predvodio Džejms Goslin.. Prva verzija Jave nije bila dovoljno dobra za razvoj nekih ozbiljnijih aplikacija i tim je morao da nastavio sa razvojem kako bi napravili konkurentan proizvod. Programski jezik Java danas je jedan od najpopularnijih i najviše korišćenih, a sve to zahvaljujući njegovim karakteristikama opisanim u narednom delu rada.

2. KARAKTERISTIKE PROGRAMSKOG JEZIKA JAVA

2.1 Jednostavnost

Jednostavnost se dobila pojednostavljenjem programskog jezika C++ [2]. Iz razloga što su fajlovi bili dovoljno mali bila je potrebna prosečna mašina za izvršavanje aplikacija.

2.3 Distribuiranost

Uz pomoć Java aplikacija veoma lako se pristupa mreži i objektima na mreži, a zamorni zadaci poput otvaranja mrežne konekcije veoma su pojednostavljeni.

2.4 Robusnost

Java je programski jezik koji je napravljen da bude pouzdan na mnogo načina. Proverava ispravnost koda kako u ranoj fazi, tako i prilikom izvršavanja i omogućava ranu eliminaciju grešaka

2.5 Bezbednost

Napadi kao što su prekoračenje izvršnog steka, pristup memoriji izvan dela dodeljenog procesu ili čitanje i upisivanje datoteka bez dozvole bili su onemogućeni od samog početka. Tokom vremena dodati su i dodatni bezbednosni mehanizmi kao što je digitalni potpis klase tako da ako neko veruje autoru iste može omogućiti istoj veće privilegije na svojoj mašini.

2.6 Neutralna arhitektura

Java programski jezik se ne prevodi direktno u mašinski nego prvo se prevodi u *bytecode* [3], a ova funkcionalnost je dopustila da bude nezavistan od bilo koje arhitekture, a i dodatno je povećana bezbednost.

2.7 Prenostivost

Za razliku od drugih programskih jezika kod Jave ne postoje aspekti koji su zavisni od implementacije. Aplikacije koje se prave pomoću Jave mogu da se pokrenu na bilo kom operativnom sistemu.

2.8 Treba da se interpretira

Kako bi se omogućila prenosivost i nezavisnost Java aplikacija od operativnog sistema inženjeri su napravili Java interpreter. Java interpreter predstavlja alat koji prevodi *bytecode* u mašinski kod.

2.9 Visoke performanse

Danas se *bytecode* se prevodi u letu, delovi koda se prevode u mašinski kod prilikom izvršavanja, i time se postižu performanse koje su konkurentne sa tradicionalnim kompajliranjem, a nekada su čak i bolje.

2.10 Višenitan

Lakoća višenitnog programiranja predstavlja jedan od glavnih razloga zašto se mnogi odlučuju da koriste Javu prilikom pravljenja serverskih aplikacija. Kod Jave, kod koji vrši višenitno pozivanje je isti svuda, dok se višenitna implementacija prebacuje na platformu na kojoj je aplikacija pokrenuta ili se koristi posebna biblioteka.

2.11 Dinamičan

Ova karakteristika omogućava dodavanje koda prilikom izvršavanja, pruža potpuni uvid u strukturu i ponašanje objekata neke aplikacije kao i prikupljanje raznih informacija.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac.

3. JAVA 8

U martu 2014. godine predstavljena je 8. verzija, ujedno i poslednja koja ima dugoročnu podršku koja je planirana sve do 2025. godine. Ova verzija donela je dosta poboljšanja od kojih će neka biti nabrojana i ukratko opisana u ovom radu.

3.1 Lambda izrazi

Lambda izrazi [4] predstavljaju novi, a ujedno i najznačajniji dodatak programskom jeziku. Sintaksa lambda izraza je sledeća **parametri -> logika,** sve što je potrebno jeste da se napišu argumenti i logika koja je predstavljena kroz funkciju**.** Kako je Java tipiziran jezik promenljiva koja predstavlja lambda izraz mora imati neki tip kao vrednost. Tip koji se dodeljuje promenljivoj predstavlja funkcionalni interfejs koji je potpuno novi koncept uveden od verzije 8 i sadrži jednu i samo jednu apstraktnu metodu koja se menja prilikom implementiranja lambda izraza. U okviru ove novine dodata je i nova anotacija *@FunctionalInterface*. Prilikom pisanja promenljivih u okviru lambda izraza moraju se poštovati ista pravila kao i za ugnježdene blokove koda. Unutar tela funkcije ne mogu se naći ista imena promenljivih kao i u prethodnom delu koda, ako se tako nešto desi kod neće biti ispravan i kompajler će prikazati sintaksnu grešku. U koliko se želi pozvati samo jedna metoda u okviru lambda izraza inženjeri su napravili prečicu čija sintaksa je sledeća **objekat :: metoda**. Ovakav način pozivanja metoda može da se koristi nad statičkim metodama, metodama nekog objekta i prilikom kreiranja novog objekta uz pomoć ključne reči *new*.

3.2 Default i Static metode

U prethodnim verzijama Jave prilikom kreiranja interfejsa nije bilo moguće kreirati i metode koje imaju implementaciju. Implementirane metode u okviru intefejsa se nazivaju *default*-ne [5]. Ukoliko se implementiraju dva ili više interfejsa koji sadrže identične *default* metode prilikom poziva doći će do kolizije iz razloga što prevodilac neće znati koju tačno da pozove. Za rešavanje ovakvog problema postoje dva načina. Prvi način je da se metoda izmeni i postavi joj se nova logika, a kod drugog načina eksplicitno se navodi od kojeg interfejsa će se metoda pozvati.

3.3 Novi API za vreme

Raditi sa vremenom u Javi nikada nije bilo jednostavno, stari API je imao dosta grešaka, za neke stvari bilo je potrebno iskucati dosta linija koda kako bi sve funkcionisalo. U novom API-ju predstavljene su nove klase koje se mogu podeliti na one koje koriste vremensku zonu i one gde to nije potrebno. *LocalDate*, *LocalTime*, *LocalDateTime* predstavljaju nove klase gde se ne koristi vremenska zona, dok *ZonedDateTime* predstavlja klasu koja omogućava korišćenje vremenskih zona. Novim klasama omogućeno je lakše i jednostavnije manipulisanje sa vremenom, a pored toga nove klase su dosta otpornije na greške. U novoj verziji predstavljene su i klase *Period* i *Duration* uz pomoć kojih se predstavlja vreme koje je razumljivije čoveku. *Period* reprezentuje razliku između dva vremena u danima, mesecima ili godinama, dok *Duration* reprezentuje vrednost zasnovanu na nekom vremenu. Klasa *TemporalAdjusters* se koristi kako bi se izvele komplikovanije matematičke operacije. Pored klasa gde se ne koristi vremenska zona uvedene su i klase gde je omogućeno korišćenje istih. Klasa *java.time.ZonedId* može da se koristi sa fiksnim vrednostima za neku vremensku zonu ili navođenjem geografskog regiona. Ovom klasom se rešava problem kao što je letnje računanje vremena.

3.4 Nova OPTIONAL klasa

Nova Java dolazi sa *Optional* klasom koja se nalazi u paketu *java.util*. Pre su programeri morali da pišu dosta koda kako bi proverili da li je neki objekat jednak *null* da bi mogli da vrše neke operacije nad njim, a da te operacije ne izazovu neku grešku koja će dovesti do prestanka rada aplikacije.

3.5 Nashron, novi alat za JavaScript

U novoj verzije Jave zamenjen je stari JavaScript [6] alat sa novim, a u okviru novog dodate su i neke nove komande kao sto je *jjs*. Nova komanda omogućava da se JavaScript kod izvršava u okviru komandne linije. *Neshorn* [7] može da se koristi i u okviru Java koda gde uz pomoć *ScripEngineManager* klase JavaScript kod može da se poziva i interpretira.

3.6 Stream klasa

Omogućeno je da se operacije kao što su filtriranje i iteriranje kroz kolekcije obavljaju brže, pregledniji kod, uz pomoć stream API-ja omogućeno je i paralelno izvršavanje mnogih operacija koje do sada nisu bile moguće. Korišćenje Stream [8] API-ja u kombinaciji sa lambda izrazima podseća na funkcionalno programiranje koje se sve više koristi u okviru objektno orijentisanog programiranja.

3.7 Asinhroni pozivi

Od verzije 5 moguće je koristiti asinhrone pozive [9], ali je dosta teško raditi sa njima. U verziji 8 predstavljen je *CompletableFuture* interfejs i on predstavlja potpuno novi koncept koji objedinjuje *Future* i *CompletionStage* interfejse. *CompletionStage* predstavlja obećanje da će se poziv izvršiti, dobra stvar kod ovog interfejsa je što pruža veliki broj metoda nad kojima se može implementirati logika u odnosu na vrstu odgovora koji se dobije. *CompletableFuture* dozvoljava kreiranje i izvršavanje velikog broja asinhronih poziva, a da se pri tome ne blokira aplikacija.

3.9 Novi Base64 koder i dekoder

Klasa *Base64* [10] bila je dostupna i u prethodnim verzijama Jave, ali nije dolazila u okviru samog programskog jezika i bilo je potrebno dodati određene zavisnosti, odnosno *jar* fajlove koji su sadržali ovu klasu. Od verzije 8 ovaj, mali, ali veoma bitan API postao je sastavni deo Jave i sva njegova funkcionalnost je postala dostupna programerima u okviru paketa *java.utils.*

4. JAVA 9

U septembru 2017. godine *Oracle* je predstavio novu verziju Jave. Za razliku od prethodne verzije ova je zamišljena kao verzija u kojoj će se pokriti neki nedostaci

koji su naknadno primećeni, ali neće biti dugoročne podrške kao kod verzija 8.

4.1 G1 garbage collector

Java ima četiri *garbage collector-a* [11], verzija 8 kao podrazumevani koristi *Parallel/Throughput Collector*. U novoj verziji Jave kao podrazumevani *garbage collector* postavljen je G1 koji je prethodno predstavljen u 7 verziji programskog jezika Java. G1 je kreiran sa namerom da podrži *heap* veći od 4GB i samim tim poveća performanse, pored povećanja performansi trebalo je smanjiti broj pauza u odnosu na prethodni.

4.2 Novi HTTP 2.0 klijent

Novi API predstavljen je u okviru *incubator module-a* [12] što znači da je ovaj dodatak još uvek na ispitivanju i na osnovu povratnih informacija od klijenata biće odlučeno da li će biti i zvanično uključen u sledeću verziju Jave. HTTP 2.0 koristi *Build pattern* [13] za kreiranje zahteva. Prethodno navedeni šablon omogućava da se lako i brzo kreiraju zahtevi i više nije potrebno koristiti *InputStream* ili *Reader*. Najznačajnije klase su *HttpRequest, HttpResponse* i *HttpClient.*

4.3 Privatne metode u okviru interfejsa

Kako ne bi dolazilo do redundantnosti koda i određena logika se pisala više puta, u okviru verzije 9 omogućeno je kreiranje privatnih metoda u okviru interfejsa. Ove metode mogu da se koriste u okviru *default*-nih metoda.

4.4 API za slike sa više rezolucija

U okviru paketa *java.awt.image* kreatori Jave predstavili su novi API koji omogućava da se slike različitih rezolucije objedine u okviru jednog objekta. Klasa *MultiResolutionImage* predstavlja objekat u kojem su smeštene slike.

4.5 Modularni sistem – *Jigsaw* projekat

Modularni sistem predstavlja jedan od najvažnijih dodataka 9. verziji Jave. Jedan od glavnih razloga za uvođenje modularnosti jeste mogućnost pokretanja modularne Java Virtualne Mašine na različitim uređajima koji imaju manje memorije. Modul čini naziv, javno dostupni delovi i zavisni delovi. Prilikom kreiranja projekta koji će predstavljati modul obavezno je kreirati *module-info.java.* Sadržaj prethodno navedenog fajla je sledeći *module ime-foldera,* ime foldera predstavlja ime novog modula i po konvenciji obično se uzima ime foldera u kome se fajl nalazi.

4.6 JShell komanda (REPL)

Mnogi moderni jezici imaju funkcionalnost da u okviru terminala pišu određeni kod i izvrše ga bez prethodnog kompajliranja. Sa devetom verzijom Jave ova mogućnost je uvedena uz pomoć *JShell-a* koji predstavlja alat koji omogućava REPL funkcionalnost. *JShell* se pokreće u okviru terminala komandom *jshell.*

4.7 Dijamant operator dozvoljen za anonimne klase

U prethodnim verzijama nije bilo moguće pravljenje anonimnih klasa uz pomoć operatora dijamant (<>) nego se eksplicitno morao navesti tip koji se koristi u slučaju implementiranja apstraktnih klasa. Prilikom pokušaja da se tako nešto izvede kompajler prepoznaje grešku i ne dozvoljava pokretanje aplikacije. Sa novom verzijom aplikacije ovakva vrsta implementacije apstraktnih klasa je omogućena.

5. JAVA 10

Deseta verzija, isto kao i deveta, ne predstavlja verziju koja će imati dugoročnu podršku, ali već naredna verzija trebala bi da ima podršku sve do 2023. godine.

5.1 Paralelni *garbage collector* za G1

Deveta verzija donela je novi podrazumevani *garbage collector* G1 koji je dizajniran kako bi se smanjile pauze koje su pravljene sa prethodnim. Prilikom rada sa kolekcijama u paraleli dolazilo je do problema kada se memorija nije oslobađala dovoljno brzo i u tim trenucima bi imali situaciju kao u starijim verzijama. U novoj verziji Jave algoritam *full garbage collector-a* je paralelizovan i sada koristi isti broj niti kao i kolekcije. Kako G1 deli memoriju u jednake regione, ova promena rezultuje time da se manje prostora gubi i bolje se iskorišćava.

5.2 Definisanje promenljivih sa var tipom

Var je nova ključna reč u Java programskom jeziku koja omogućava deklarisanje promenljivih bez prethodnog navođenja tipa i predstavlja potpuno novi koncept. Lokalna promenljiva dobija tip na osnovu rezultata vrednosti koja mu je dodeljena, ako se istoj promenljivoj kasnije dodeljuje vrednost koja je različitog tipa dolazi do greške zbog loše konverzije između tipova. Prilikom deklarisanja promenljive sa *var* vrednost se mora odmah dodeliti inače kompajler vraća grešku. Ako se var promenljiva instancira sa operatorom dijamant, a da se pri tome ne navede tip neće biti moguće da se koriste metode koje su specifične za neki tip podataka iz razloga što će kompajler gledati na elemente kao *Object* tipove, ako se prilikom instanciranja navede tip sve metode će biti dostupne.

5.3 Dodavanja klasa u arhivu

Još u verziji 5 omogućeno je skladištenje klasa u arhivu i njihovo ponovno korišćenje kada je to potrebno. Glavna ideja je bila da se prilikom prvog pokretanja Java Virtualne Mašine sve klase koje se često koriste kompajliraju i skladište u arhivu na disku. Skladištenje je bilo moguće samo za klase koje su imale poverenje Jave, a nije bilo moguće dodati sopstvene klase. Od nove verzije dodata je mogućnost dodavanja sopstvenih klasa u arhivu i tom prilikom je otvorena mogućnost da se prilikom pokretanja aplikacija smanji potrebno vreme za kompajliranje.

5.4 Garbage collector interfejs

Još jedan interesantan dodatak novoj Javi koji povećava izolaciju koda za *garbage collector*-e, ali predstavlja čistiji interfejs koji je dostupan. U prethodnim verzijama moralo se objediniti znanje iz svih delova kako bi se određeni *garbage collector* izbacio iz upotrebe ili dodao novi. Novom verzijom Jave implementiranje *garbage collector-a* svelo se na implementiranje određenih metoda.

5.5 Dodavanje klasa u arhivu

Još u verziji 5 omogućeno je skladištenje klasa u arhivu i njihovo ponovno korišćenje kada je to potrebno. Glavna ideja je bila da se prilikom prvog pokretanja Java Virtualne Mašine sve klase koje se često koriste kompajliraju i skladište u arhivu na disku. Skladištenje je bilo moguće samo za klase koje su imale poverenje Jave, a nije bilo moguće dodati sopstvene klase. Od nove verzije dodata je mogućnost dodavanja sopstvenih klasa u arhivu i tom prilikom je otvorena mogućnost da se prilikom pokretanja aplikacija smanji potrebno vreme za kompajliranje.

5.6 Objedinjavanje JDK repozitorijuma u jedan

Većina novina 10. verzije Jave vezana je za sređivanje repozitorijuma, unapređenje delova koda koji je vezan za Java Virtuelnu Mašinu i generalno delova koji nisu lako dostupni krajnjem korisniku i omogućavaju pravilno izvršavanje koda koje je korisnik napisao. U prethodnoj verziji JDK (Java Development Kit) je bio podeljen u devet repozitorijuma: *root, corba, hotspot, jaxp, jaxws, jdk, langtools i nashorn*. Da se ne bi događali ovakvi problemi inženjeri su objedinili sve repozitorijume u jedan i tako omogućili lakše održavanje brže otklanjanje problema koji nastaju na Java Virtuelnoj Mašini.

5.7 Eksperimentalni Graal JIT (Just in time) kompajler

U novoj verziji postoji mogućnost da se omogući korišćenje JIT kompajlera, a njegova upotreba moguća je na Linux/x64 platformi i u zavisnosti od povratih informacija koje tim inženjera bude dobijao od korisnika postoji mogućnost da u budućnosti on postane standardan. Ovaj kompajler je deo JVM (Java Virtual Machine) kompajlera i njegov zadatak je da pomogne prilikom prevođenja bytecode-a u mašinski jezik, a sve to kako bi se omogućilo brže izvršavanje aplikacije pošto je izvršavanje bytecode-a dosta sporije od izvršavanja mašinskog.

6. ZAKLJUČAK

Vlasnici Java programskog jezika odlučili su da na svakih šest meseci predstavljaju novu verziju kako bi jezik ostao konkurentan i pre svega zanimljiv korisnicima, a i kako bi se eventualni nedostaci popunili.

Sa verzijama koje su opisane u ovom radu svakako zainteresovali okruženje i pružili dosta dobrih stvari koja do sada nisu bile prisutne. Pored novina i unapređenja koja su nove verzije donele dosta je rađeno i na bezbednosti tako da je Java jedan od najbezbednijih i najboljih jezika za pravljenje aplikacija, a to dokazuje i procenat aplikacija koje su napravljene sa njom.

7. LITERATURA

[1]https://en.wikipedia.org/wiki/Java_(programming_language) – Java programski jezik

[2] https://en.wikipedia.org/wiki/C%2B%2B – C++

[3] https://en.wikipedia.org/wiki/Java_bytecode - Bytecode

[4]https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html - Lambda izrazi
[5]https://docs.oracle.com/javase/tutorial/java/IandI/defaultmethods.html - Default-ne metode

[6] https://en.wikipedia.org/wiki/JavaScript - JavaScript
[7]https://docs.oracle.com/javase/10/nashorn/nashorn-java-api.htm#JSNUG119 – Nashron

[8]https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html - Stream
[9]https://en.wikipedia.org/wiki/Asynchronous_method_invocation - Asinhroni pozivi

[10] https://en.wikipedia.org/wiki/Base64 - Base64

[11]https://docs.oracle.com/javase/9/gctuning/garbage-first-garbage-collector.htm#JSGCT-GUID-0394E76A-1A8F-425E-A0D0-B48A3DC82B42 – G1 garbage collector
[12] http://openjdk.java.net/jeps/11 - Incubator module

[13]https://www.tutorialspoint.com/design_pattern/builder_pattern.htm - Build pattern

Kratka biografija:

Dejan Urošević - rođen 1993. godine u Rumi. Završio je srednju gimnaziju „Branko Radičević“ u Staroj Pazovi 2012. godine. Osnovne akademske studije studijskog programa Računarstvo i automatika upisao je 2012. godine na Fakultetu tehničkih nauka, Univerziteta u Novom Sadu. 2016. godine završava osnovne studije. Master akademske studije na studijskom programu Računarstvo i automatika, Dejan upisuje te iste godine (2016.), pri čemu bira modul Elektronsko poslovanje. Dejan je ispunio sve obaveze i položio sve ispite predviđene studijskim programom.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE34Maric

SOFTVERSKO REŠENJE ZA PRIKAZ GRAF MODELA

SOFTWARE SOLUTION FOR GRAPH MODEL PRESENTATION

Momir Marić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENE RAČUNARSKE NAUKE

Kratak sadržaj – *Dokument uopšteno opisuje problem, strukturu, rešenje i dobijene rezultate softverskog rešenja. Softversko rešenje je u stanju da stvori, rasporedi i prezentuje jedan matematički graf uprošćenog realnog sistema.*

Ključne reči: *Graf, Algoritmi raspoređivanja grafa, Prezentacija grafa*

Abstract – *This paper describes software solution for graph modelling, layout and visualization. The solution provides basic framework to describe and visualize a graph model of a system.*

Keywords: *Graph, Graph layout algorithms, Graph visualization.*

1. UVOD

Realni sistemi se modeluju iz dana u dan sa sve većim obimom i kompleksnošću. Mašine su u stanju da prate u stopu novonastale rastuće promene, ali ljudi i ljudska percepcija postaju preopterećeni. Glavni problem aplikacija postaje predstaviti ljudskom korisniku masivan protok količine podataka. Tabele su se pokazale najprikladnije za čuvanje i skladištenje podataka realnog sistema ali ne tako dobre za svakodnevni rad korisnika. Korisniku i ljudskom oku su umesto redova tabela teksta mnogo ugodnije vizuelne reprezentacije tih istih podataka. Vizuelne reprezentacije su bliske oku, prezentuju veliku količinu podataka i najčešće su u obliku slika, grafova, boja, ...

U toku sticanja iskustva u modelovanju i implementaciji aplikacija, uočen je mali broj komponenti za prikaz matematičkih grafova. Jednostavan matematički graf (u nastavku rada se koristi samo graf) je uređen par $G = (N, E)$, gde je N konačan, neprazan skup čvorova, a E je skup grana koje predstavljaju povezanost čvorova.

Za temu rada odabrani su dizajn i implementacija WPF komponente za stvaranje modela grafa, njegovo raspoređivanje i na kraju prezentaciju dobijenog modela. Komponenta se instancira u XAML kodu gde joj se prosleđuje model grafa kojeg prikazuje. Za ispunjavanje ciljeva komponente modelovana je implementacija grafa koja se dobija učitavanjem XML direktorijuma.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivetić, red. prof.

Raspoređivanje elemenata vrši se uz pomoć spoljašnje biblioteke. Razvijanje sopstvenog algoritma raspoređivanja je veliki obim za svrhe ovog rada i time se neće biti bavljeno. Odabrana je biblioteka MSAGL. Biblioteka podržava *Sugiyama, MDS* i *Incremental* algoritme [2]. Nakon raspoređivanja, elementi se prezentuju tako što se iscrtavaju na platnu uz pomoć XAML opisa za svaki odgovarajući element grafa.

2. MODEL GRAFA

Softversko rešenje teži ka integraciji sa većim brojem biblioteka za raspoređivanje elemenata. Za potrebe navedene integracije model grafa je implementiran koristeći interfejse. Izdvojeni su atributi čvora neophodni za raspoređivanje: granični okvir, širina, visina, koordinata tačke i reference na ivice preko kojih je jedan čvor povezan sa drugim čvorovima. Atributi ivice su reference ka prvom i drugom čvoru i opciono naziv ivice.

Granični okvir je poligon koji čvor obuhvata. Softversko rešenje podržava bilo koji poligon, ali se radi jednostavnosti implementacije i integracije sa drugim bibliotekama koristi samo pravougaonik. Za koordinatnu tačku odabrana je gornja leva tačka pravougaonika. Inicijalno svi čvorovi poseduju koordinate nulte tačke i tek nakon algoritma raspoređivanja dobijaju svoje koordinate na platnu.

Struktura grafa se dobija učitavanjem elemenata iz XML fajla. Povezanost se dobija vezama predak-potomak i referentnog elementa. Veze predak-potomak preslikavaju elemente koji su ugnježđeni u druge elemente. Veza referentnog atributa se dobija tako što se jedan element proglasi za referentni. Svaki element koji u sebi sadrži referetni element biće preslikan na čvor grafa. Na osnovu vrednosti referentnih atributa prvi čvor će biti povezan sa drugim čvorom ukoliko prvi čvor sadrži vrednost bilo kojeg elementa koja odgovara vrednosti referetnog atributa drugog čvora. Za naziv novonastale ivice se uzima naziv elementa prvog čvora.

Prethodno dobijen model grafa se pretvara u model grafa koji odgovara biblioteci koja će izvršiti raspoređivanje elemenata grafa. Preslikani model se raspoređuje prema implementaciji nekog od algoritama raspoređivanja kao što su Sugijama i MDS.

Dobro raspoređen graf sadrži minimalan broj preklapanja i presecanja svojih elemenata. Ne postoji univerzalan algoritam raspoređivanja koji je u stanju da najbolje rasporedi bilo koji model grafa realnog sistema. Softversko rešenje je moguće upotrebiti kao alat za

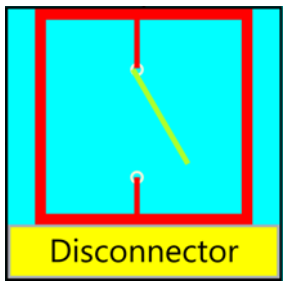
pronalaženje algoritma koji najviše odgovara jednom realnom sistemu. Pored pronalaženja odgovarajućih algoritama, softversko rešenje može da prikaže i rezultate različitih implementacija algoritama raspoređivanja.

Nakon izvršavanja algoritma raspoređivanja, model grafa se ponovo konvertuje nazad u model grafa softverskog rešenja i prikazuje.

3. PREZENTACIJA GRAFA

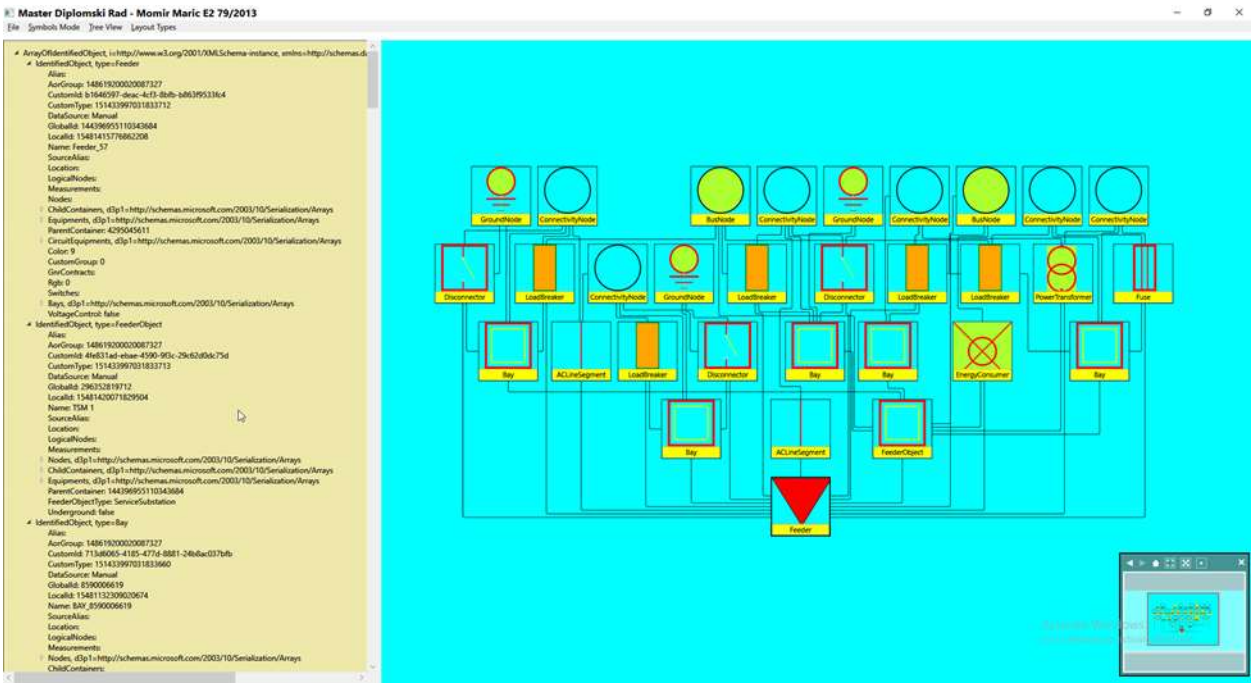
Nakon stvaranja i raspoređivanja, elementi grafa se prikazuju na platnu. Svaki element koji se prikazuje poseduje sopstveni XAML opis.

Opis se sastoji iz skupa geometrijskih oblika iscrtanih na platnu na osnovu širine, visine i koordinatne tačke čvora grafa. Osnovne oblike koje XAML podržava su: elipsa, linija, putanja, poligon, skup linija i pravougaonik [1].



Slika 1. *Primer elementa*

Na slici 1 prikazan je primer implementacije opisa jednog elementa realnog sistema. Opis se sastoji od ivičnog pravougaonika, centralnog obojenog pravougaonika i labele oznake tipa. Postoje još dve elipse i tri linije koje opisuju unutrašnje oblike.



Slika 2. *Rezultat prikaza grafa realnog sistema*

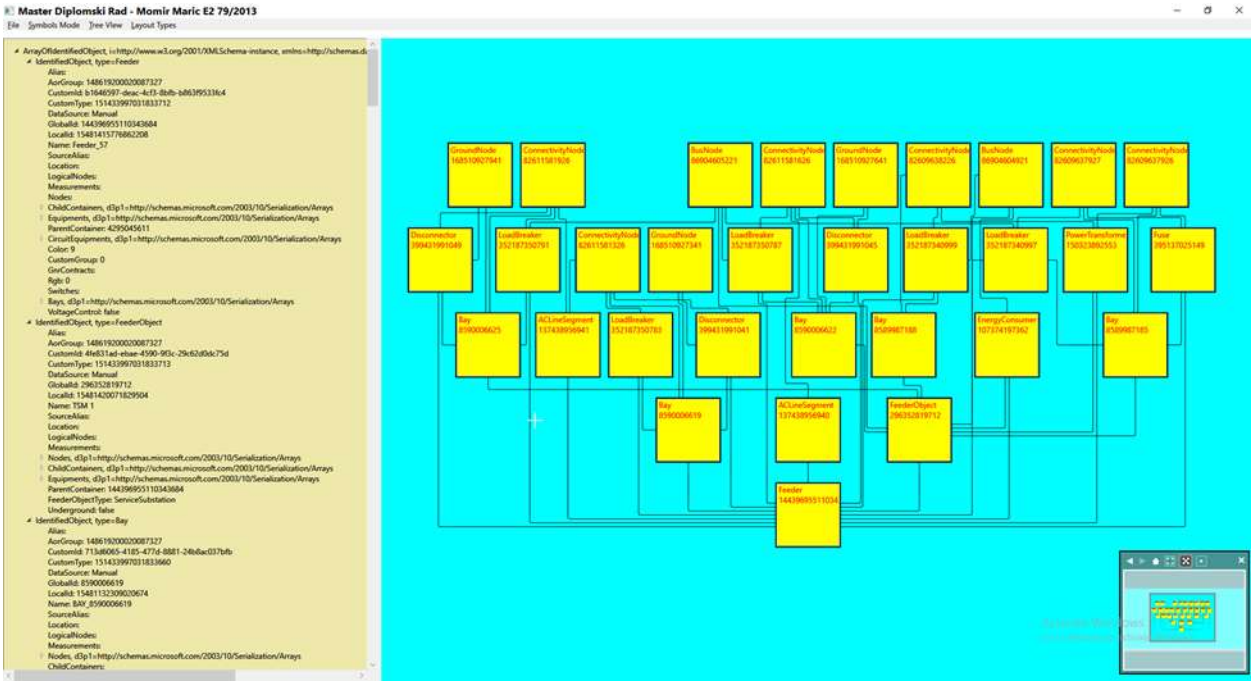
Slika 2 prikazuje konačan rezultat softverskog rešenja. Na levoj strani u obliku drveta je prikazan učitani XML direktorijum koji se prikazuje, a na desnoj strani su prikazani pojedinačni elementi raspoređeni Sugijama algoritmom. Platno je moguće pomerati i zumirati. U donjem desnom uglu nalazi se globalni pregled platna.

Softversko rešenje je u stanju da podrži prikaz više različitih realnih sistema, ili prikaz istog sistema na različlite načine u vidu iscrtavanja različitih simbola elemenata ili prikazivanja različitih povezanosti elemenata odabirom drugačijih referentnih tački. Takva osobina se podržava stvaranjem XML profila. XML profil sadrži skup XAML opisa i informaciju koji XML elementi su referentni.

Na slici 3 prikazan je implementirani tekstualni profil sa istim grafom i opcijama raspoređivanja kao na slici 2. Svi elementi se iscrtavaju univerzalnim simbolom koji se sastoji iz osnovnog graničnog okvira i teksta elementa čiji sadržaj odgovara podacima XML elementa kojeg prikazuje.

Čvorovi grafa sadrže atribute tipa na osnovog kojeg se određuje koji XAML opis će se koristiti za njegovo iscrtavanje. Ukoliko element nema grafičku reprezentaciju u odabranom XML profilu, on će biti iscrtan tekstualnim profilom.

Softversko rešenje podržava sistem plugin-a. Aplikacija se može proširiti novim algoritmima raspoređivanja i XML profilima jednostavno tako što se implementacija interfejsa smesti na odgovarajuću lokaciju, nakon čega oni postaju dostupni. Ovakva funkcionalnost se postiže posmatranjem direktorijuma za plugin-e upotrebom *FileSystemWatcher* klase i dinamičkim učitavanjem assembly.-a (.dll-a) [3]. Kada se doda nova kompajlirana klasa koja ispunjava odgovarajući interfejs, aplikacija dinamički dodaje njen sadržaj u glavni *application domain* i njene funkcionalnosti postaju dostupne.



Slika 3. *Tekst profil prikaza grafa*

4. ZAKLJUČAK

Softversko rešenje omogućuje jednostavnu integraciju prikaza grafa u sklopu C# XAML aplikacije. Dovoljno je instancirati platno u XML delu koda i proslediti model grafa. Pošto se softversko rešenje oslanja na XML model grafa i pošto se svaki objektni model može serijalizovati u vidu XML datoteke moguće je prikazati bilo koji objektno modelovani sistem.

Kvalitet prikaza grafa uveliko zavisi od dobro napravljenog modela, profila i odabira odgovarajućeg algoritma. Ukoliko ponuđeni algoritmi ne odgovaraju, softversko rešenje omogućuje proširenje i integraciju sa drugim bibliotekama kao što su *GraphViz* i *yEd*.

Pošto se elementi iscrtavaju XAML opisima, moguće je napraviti prikaze sa mešovitom i raznolikom grafikom u vidu rasterskih i vektorskih slika i oblika.

Softversko rešenje ostavlja mesta za unapređenje u vidu performansi. Nije moguće prikazati elemente realnog sistema čija brojnost prevazilazi 10 000. Komponenta za stvaranje modela grafa i njegovo raspoređivanje je ta koja je najviše usporena. Pored performansi potrebno je proširiti XAML instanciranje komponente registrovanjem specijalizovanih atributa. Primeri takvih atributa bili bi vezani za podešavanja parametara algoritama za raspoređivanje u cilju dobijanja što kvalitetnijeg i lepšeg prikaza grafa. Primeri navedenih parametara su: minimalno rastojanje elemenata, širina i visina elementa.

Pored navedenog, potrebno je i implementirati funkcionalnost ocenjivanja prikazanog grafa. Ocenjivanje grafa se može izvršiti na osnovu broja čvorova i ivica koji se preklapaju ili presecaju.

5. LITERATURA

[1] https://docs.microsoft.com/en-us/uwp/api/windows.ui.xaml.shapes (pristupljeno u septembru 2018.)
[2] https://github.com/Microsoft/automatic-graph-layout (pristupljeno u septembru 2018.)
[3] https://docs.microsoft.com/en-us/dotnet/api/system.io.filesystemwatcher?view=netframework-4.7.2 (pristupljeno u septembru 2018.)

Kratka biografija:

Momir Marić rođen je u Travniku 1990. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva odbranio je 2018.god.

kontakt: maric.momir@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE35Paunovic

WPF IMPLEMENTACIJA KONFIGURABILNE RIBBON KONTROLE

WPF IMPLEMENTATION OF THE CONFIGURABLE RIBBON CONTROL

Srđan Paunović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je prezentovan projekat u kom je implementirana komandna traka sa kontrolama, pomoću koje se ubrzava korišćenje aplikacije. Traka je korisnički interfejs koji povećava mogućnost otkrivanja aplikacionih mogućnosti i funkcija. Rad je realizovan u programskom jeziku C# korišćenjem WPF grafičkog podsistema.*

Ključne reči: *WPF, DataTemplate, CustomControls, UserControls, XML, Ribbon*

Abstract – *The document presents a project in which a command ribbon with controls is implemented, thereby speeding up the use of the application. The ribbon user interface increases discoverability of features and functions. The project was realized in the C# programming language using the graphical subsystem WPF.*

Keywords: *WPF, DataTemplate, CustomControls, UserControls, XML, Ribbon*

1. UVOD

Ribbon je komandna traka koja organizuje funkcije aplikacije u serije kartica (eng. tab) na vrhu aplikativnog prozora. Traka je korisnički interfejs koji pruža mogućnost lakšeg otkrivanja funkcionalnosti, omogućava brže učenje aplikacije i korisnici stiču osećaj veće kontrole prilikom korišćenja iste. Traka ujedno zamenjuje tradicionalne menije i trake sa alatkama. Sve funkcionalnosti su odjednom prikazane i korisnik na lakši način stiže do njih. Srodne funkcije su grupisane i postavljene u određeni tab. Implementirane funckije su prikazane preko dugmića koji su predstavljeni su ikonicom i nazivom. Naziv jasno govori o tome kakvo je njeno ponašanje, dok ikonica ilustruje to ponašanje. Prelaskom kursora preko dugmeta dobija se *ToolTip* koji detaljno opisuje koja je uloga funkcije. Zbog takvog načina prikazivanja uloge dugmeta naziv i nije obavezan, što je pogodno kod manjih dugmića. Veličina i pozicija dugmeta se određuju na osnovu toga koliko je ono važno za korišćenje aplikacije, kao i koliko se često koristi. Funkcije koje su relevantnije za aplikaciju i koje se češće koriste su predstavljene dugmićima koji su veći u odnosu na one funkcionalnosti koje se ili retko koriste ili nisu od posebnog značaja za domen problema kojim se sama aplikacija bavi.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivetić, red. prof.

Funkcije koje imaju iste ili slične osobine se mogu smestiti unutar grupe. Grupa ima svoj naziv koji bi trebalo da približno i asocijativno opisuje ponašanje funkcija unutar nje. Unutar jednog taba se, pored dugmića, mogu smestiti i grupe koje su slične na određenom nivou.. Naziv taba bi trebalo da asocira korisnika koje se to grupe i funkcije nalaze unutar njega. Kartice su raspoređene na sličan način kao što bi to bilo i u meniju. Organizovanje kartica na ovakav način pogodan je korisnicima koji su se već navikli na korišćenje starog menija i trake sa alatima, što je od velikog značaja jer korisniku omogućava komforniji, sigurniji i brži rad.

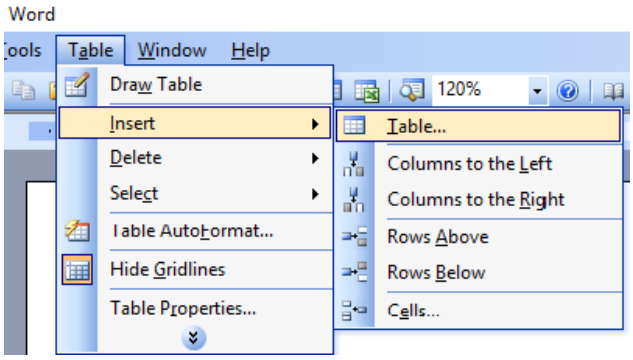
Jedna od većih prednosti *Ribbon* kontrole je ta da su korisniku prikazane samo one funkcije koje korisnik može da koristi u trenutno aktivnom prozoru. Na taj način se izbegavaju dugmići koji su konstantno prikazani u meniju i zauzimaju mesto, a nisu aktivni. Skup aktivnih kontrola se dinamički menja u zavisnosti od aktivnog konteksta u kom se aplikacija nalazi.

Traka takođe ima i funkciju sakrivanja, i u tom slučaju korisniku će se proširiti radni prostor. U sakrivenom režimu vidi se samo nazivi kartica). Klikom na određenu karticu prikazaće se *PopUp* meni koji prikazati sadržaj kartice. Na takav način se ne smanjuje radni prostor, a *PopUp* je vidljiv dokle god korisnik ne promeni fokus sa trake.

Prilikom smanjivanja veličine prozora aplikacije dolazi do transformisanja kontrola unutar trake. Tačnije, veće kontrole se smanjuju, dok se manje kontrole premeštaju u *PopUp* kontrolu unutar grupe u kojoj se prethodno kontrola nalazila. Ukoliko neka kontrola vizuelno ne može da stane u grupu kojoj pripada, premešta se u *PopUp*. Nakon što se premesti prva kontrola sakrije, pojavljuje se dugme pomoću kojeg je moguće otvoriti padajući prozor. Na taj način se pristupa skrivenim kontroloma. Traka prvo sakriva one manje prioritetne akcije, a one sa većim prioritetom pokušava da održi vidljive što je duže to moguće.

2. Ubrzavanje interakcije

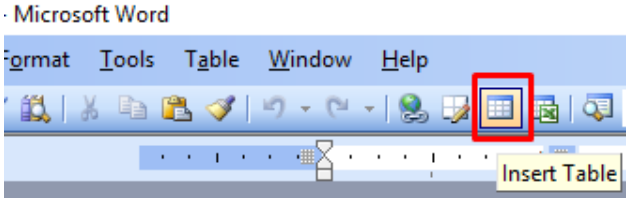
Kod starih menija sve se nalazilo sakriveno u svega par opcija, na koje se prvo moralo kliknuti kako bi se pokazao *PopUp* meni sa ostatkom mogućih akcija. Dolazak do željenih akcija zahtevao je kretanje kroz razgranato stablo menija i podmenija, što je oduzimalo prilično veliku količinu korisnikovog vremena. Primer korišćenja *Menubar*-a za umetanje nove tabele u dokument pomoću *Microsoft Word*-a verzije iz 2003. godine, prikazan je na Slici 1.



Slika 1 *Prikaz umetanja tabele iz menija u Microsoft Word-u iz 2003. godine*

Za izvršavanje pomenute akcije je bilo potrebno da korisnik prvo odabere opciju *Table* koja se nalazi na *Menubar*-u. Zatim se kursorom pomeri na *Insert* kako bi mu se pokazao ostatak funkcionalnosti, i tek na kraju odabere da ubaci tabelu. Opisani način je vremenski prilično zahtevan, iz razloga što korisnik mora da iz tri različita menija odabere željenu stavku.

Takav nedostatak je ispravio je *Toolbar*. Na *Toolbar*-u su se nalazile prečice frekventnijih akcija koje su se takođe već nalazile i u meniju. U sebi je sadržavao dugmad, koja su predstavljena ikonicom koja ilustruje ponašanje funckije. Primer umetanja tabele pomoću *Toolbar*-a prikazan je na Slici 2.



Slika 2 Prikaz umetanja tabele pomoću *Toolbar*-a u *Microsoft Word*-u iz 2003. godine

Upotreba *Toolbar*-a je drastično smanjila vreme potrebno za umetanje nove tabele, ali potencijalno samo za napredne korisnike koji već imaju iskustvo u radu sa aplikacijom ili nekom sličnom iz istog domena problema. Aplikacije koje u svom korisničkom interfejsu imaju i *Toolbar* pružaju korisnicima bar dva različita načina kako mogu da obave istu funkcionalnost: preko tradicionalnog menija ili upotrebom prečice. Međutim, ikonice su male i nekada nije jasno koja se tačno funkcionalnost krije iza određene prečice. Korisnik dok bi tražio prečicu koja odgovara funkcionalnosti koju želi da izvrši, morao je kursorom da prelazi preko svake ikonice i da čita *ToolTip*, kako bi bio siguran šta tačno ona radi. Ovo prelaženje kursorom je takođe povećavalo korisnikovo vreme potrebno za ispunjenje željene akcije. Iz tog razloga je uveden *Ribbon*.

Ribbon je nastao kao zamena za prethodno navedene kontrole. Traka izgledom podseća na obe kontrole. Kartice su raspoređene na sličan način kao što su bile glavne opcije na meniju. Svaka kartica u sebi sadrži grupe sa raznim kontrolama.

Samo je jedna kartica u jednom trenutku selektovana, a njegove kontrole su prikazane korisniku. Kontrole su veće i preglednije nego što su bile u *Toolbar*-u, što korisniku preciznije govori koja je njena svrha, i tako mu omogućava da na brži način dolazi do željenih akcija.

Primer umetanja tabele pomoću *Ribbon*-a prikazan je na Slici 3.

Slika 3 *Prikaz umetanja tabele pomoću Ribbon-a u Microsoft Word-u iz 2010. godine*

Ikonice na *Ribbon*-u su dosta veće i jasnije, što značajno doprinosi razumevanju koja je uloga akcije. Ispod dugmeta se nalazi naziv akcije, a prelaskom kursora preko ikonice se prikazuje detaljan opis koja je njena uloga.

Procena efikasnosti izvršavanja definisanog zadatka (umetanje tabele na specificarane načine) je predstavljena preko KLM-GOMS modela. U analizi je razmatran prosečan, manje vešt korisnik (40 r/m)[1]. U zavisnosti od odabranog načina korisniku bi trebalo za umetanje tabele:

1. Preko menubar-a: 7.95s
2. Preko toolbara-a: 2.65s
3. Preko ribbon-a: 2.65s

Iz priloženog jasno se vidi da će najsporije do željene akcije doći putem *Menubar*-a, a da će mu trebati jednako vremena ukoliko za to bude koristio *Toolbar* ili *Ribbon.*

3. IMPLEMENTACIJA

Traka je osmišljena da služi kao unapređenje tradicionalnih menija i traka sa alatkama. Kao takva ona je napravljena od nekoliko komponenti:

- kartice,
- grupe,
- sekcije,
- kontrole i
- *toolbar* za brzi pristup

Glavni cilj implementiranog projekta jeste da se korisnicima pruži mogućnost da, koristeći *Ribbon* kontrolu, na osnovu predefinisanih komandi (akcija) mogu sebi da kreiraju i uređuju izgled radnog prostora po svojim preferencijama. Kako je već napomenuto, sve komande su serijalizovane u XML dokument. Podsistem za čitanje XML sadržaja pročita ovaj dokument i pruža korisniku mogućnost da pročitane komande rasporedi i grupiše po svojoj volji. Kada napravi željenu konfiguraciju, korisnik je serijalizuje nazad u XML format dokumenta i sačuva je na disku računara. Kao takva, traka se dalje koristi u aplikaciji umesto menija i traki sa alatkama.

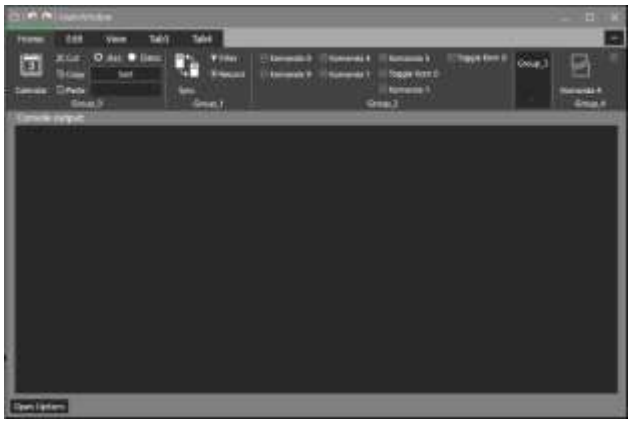
[1] Jedinica za merenje brzine kucanja. Označava broj otkucanih reči u jednom minutu.

3.1. Korisnički interfejs

Prilikom dizajniranja korisničkog interfejsa, velika pažnja je posvećena njegovom prilagođavanju tipu korisnika koji će zapravo koristiti aplikaciju. Boje, fontovi i nazivi opcija su konzistentni na svakom fragmentu korisničkog interfejsa, pružajući korisniku komforniji i brži rad. U svakom trenutku, aplikacija prikazuje optimalan skup kontrola koje su korisniku potrebne za izvršavanje željenog cilja.

Optimalan skup vidljivih komandi je kontekstno zavistan, odnosno određuje se na osnovu selektovane kartice i zadataka koji su izvršeni.

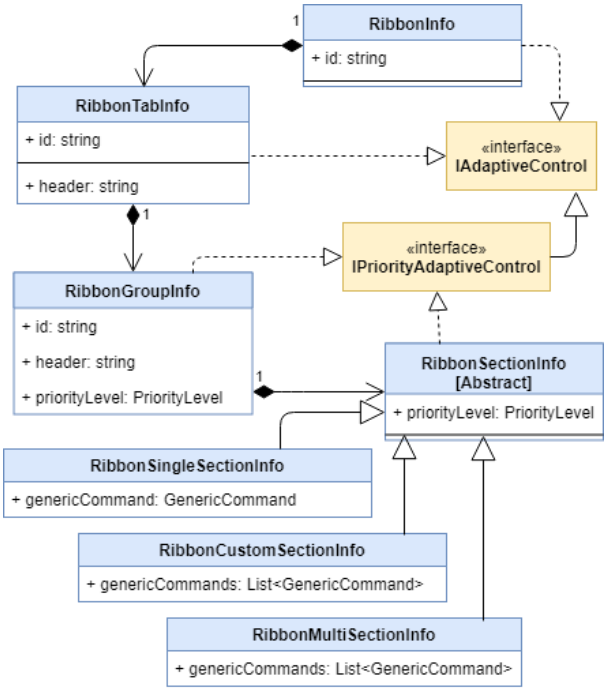
Ovaj skup je sa jedne strane dovoljno velik da zadovolji korisnikovu potrebu za obavljanjem zadatka, dok je sa druge strane dovoljno mali da korisnički interfejs ostane čist, a ne pretrpan raznolikim opcijama koje nisu relevantne u datom trenutku. Na Slici 4 prikazan je kompletan izgled glavnog prozora aplikacije.



Slika 4 *Prikaz korisničkog interfejsa aplikacije*

3.2. Model podataka

Svaka kontrola koja može biti deo glavne *Ribbon* kontrole ima i svoj odgovarujući model. Diagram klasa modela komponenti je prikazan na Slici 5.



Slika 5 *Dijagram klasa modela Ribbon aplikacije*

3.3 Implementacija kontrola

Kontrole imaju mnoge osobine koje se mogu iskoristiti za prilagođavanje njihovog izgleda. Dugme, na primer, ima opcije da promeni boju pozadine ili teksta, kartice *TabControl*-e mogu biti premeštene postavljanjem *TabStripPlacement* funkcije, i tako dalje. Ali to je sve što se može učiniti sa takvim svojstvima [1].

Šabloni (*Template*), sa druge strane, omugućavaju da se u potpunosti zameni vizuelno stablo elementa sa bilo čim što može pasti na pamet, a da sve funkcionalnosti ostanu netaknute. Šabloni, kao i mnoge druge stvari u WPF-u, nisu samo neki dodatni mehanizmi. Podrazumevani vizuelni prikazi za svaku kontrolu u WPF-u su definisani u šablonima i prilagođeni za svaku temu *Windows*-a. Izvorni kod za svaku kontrolu je potpuno odvojen od podrazumevanih vizuelnih prikaza stabla [1].

Šabloni i želja da se razdvoji vizuelni prikaz od logike su takođe razlozi da WPF kontrole ne izlažu jednostavnije osobine za podešavanje izgleda. Na primer, bilo bi lepo da postoji mogućnost da se promeni boja strelice *Expander*-a, ali siva boja se neće lepo prikazati na ljubičastoj pozadini. Ova relativno jednostvana promena može se postići samo definisanjem novog šablona za *Expander*. *On* u sebi nema definisan *ArrowBrush* ili *ArrowColor* [1].

Postoje nekoliko različitih vrsta šablona. Jedan od njih je i šablon podataka (*DataTemplate*). Šabloni podataka predstavljeni su klasom *DataTemplate* koja potiče, kao i većina šablona, od abstraktne klase *FrameworkTemplate*. Šabloni podataka prilagođavaju izgled nekog .NET objekta[2].

RibbonWindow je prozor na kom se nalazi *Ribbon* kontola. *Template* tog prozora je izmenjen kako bi se omogućile sve funkcionalnosti koje *Ribbon* ima. *Template* se sa stoji od *Border*-a koji se nalazi oko celog prozora i predstavlja deo koji omogućava promenu veličine prozora (*resize*). Unutar *ResizeBorder*-a nalaze se dva dela, *Title* deo i sadržaj tog prozora. *Title*, naslov, se nalazi na gornjem delu, kao i kod standardnog prozora. Jedini razlog zbog kog se i pravi kompletno ceo *template* prozora jeste to što *Title* deo nije moguće promeniti. Kod standarnog prozora predefinisana je pozadina *TitleBar*-a i nju dodeljuje *Windows* operativni sistem i jedino je moguće menjati tekst naslova. Zbog potrebe da se kontrola za brzi pristup nađe u *TitleBar*, potrebno je promeniti i njegov *template*. *Template* se sastoji iz ikonice prozora, kontrole za brzi pristup, naslova i dugmadi za minimizaciju, maksimizaciju i zatvaranje prozora.

QuickAccessToolbar je u osnovi *UserControl*-a koja u sebi ima *StackPanel*. *StackPanel* ima horizontalnu orijentaciju i u njemu se nalaze kontrole. Kontrole koje se mogu nalaziti su obično dugme (*Button*), isključivo dugme (*ToggleButton*) i korisnički definisana kontrola (CustomControl). *Button* i *ToggleButton* imaju posebno definisan izgled za svoju prezentaciju na *QuickAccessToolbar*-u. *CustomControl*-a će biti prikazana onako kako je i napravljena, što nekad može biti problem. Problem bi bio ukoliko je kontrola većih dimenzija nego što je prostor u koji se smešta. To bi dovelo da se određeni deo kontrole ne vidi, što gubi

smisao same kontrole. Rešenje problema će biti objašnjeno kasnije u okviru ovog poglavlja.

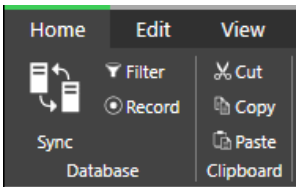
TabControl-a ima svoje kartice i svaki taj kartica ima svoj sadržaj. Taj sadržaj je *UserControl*-a koja se zove *TabContent*. *TabContent* je u osnovi *StackPanel*-a koji svoj sadržaj skladišti u horizontalnoj orijentaciji. Elementi unutar njega se nižu sa leva na desno. Ima javnu metodu *AddControl* pomoću koje se dodaju *Ribbon Group Control*-e. Metoda *AddControl* kao parametar prima *FrameworkElement* i može da primi bilo koju kontrolu koja je u osnovi *FrameworkElement*-a (kao što je i *RibbonGroupControl*).

RibbonGroupControl je takođe *UserControl*-a koja u osnovi isto ima *StackPanel* i *TextBlock*. Na isti način kao i kod *TabContol*-e, *RibbonGroupControl* ima svoju implementaciju metode *AddControl*, pomoću koje se dodaju elementi grupe. *TextBlock* se nalazi u donjem delu grupe i prikazuje naziv grupe. Grupa unutar sebe može da ima proizvoljan broj sekcija. Sekcije su *UserControl*-e i postoje tri razlicite vrste.

Vrste sekcija koje postoje: *multi*, *single* i *custom*. Svaka od njih je predstavljena odgovarajućim kontrolama. *RibbonSingleSectionControl* je kontola koja vizuelno predstavlja sekciju koja ima jednu komandu unutar sebe. U osnovi je *Grid* kontrola, jer se prikazuje samo jedan element. *Grid* podrazumevano proširi i uklopi elemente unutar sebe, i zbog toga je jako pogodan za ovakav vid prikaza. Postavljanje kontrole na sekciju je omogućeno pomoću *SetControl* metode. Kontrole koje prikazuju u *single* sekciji su ili *Button* ili *ToggleButton*, u zavisnosti da li komanda *push* ili *toggle*.

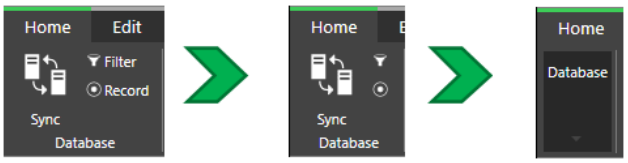
RibbonMultiSectionControl je kontrola koja može maksimalno da prikazuje do tri kontrole. Isto kao i kod *single* sekcije, kontrole mogu biti samo *Button* ili *ToggleButton*. U osnovi se nalazi *StackPanel* sa vertikalnom orijentacijom, koji niže kontrole od gore ka dole. Dodavanje omogućuje metoda *AddControl*, koja dodaje odgovarajuće kontrole u sekciju.

Button i *ToggleButton* imaju posebno definisan izgled za prikaz u odgovorajućoj sekciji. *Template* za oba dugmeta je isti, sa tim da *ToggleButton* ima mogućnost da ostane u pritisnutom stanju i da promeni boju kontrole kako bi dao indikaciju da je kontrola pritisnuta. Izgled kontrola kada se nalaze u *single* sekciji je takav da se ikonica nalazi na sredini u gornjem delu, a tekst kontrole se nalazi ispod nje. Visina takve kontrole je uvek fiksna dok se širina može menjati u zavisnosti od dužine teksta. Izgled kontrole za *multi* sekciju je takav da se ikonica nalazi sa leve strane, a pored nje sa desne strane se nalazi tekst. Visina i ovakve kontrole je fiksna i omogućava da mogu da stanu sve tri kontrole unutar sekcije. Širina je promenljiva u zavisnosti od teksta, sa tim da sve tri kontrole u sekciji zauzimaju širinu najšire kontrole. Izgled je moguće videti na Slici 6.



Slika 6 *Prikaz single i mutli sekcije sa svojim kontrolama u 2 grupe*

Adaptacija je implementirana da kontrole same brinu o tome kako će da se uklope u slobodan prostor. Prilikom skupljanja prozora u kom se *Ribbon* kontrola nalazi, kontrole se dinamički skupljaju, kolko god je to moguće, tako se uklapaju u raspoloživ prostor. Svaka kontrola od koje je sačinjen *Ribbon* ima implementiran interfejs *IAdaptiveControl.* To obezbeđuje da svaka kontrola ima implementiranu metodu za širenje i skupljanje, kao i informaciju da li je kontrola skroz skupljena ili ne. Kada kontrole dostignu maksimum svog skupljanja, i u okviru grupe ne postoji ni jedna kontrola koja može još da se skupi, cela grupa prelazi u *DropDown* kontrolu. Proces skupljanja je prikazan na Slici 7.



Slika 7. *Proces skupljanja kontrola unutar grupe*

4. ZAKLJUČAK

Projekat koji je tema ovoga rada predstavlja primenu stečenog znanja iz oblasti grafičkog korisničkog interfejsa. Posebna pažnja je posvećena dizajnu, tako da aplikacija bude dostupna i pogodna širokom spektru korisnika.

Glavni fokus ovog rada jeste na primeni stilova i šablona za pravljenje prilagođenih, nestandardnih kontrola. Ideja je bila da se što vernije napravi kontrola koja će imati sličnu namenu kao i *Microsoft*-ova *Ribbon* komandna traka. Pojedinačne kontrole unutar trake pravljene su svaka na specifičan način i prikazane su svojim šablonom. Glavna razlika, a samim tim i prednost, u odnosu na originalnu Microsoft-ovu traku jeste proizvoljna konfiguracija dugmadi prilikom rada aplikacije.

Mogućnost konfiguracije interfejsa je od posebnog značaja pre svega naprednijim korisnicima, koji mogu da prilagode svaki segment svojim potrebama i na taj način ubrzaju svoj rad. Uvođenjem ove funkcionalnosti, cela oblast dizajna se diže na jedan viši nivo i otvara čitav niz mogućnosti za dalja unapređenja.

U nekim budućim verzijama aplikacije bi bilo poželjno da postoji opcija premeštanja *DragAndDrop*-om, gde bi korisnik prevlačenjem kontrole određivao gde će kontrola da se nalazi na traci, a nova pozicija bi se automatski serijalizovala u XML dokument.

5. LITERATURA

[1] A. Nathan, WPF 4 Unleashed, Indianapolis: Sams, 2010 , pp430-432

[2] Microsoft, „C# documentation“ https://docs.microsoft.com/en-us/dotnet/csharp/

Kratka biografija:

Srđan Paunović rođen je 04.04.1994. god. u Novom Sadu. Osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu završio je 2017. godine na smeru Primenjeno softversko inženjerstvo. Iste godine, na istom fakultetu, upisuje master akademske studije**,** smer Primenjeno softversko inženjerstvo.
Kontakt: srdjan.paunovic94@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE36Stanojevic

IMPLEMENTACIJA APLIKACIJE ZA ANALIZU POTROŠNJE ELEKTRIČNE ENERGIJE

IMPLEMENTATION OF THE APPLICATION FOR POWER CONSUMPTION ANALYTICS

Vladimir Stanojević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je prikazana specifikacija i implementacija sistema za analizu potrošnje električne energije i generisanje računa za pametna brojila. Administrator je zadužen za nadgledanje funkcionalnosti i izmena sistema. Uloga pretplatnika je mogućnost detaljnog uvida u potrošnju električne energije.*

Ključne reči: *pametna brojila, merenje potrošnje električne energije, generisanje računa*

Abstract – *This paper presents the specification and the implementation of a system for analyzing measurement of electricity consumption and generating bills for smart meters. The role of administrator is monitoring all functionalities and changes in system. The role of subscribers is the possibility of a detailed insight into the consumption of electricity.*

Keywords: *smart meter, energy, measurement of electricity consumption, generating a bill*

1. UVOD

Aplikacija, čija su specifikacija i implementacija tema ovog rada bavi se analizom potrošnje energije pametnih brojila (engl. *smart meter*). Pametna brojila su povezana sa koncentratorima i transformatorima, dok su transformatori povezani sa podstanicama. Uloga koncentratora je obuhvat poslatih očitavanja od strane pametnih brojila dok transformatori i podstanice kao zadatak imaju transformaciju napona sa viših vrednosti na niže.

Aplikacija je namenjena za dva tipa korisnika. Prvi tip korisnika predstavlja potrošač koji ima potpisan ugovor sa snabdevačem električne energije, dok drugi tip korisnika predstavlja administrator sistema. Korisnik kao potrošač može imati uvid u potrošnju električne energije za svako brojilo za koje se vodi kao pretplatnik. Administrator sistema ima mogućnost uvida u informacije o pojedinačnim brojilima, transformatorima, koncentratorima i podstanicama. Cilj sistema ovog tipa je da se automatizuje proces formiranja računa za potrošače i da pruži mogućnost analize potrošnje pametnog brojila, transformatora i podstanica.

Analiza potrošnje vrši se sa ciljem uočavanja nepravilnosti u radu brojila, koji mogu biti izazvane od strane čoveka ili samog brojila. Glavni razlog uvođenja pametnih brojila i praćenja njihovog stanja ogleda se u prednosti uštede energije i održavanja.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Miroslav Zarić.

2. OPIS INFORMACIONOG SISTEMA

Koncept pametnog merenja *(smart metering)* obuhvata instalaciju pametnih brojila za rezidentne korisnike, nakon čega se vrše očitavanja, procesiranje podataka i evidenciju o potrošnji korisnika [1].

Pametna brojila mogu imati sledeće karakteristike [1]:

- trenutno ili približno trenutno registrovanje potrošnje električne energije
- nude mogućnost očitavanja brojila i lokalno i sa udaljene lokacije
- ograničavanje potrošnje na brojilu sa udaljene lokacije (u ekstremnim situacijama i ukidanje struje korisniku)
- mogućnost očitavanja i drugih tipova brojila (npr. gas, voda)

Pametna brojila su povezana sa koncentratorima i transformatorima.

Koncentratori su posrednici između pametnih brojila i sistema za prikupljanje podataka. Njihova svrha je da smanje količinu mrežnog saobraćaja u sistemu, tj, namenjeni su da sva očitavanja sa pametnih brojila za određeni period uskladište kod sebe, te da ih periodično prosleđuju sistemu za prikupljanje podataka. Često su implementirani tako da koriste radio vezu (RF) za komunikaciju sa sistemom za prikupljanje podataka, dok sa pametnim brojilima komuniciraju putem Wi-Fi mreže.

Podstanica je deo sistema za generisanje, prenos i distribuciju električne energije. Podstanice pretvaranju visoki napon u niži. Zadužene su da napone od 33kV do 11kV koliko primaju transformatori konvertuju na niže naponske nivoe.

Transformatori transformišu 11kV na 400-220V za upotrebu u domovima.

S obzirom da svu potrošnju energije evidentiraju pametna brojila, kompanije dobijaju realniji i precizniji pregled potrošnje energije u njihovom regionu. Ovaj proces omogućuje ispitivanje sumnjivih područja gde je upotreba energije veća ili manja od očekivanog.

Postoji više zainteresovanih strana za realizaciju instalacije i korišćenja pametnih brojila:

- kompanije koje se bave merenjem potrošnje smanjuju troškove očitavanja brojila
- dobavljači energije smanjuju troškove korisničke podrške i uvode nove mogućnosti korisnicima
- vlade postižu ciljeve uštede energije i efikasnosti
- krajnji korisnici povećavaju energetsku svest i smanjuju potrošnju energije i troškove energije.

Uvođenje pametnih brojila deluje kao logičan korak u svetu u kojem sve komunikacije teže da se digitalizuju i standardizuju (Internet, E-mail, SMS) i gde troškovi "digitalne inteligencije" postaju konstantno niži.

3. KORISĆENE TEHNOLOGIJE

Za razvoj rešenja neke od upotrebljenih tehnologija i alata su ASP.NET okruženje, uz *ORM Entity Framework* i *Spring Boot* okruženje, uz *Maven*. Klijentski (*front-end*) deo aplikacije bazira se na *Java Script*-u, *jQuery*-ju i *Bootstrap*-u.
Koristi se *MS SQL Server* uz *MS SQL Management Studio* za rukovanje bazom podataka. *Activiti* okruženje se koristi za kreiranje dijagrama. Neke od sledećih tehnologija opisane su u nastavku.

3.1 ASP.NET

ASP.NET je javno dostupno okruženje za izgradnju serverske strane *web* aplikacija. Razvijen je od strane *Microsoft* korporacije u cilju da se programerima omogući kreiranje dinamičkih web sajtova, web aplikacija, aplikacija i servisa [2].

3.2 Entity Framework

Entity Framework 6 (*EF6*) je objektno relacioni maper (eng. *object-relational mapper*) za *.NET okruženje* [3]. Kao *ORM*, *Entity Framework* smanjuje neusklađenost između relacionog i objektno orijentisanog pristupa. Omogućuje programerima da kreiraju aplikacije tako da interaguju sa podacima koji su smešteni u relacionu bazu podataka koristeći isključivo *.NET* objekte.

3.3 *Internet Information Services*

Internet Information Services (IIS, ranije *Internet Information Server)* je proširivi web server kreiran od strane *Microsoft-a* za korišćenje sa *Windows NT* familijom. [4]

3.4 Spring Boot

Spring je *open-source Java* platforma sa izuzetno kvalitetnom podrškom za razvoj aplikacija. Programiranje u *Springu* je vezano za koncepte *Inversion of control* i *Dependency Injection*. Ovi koncepti obezbeđuju kreiranje *loosely-coupled* objekata i njihovo automatsko povezivanje u aplikacionu celinu od strane okruženja [5].

3.5 jQuery

jQuery je *JavaScript* biblioteka i može da se koristi na više platformi. Dizajnirana je da klijentsku stranu uprosti uz korišćenje *HTML*-a [6]. Sintaksa *jQuery* biblioteke dizajnirana je da uprosti navigaciju kroz dokument odabirom *DOM* elemenata, da kreira animacije, rukuje događajima i koristi *Ajax* pozive.

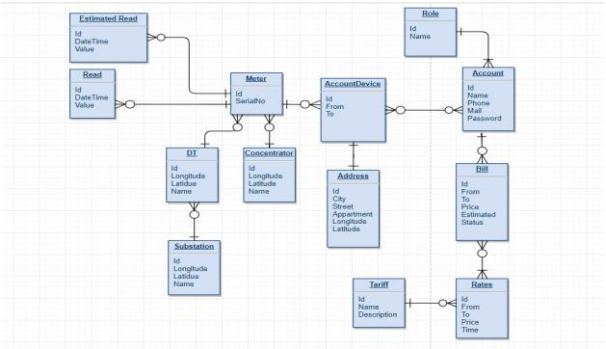
3.6 Activiti

Activiti je jednostavna *Business Process Management* (*BPM*) platforma korišćena uglavnom od strane programera i sistem administratora. Automatizacija ovih procesa omogućava brži razvoj i kompanija zauzima prednost u odnosu na konkurente. Postoji veliki broj alata koji se koriste za automatizaciju, ali *Alfresco Activiti*, ili njegova javno dostupna verzija *Activiti*, pruža dobre mogućnosti. Osnovni cilj *BPMN* jeste da omogući standardnu notaciju razumljivu svim učesnicima u procesu. Tu su uključeni poslovni analitičari koji kreiraju i menjaju procese, programeri zaduženi za implementaciju i menadžeri procesa koji procese nadgledaju i upravljaju njima. *BPMN* se predstavlja kao zajednički jezik koji savladava komunikacioni jaz koji se često javlja između dizajna poslovnog procesa i njegove implementacije.

4. IMPLEMENTACIJA I DEMONSTRACIJA

Kao početni korak u procesu implementacije softverskog rešenja neophodno je projektovati sistem i modelovati bazu podataka sa kojom će se rukovati. Razvoj konceptualnog modela omogućava razumevanje entitetea na kojima se zasniva rad sistema Iz ovog modela tj. iz njegovih relacija entiteti-veze dobija se relaciona baza podataka.

Na slici 1. prikazan je konceptualni dijagram sistema.



Slika 1. *Konceptualni dijagram*

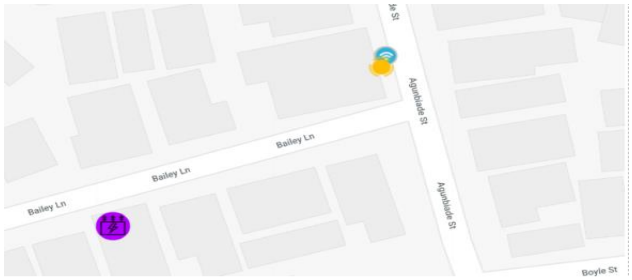
U ovoj sekciji takođe su prikazan i procesi implementirani u sistemu i to su:

- Grafički prikaz pozicija podstanica, transformatora, koncentratora i pametnih brojila iz sistema pomoću Google mapa
- Prikaz osnovnih informacija za odabranu podstanicu sa mape
- Prikaz povezanih pametnih brojila za odabrani koncentrator
- Prikaz osnovnih informacija za odabrano brojilo sa mape
- Tabelarni prikaz potrošnje svih brojila na osnovu izabranog transformatora i perioda očitavanja
- Prikaz svih brojila za koje je pretplatnik vezan
- Prikaz potrošnje odabranog brojila za pretplatnika
- Pretraga elemenata na mapi

4.1 Grafički prikaz pozicija podstanica, transformatora, koncentratora i pametnih brojila iz sistema pomoću Google mapa

Funkcionalnost grafičkog prikaza podrazumeva učitavanje *Google* mape na kojoj su prikazane lokacije podstanica, transformatora, koncentratora i pametnih brojila.

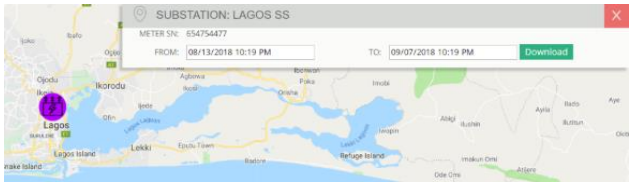
Odabirom jedne od ponuđenih podstanica, na mapi se prikazuju svi transformatori sa kojim je odabrana podstanica povezana. Primer grafičkog prikaza podstanice sa povezanim transformatorima nalazi se na slici 2.



Slika 2. *Grafički prikaz podstanice i povezanih transformatora*

4.2 Prikaz osnovnih informacija za odabranu podstanicu sa mape

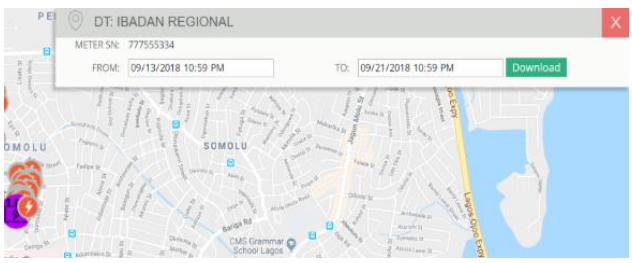
Odabirom podstanice objekta na mapi dobija se informacija o imenu, geografskoj širini i dužini i vremenu zadnjeg pristiglog očitavanja podstanice. Osim prikaza, omogućena je i pretraga potrošnje po datumu. Pretraga po datumu prikazuje potrošnju podstanice za odabrani vremenski period, ali i potrošnju povezanih transformatora. Informacije o podstanici i prikaz podstanice dat je na slici 3.



Slika 3. *Osnovne informacije o podstanici i pretraga potrošnje*

4.3 Prikaz osnovnih informacija za odabrani transformator sa mape

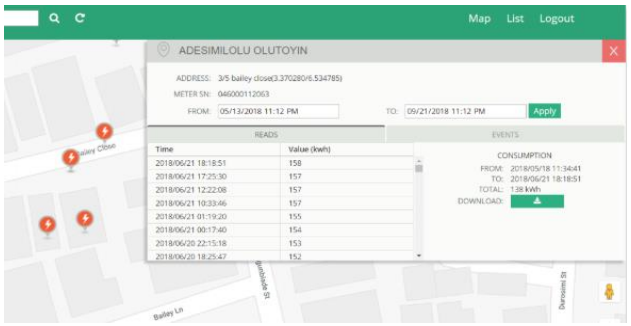
Informacije koje se prikazuju administratoru su ime, geografska širina i dužina i vreme zadnjeg pristiglog očitavanja transformatora.
Osim informacija, administratoru se prikazuju i pametna brojila povezana sa odabranim transformatorom na mapi.
Omogućena je i pretraga po datumu, koja za rezultat daje potrošnju transformatora i brojila vezanih za odabranu stanicu. Forma za pretragu prikazana je na slici 4.



Slika 4. *Osnovne informacije o odabranom transformatoru i pretraga potrošnje*

4.4 Prikaz osnovnih informacija za odabrano brojilo sa mape

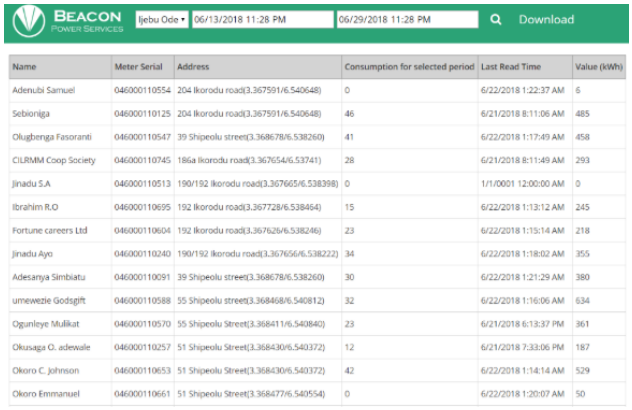
Kada se na mapi formiraju objekti koji reprezentuju pametna brojila, administrator poseduje mogućnost da odabere prikaz informacija o pametnom brojilu. Na taj način dobija informacije o imenu, geografskoj širini i dužini i vremenu zadnjeg pristiglog očitavanja pametnog brojila. Informacije vezane za određeni datum dobijaju se pretragom po datumu. Prikaz i rezultat pretrage nalazi se na slici 5.



Slika 5. *Prikaz osnovnih informacija o pametnom brojilu pretraga i očitavanja brojila*

4.5 Tabelarni prikaz potrošnje svih brojila na osnovu izabranog transformatora i perioda očitavanja

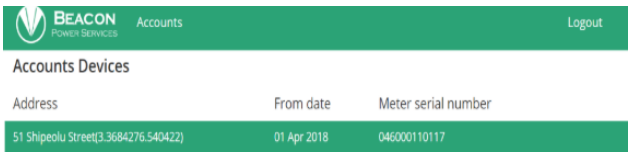
Na osnovu odabranog transformatora i perioda očitavanja vrši se prikaz brojila. U tabelarnom prikazu, koji se nalazi odvojeno od mape, prikazan je rezultat ove pretrage. Tabela se sastoji od serijskog broja brojila, imena pretplatnika, vremena poslednjeg pristiglog očitavanja, ukupne potrošnje brojila od postavljanja, ukupne potrošnja brojila u navedenom periodu i naziva transformatora kojem pripada. Omogućena je opcija prikaza brojila za sve transformatore i primer je dat na slici 6.



Slika 6. *Tabelarni prikaz potrošnje svih brojila odabranog transformatora*

4.6 Prikaz svih brojila za koje je pretplatnik vezan

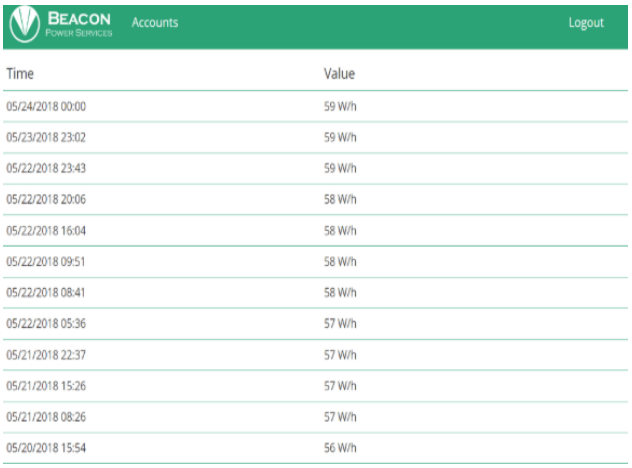
Nakon prijave na sistem, pretplatniku se formira tabela sa brojilima na koje je pretplaćen. Na ovom mestu ima uvid u datum početka pretplate, serijski broj brojila i adresu na koju je brojilo postavljeno. Prikaz informacija o brojilima pretplatnika dat je na slici 7.



Slika 7. *Prikaz brojila koji pripadaju* pretplatniku

4.7 Prikaz potrošnje odabranog brojila za pretplatnika

Pretplatnik ima mogućnost da, nakon pregleda tabele sa brojilima na koje je pretplaćen, odabere prikaz očitavanja za odabrano brojilo. Potrošnja koja je prikazana predstavlja ukupnu potrošnju od ugradnje brojila do trenutku kada je merenje izvršeno i prikazana je na slici 8.



Slika 8. *Tabelarni prikaz potrošnje odabranog brojila*

4.8 Pretraga elemenata na mapi

Administrator sistema ima mogućnost da pretražuje elemente koji se nalaze na mapi po bilo kom njihobom atributu.

5. ZAKLJUČAK

Glavni razlog za uvođenje *smart meteringa* ogleda se u mogućnosti uštede energije i sigurnosti snabdevanja. Mnogo država iz EU i izvan nje već su uključene u projekte sa pametnim brojilima na demonstracionom nivou ili više od toga.

Pokazano je da je smart metering tehnički izvodljiv i da nema prepreka za uvođenje pametnih brojila. Korisnici pametnih brojila u svakom trenutku mogu videti svoju trenutnu potrošnju te mogu prilagoditi potrošnju u cilju smanjenja troškova.

6. LITERATURA

[1] Rob van Gerwen, Saskia Jaarsma and Rob Wilhite, KEMA (2016),„Smart metering”[Internet] Dostupno na :https://idc-online.com/technical_references/pdfs/electrical_engineering/Smart_Metering.pdf

[2] Microsoft. May 14, 2013. "ASP.NET is part of a great open source .NET community".

[3] Krill, Paul (20 July 2012). "Microsoft open-sources Entity Framework".

[4] TechNet. Microsoft. "Running IIS 6.0 as an Application Server (IIS 6.0)"

[5] Almir Pehratović (2017). „Uvod u SpringFramework“ [Internet] Dostupno na: https://www.info.ba/ostalo/specijal/8291/uvod-u-spring-framework

[6] The jQuery Project. "jQuery: The write less, do more, JavaScript library"

Kratka biografija:

Vladimir Stanojević rođen je 17.09.1994. godine u Ljuboviji. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva - Računarstvo i automatika odbranio je 2018. god.

Kontakt: vladimir94@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/01BE28Mirkovic

STATIČKA ANALIZA KODA ZASNOVANA NA UPOTREBI *ROSLYN* KOMPAJLERA

STATIC CODE ANALYSIS BASED ON USAGE OF *ROSLYN* COMPILER

Željka Aleksić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – Primenjeno softversko inženjerstvo

Kratak sadržaj – *Statička analiza koda predstavlja proces analize izvornog ili binarnog koda softvera. Njen cilj jeste da se otkriju potencijalne slabosti softvera bez potrebe da se on prethodno izvrši, kao i da se proveri kompatibilnost stila sa postojećim preporukama i opštim standardima. Ovaj rad razmatra mogućnosti .NET kompajlerske platforme pod nazivom Roslyn i način na koji se upotrebom Roslyn API-ja može kreirati specijalizovani alat za statičku analizu koda zasnovan na smernicama .NET standarda za programski jezik C#.*

Ključne reči: *Statička analiza koda, Roslyn, kompajler, C# standard kodiranja*

Abstract – *Static code analysis is a process of analyzing software's source or binary code. The aim of analysis is to check the compliance to specific coding rules and discover potential vulnerabilities of software without actually executing the code. This work analyzes the possibilities of .NET compiler platform named Roslyn and describes the implementation of a custom tool for static code analysis based on .NET coding standard for C#.*

Keywords: *Static code analysis, Roslyn, compiler, C# coding standard*

1. UVOD

Sa porastom obima softverskog proizvoda, raste i potreba za automatizovanim alatima koji pomažu u otkrivanju potencijalnih problema u kodu. Statička analiza koda predstavlja prvi korak u analizi koda, pre puštanja u rad samog softvera. Statička analiza koda može da se primenjuje u ranim fazama razvoja softvera i za razliku od testiranja, može da se primeni i na nedovršenim izvornim kodom. Štaviše, ona otkriva koren problema dok dinamičko testiranje ističe samo posledice.

Na tržištu postoji veliki broj alata namenjenih statičkoj analizi koda kao što su *FxCop*, *StyleCop* i *Resharper*. Njihova zajednička karakteristika jeste da moraju da parsiraju izvorni kod pre same analize, što je u stvari zadatak kompajlera. Međutim, tradicionalni kompajleri se mogu opisati kao crne kutije, gde su poznati ulazi, u vidu izvornog koda, i izlazi, u obliku objektnog fajla, dok sam proces razumevanja koda u toku kompajliranja ostaje nepoznat krajnjim korisnicima kompajlera.

.NET kompajler platforma, poznata pod nazivom Roslyn, je grupa open-source kompajlera i API-ja za analizu koda namenjenih C# i Visual Basic programskim jezicima. Glavna motivacija za razvoj Roslyn-a je upravo otvaranje crnih kutija kako bi se krajnjim korisnicima, inženjerima, omogućio pristup bogatstvu informacija o kodu kojima kompajleri raspolažu. Na ovaj način kompajler postaje API. Upotrebom .NET kompajler platforme (*Roslyn*), eliminiše se potreba da se izvorni kod parsira više puta, tako da se alati zasnovani na ovoj platformi mogu koncentrisati isključivo na analizu izvornog koda.

Cilj ovog rada jeste upoznavanje sa principima i mogućnostima .NET kompajler platforme, poznate pod nazivom *Roslyn* kao i kreiranje specijalizovanog alata za statičku analizu koda upotrebom *Roslyn* kompajlera. Implementacija alata je zasnovana na smernicama C# standarda za pisanje koda. To su smernice vezane za imenovanje, formatiranje i struktuiranje izvornog koda.

2. STATIČKA ANALIZA KODA

Statička analiza koda predstavlja proces evaluacije sistema ili komponente na osnovu njene strukture, sadržaja ili dokumentacije. Ona adresira slabosti programskog koda koje mogu dovesti do ranjivosti sistema. Njen zadatak je donošenje zaključaka o mogućem ponašanju programa bez potrebe da se on prethodno izvrši. Cilj analize jeste provera stepena podudaranja izvornog koda sa specificiranim pravilima i identifikacija delova koda koji potencijalno mogu biti štetni.

Statička analiza koda može biti manuelna, kao što je revizija koda ili može biti automatizovana upotrebom softverskih alata. Automatizovani alati uglavnom vrše analizu izvornog koda. Međutim. postoji i manji broj alata koji vrše analizu nad kompajliranim, binarnim kodom [1].

2.1. Analiza izvornog koda

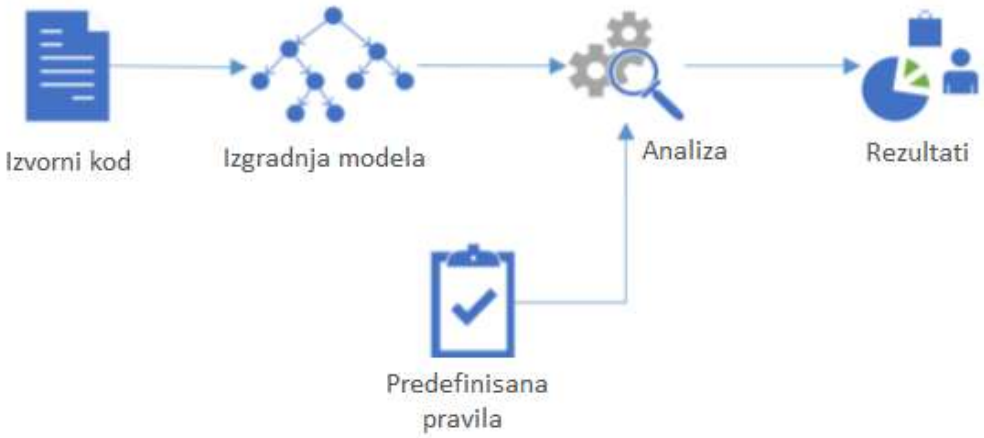
Kako bi alat mogao da analizira izvorni kod, potrebno ga je razumeti. Stoga prvi korak jeste da se kreira strukturni model koji prestavlja izvorni kod. Formiranje modela u statičkoj analizi koda je postupak sličan prvom delu procesa kompajliranja koji obuhvata parsiranje, leksičku i semantičku analizu koji rezultira sintaksnim stablom izvornog koda.

Nakon formiranja modela, sledeći korak je sama analiza. Mogu se koristiti mnogi algoritmi za analizu koda i uobičajeno je da se oni kombinuju u jedno rešenje. Algoritmi su veoma često izvedeni iz tehnika koje i sam kompajler primenjuje.

Predefinisana pravila predstavljaju početnu tačku statičke analize. Ona su podjednako važna, ako ne i važnija, od heuristika i algoritama na kojima se zasniva sama implementacija alata.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Branislav Atlagić, docent.

Način na koji se vrši izveštavanje rezultata statičke analize koda je od velike važnosti. Ukoliko izveštaj nije razumljiv, sam rezultat statičke analize je neupotrebljiv, jer će nerazumevanje dovesti do toga da se greška ignoriše, ili još gore, označi kao lažno pozitivna. Dobar alat za statičku analizu koda treba da obezbedi grupisanje i sortiranje rezultata, kao i da izbegne izveštavanje nerelevantnih rezultata. Svaki detektovani problem, mora biti praćen detaljnim opisom problema, nivoom kritičnosti, kao i preporukama kako bi problem mogao biti rešen [2]. Struktura procesa statičke analize je prikazana na slici 1.



Slika 1. *Proces statičke analize [2]*

2.2. Problemi kojima se bavi statička analiza koda

Postoje različiti tipovi alata za statičku analizu koda koji se bave različitim oblastima problematike: provera sintakse, stila, razumevanje programa, pronalaženje grešaka.

Provera sintakse je funkcionalnost koja je integrisana u svaki kompajler. Sintaksna pravila su uglavnom izvedena iz samog programskog jezika, te se na njih oslanjaju alati za proveru sintakse.

Alati za proveru stila vrše proveru kompatibilnosti izvornog koda sa definisanim pravilima za imenovanje, komentarisanje, kao i generalnu strukturu programa koja će najviše doprineti čitljivosti i održavanju programa.

Pojedini alati imaju za cilj da obezbede razumevanje programa na visokom nivou. Najefikasniji su kad su integrisani u razvojno okruženje (IDE) gde mogu da podrže *go to implementation* ili *find all references* funkcionalnosti, ili čak automatsku modifikaciju izvornog koda u vidu promene naziva promenljive.

Svrha alata za pronalaženje grešaka jeste da se predstave uobičajene greške u kodu. Oni upozoravaju na delove programa koji su u skladu sa specifikacijama programskog jezika ali ne izražavaju nameru inženjera.

3. NIVOI *ROSYLN* API SPREGE

Roslyn kompajlerski API se sastoji iz dva osnovna nivoa, kompajlerski API i API namenjen radnom prostoru. Pored ova dva nivoa, postoji i funkcionalni API višeg nivoa. Kompajlerski API pruža objektni model koji predstavlja rezultat sintaksne i semantičke analize procesa kompajliranja. Ovaj sloj sadrži presek stanja referenci, opcija vezanih za kompajliranje i izvornog koda koji nije podložan modifikaciji. Ovaj sloj je nezavisan od bilo koje *Visual Studio* komponente i kao takav se može koristiti i kao nezavisna aplikacija.

Sloj radnog prostora je početna tačka statičke analize i refaktorisanja koda. Unutar ovog sloja, API za radni prostor je zadužen za organizovanje informacija o projektima u okviru *solution*-a u jedan objektni model. API za radni prostor nudi direktan pristup objektnim modelima sloja kompajlera kao što su izvorni tekst, sintaksno stablo, semantički model i kompilacija bez potrebe za parsiranjem fajlova ili promenom podešavanja konfiguracije.

Funkcionalni API se oslanja na kompajlerski sloj i sloj namenjen radnom prostoru i dizajniran je tako da nudi API namenjen refaktorisanju i automatskim ispravkama koda [3].

3.1. Dijagnostifikacija problema

Implementacija logike statičke analize koda može da se sastoji od više pravila za detekciju domenski specifičnih grešaka i problema u kodu. Svako pravilo je potrebno definisati kao tip *DiagnosticDescriptor*.

Detektovani problemi se korisniku prijavljuju kao upozorenje. Linija ili deo koda koji nije u skladu sa definisanim pravilom će biti podvučen zelenom linijom. Prebacivanjem fokusa na podvučen deo koda, otvara se dodatni prozor sa porukom obaveštenja o grešci.

Code Fix je brza akcija koja nudi predlog mogućeg rešenja za dijagnostifikovane probleme pronađene u kodu od strane alata za statičku analizu koda. Može se primeniti pritiskom na prečicu Crtl+. koja je sastavni deo *Visual Studio*-a. Pre same primene ponuđenih izmena, programer može da vidi kako bi te izmene izgledale

4. IMPLEMENTACIJA SPECIJALIZOVANOG ALATA ZA STATIČKU ANALIZU KODA

Implementacija specijalizovanog alata za statičku analizu koda zasnovana je na smernicama za pisanje koda po

.NET standardu namenjenom programskom jeziku C# [5]. Ovaj dokument opisuje pravila i preporuke za razvoj aplikacija i biblioteka pisanih C# programskim jezikom. Cilj jeste da se definišu opšte smernice kako bi se obezbedila konzistencija u stilu i formatiranju izvornog koda, kao i da bi se izbegle uobičajene greške softver inženjera. Dokument je podeljen na 4 oblasti: konvencija imenovanja, stil pisanja koda, upotreba jezika i dizajn objektnog modela. Cilj ovog rada jeste implementacija specijalizovanog alata za statičku analizu koda koji vrši proveru prilagođenosti izvornog koda nekim od smernica .NET standarda iz oblasti konvencije imenovanja, stila pisanja koda, kao i upotrebe jezika. Ukoliko je to moguće, alat treba da ponudi i odgovarajuću ispravku dijagnostifikovane greške.

Implementirani alat se može uključiti u *Visual Studio* razvojno okruženje kao ekstenzija. Kreiranjem projekta tipa *.Vsix*, kreira se i *Visual Studio Extension* instalacija koja kada se pokrene, instalira novi alat u razvojno okruženje. Analiza i *code fix* postaju dostupni onog trenutka kada se dodaju u *Visual Studio*. Sama analiza se odvija u vreme pisanja koda. Ukoliko neki deo izvornog koda nije napisan u skladu sa pravilima definisanim analizom, generisaće se upozorenje u vidu zelene podvučene linije spornog dela koda sa porukom obaveštenja o grešci. Kada se prebaci fokus na označeni deo izvornog koda, otvara se padajući meni sa *code fix*-om i prikazom primene ispravke. Klikom na ponuđeni *code fix* izmeniće se sporni deo koda u skladu sa ponuđenom ispravkom, tako da se nakon primene ispravke, upozorenje više neće prikazivati.

Karakteristika alata za statičku analizu koda jeste da se uglavnom sastoje od kratkih provera poštovanja pravila koje se mogu implementirati u svega par linija koda. Pošto se dijagnostike pravila razlikuju po jedinstvenom identifikatoru, implemetirana pravila su definisana identifikatorom *CR00X* gde oznaka *CR* označava *custom rule*, dok *X* predstavlja redni broj pravila. Prilikom ispisa dijagnostike, prikazuje se identifikator pravila i poruka, dok se otvaranjem dodatnog prozora prikazuje i opis dijagnostike.

Stil pisanja koda je najveći uzročnik nekonzistentnosti koda. Većina inženjere tokom godina i radnog okruženja i zahteva kompanije u kojoj radi stiče određeni stil pisanja koji je neretko u suprotnosti od preferenci njegovih kolega. Međutim, konzistentan format i organizacija koda su ključni prilikom kreiranja održivog koda. .NET standard nudi niz smernica kako bi se kreirao čitljiv i konzistentan softver koji se lako razume i lako održava.

Kako bi se povećala čitljivost koda, .NET predlaže da se koriste vitičaste zagrada kada god je to moguće, i uvek u novom redu, što povećava preglednost koda i stvara utisak hijerarhije. Za iskaze uslova koji nisu duži od jedne linije, u C# programskom jeziku nije potrebno navoditi vitičaste zagrade, ali s obzirom na to da njihova upotreba dovodi do lakšeg razumevanja koda, preporučljivo je da se koriste. Pravilo *CR003* je namenjeno proveri iskaza uslova *if*, tačnije proveri da li je neki od njegovih čvorova dece tipa *Block*.

Kako bi se obezbedila modularnost koda, preporuka je da se po jednom fajlu definiše najviše jedan imenski prostor i jedna klasa. Definisanje više klasa u istom fajlu smanjuje čitljivost koda i otežava njegovo razumevanje. Pravilo CR004 proverava da li se unutar jednog *CompilationUnit*-a nalazi više od jednog *NamespaceDeclaration*-a ili *ClassDeclaration*-a. Ukoliko se detektuje narušavanje ove sugestije, generisaće se upozorenje u vidu zelene podvučene linije *CompilationUnit*-a u celosti.

Komentari su elementi izvornog koda koji ne utiču na funkcionalnost izvršavanja programa. *Roslyn* kompajler ih svrstava u sintaksne trivije koje se vezuju za tokene. Njihova uloga može biti u dokumentovanju metode i klase, međutim, prekomerna upotreba komentara može loše da utiče na razumevanje koda i oteža navigaciju i manipulaciju nad izvornim kodom. Za komentare važi preporuka da treba da se izbegava upotreba takozvanih *flowerbox* komentara koji se navode između znakova /* i */. Umesto ovog tipa komentara, savetuje se primena jednostavnijeg oblika u vidu // ili ///. Pravilo CR005 prepoznaje *flowerbox* komentare i generiše upozorenje sa prikladnom porukom koja sugeriše format u kojem treba da se navode komentari.

Opšta praksa za navođenje imenskih prostora jeste da naziv imenskog prostora treba da sadrži i naziv projekta, i nazive svih foldera u kojem se posmatrani .cs fajl nalazi. Prilikom kreiranja novog .cs fajla u projektu, imenski prostor će u nazivu sadržati i naziv projekta i nazive fajlova u kojima se nalazi. Međutim, ukoliko dođe do izmene u organizaciji foldera ili promene lokacije .cs fajla, potrebno je promeniti i naziv imenskog prostora.

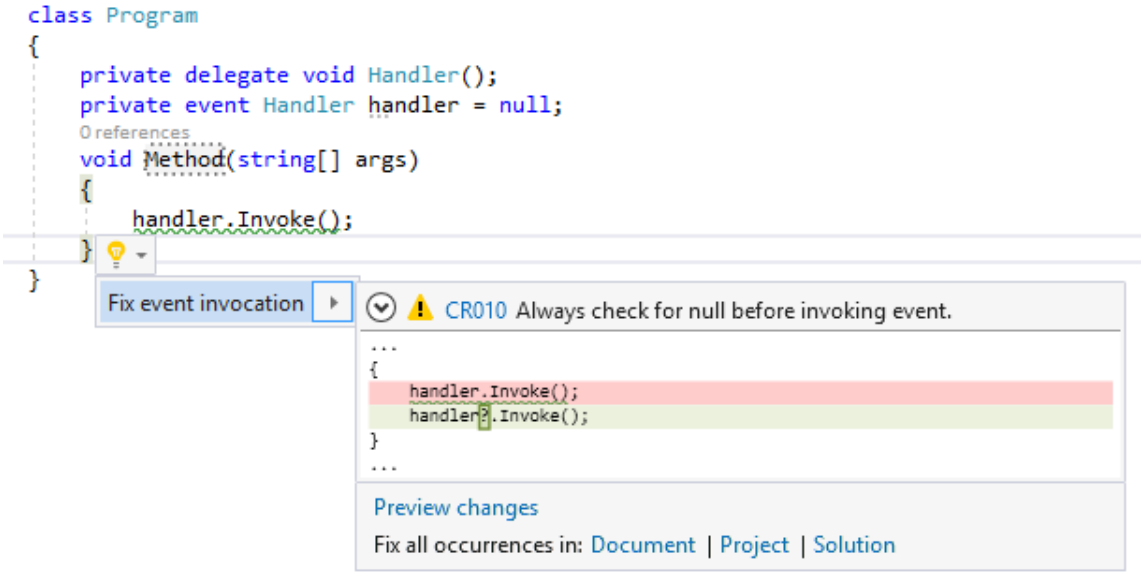
Pored ovog opšteg pravila, mnoge kompanije poseduju specijalizovano pravilo za navođenje imenskih prostora koje glasi da je potrebno da se u nazivu imenskog prostora, pored naziva svih hijerarhijski nadređenih fajlova, nalazi i naziv *assembly*-ja. Kako bi se promena naziva imenskog prostora automatizovala, implementirana je analiza i *code fix* koji vrše proveru i ispravku naziva na poziv prečice *Ctrl+.* .

Ključna reč *base*, programskog jezika C#, koristi se za pristup članovima bazne klase iz izvedene klase, dok ključna reč *this* predstavlja referencu na trenutnu instancu klase. Smernice za pravilno korišćenje konstrukcija ovog programskog jezika nalažu da se ove dve ključne reči koriste samo u okviru konstruktora ili metoda označenih sa *override* modifikatorom. Kako bi se izvršila provera upotrebe pomenutih ključnih reči potrebno je da se pristupi njihovim roditeljskim čvorovima sintaksnog stabla. Potrebno je da neki od čvorova hijerarhijski viših on *BaseKeyword* ili *ThisKeyword* budu tipa *ConstructorDeclaration* ili *MethodDeclaration*, dodatno, u slučaju da je reč o metodi, potrebno je da neki od modifikatora metode ima vrednost *override*. Ukoliko prethodni uslovi nisu zadovoljeni analiza će rezultirati upozorenjem.

Događaji omogućuju da se neka klasa ili objekat obaveste kada se desi nešto od interesa. Publikacija događaja se vrši pozivom metode *Invoke()*, ali pre toga je potrebno proveriti vrednost promenljive tipa *event* koja ne sme da ima vrednost *null* kako bi publikacija mogla da se izvrši (slika 2). Postoje dva načina povere vrednosti promenljive tipa *event*. Jedan način jeste da se poziv *Invoke()* metode navodi unutar uslovnog iskaza *if* koji vrši proveru vrednosti promenljive, dok drugi pristup zahteva upotrebu uslovnog *null* operatora (?). Pravilo *CR010* vrši proveru

poštovanja navedene smernice. Analiza se sastoji iz dva dela, prvog koji pronalazi promenljive tipa *event* kako bi se sačuvao njegov identifikator, i drugog koji vrši proveru poziva metode *Invoke()*.

Ponovo je potrebno pristupiti roditeljskim čvorovima *InvocationExpression*-a, te proveriti da li je neki od njih tipa *ConditionalAccessExpression* (?) ili *IfStatement*. U slučaju kada je poziv *Invoke()* metode enkapsuliran u uslovni iskaz *if*, treba izvršiti i proveru samog uslova, kako bi sa sigurnošću utvrdili da se odnosi tačno na promenljivu tipa *event*.



Slika 2: *Prikaz dijagnostike i preporuke ispravke dela izvornog koda koji nije u skladu sa smernicama C# standarda*

5. ZAKLJUČAK

Glavi cilj rada jeste istraživanje mogućnosti *Roslyn* kompajlera sa aspekta statičke analize koda. Činjenica, da je *Roslyn* relativno nova platforma i da postoji malo alata zasnovanih na njegovoj upotrebi, glavni je razlog ovog rada i implementacije specijalizovanog alata za statičku analizu koda. *Roslyn* omogućuje pisanje alata koji nude opciju analize izvornog koda već pri samom pisanju koda, gde se softver inženjeru ukazuje potencijalni problemi već u ranim fazama razvoja.

Pored toga, upotrebom intuitivnih *Roslyn* API-ja, koji izlažu veliki broj metoda za rukovanje objektima koji reprezentuju delove izvornog koda, može da se kreira alat koji će vršiti proveru kompatibilnosti koda sa pravilima nekog standarda ili kompanijskim pravilima. Pravila mogu biti različitog tipa, od konvencije imenovanja do načina upotrebe programskih iskaza.

Pravac daljeg razvoja jeste proširenje skupa implementiranih pravila. Do sada je akcenat bio na stilskim pravilima, dok je cilj budućeg razvoja unapređenje alata sa dodatnim pravilima namenjenim detektovanju šablona koji se često pojavljuju u izvornom kodu, a čija zamena može da se izvrši na automatizovan način sa ciljem poboljšanja performansi ili refaktorisanja koda.

Reč je o naprednoj analizi koda koja može da detektuje beskonačne petlje, rekurzivne pozive metoda i duboko ugnježdene petlje, a čijim izmenama se mogu poboljšati performanse softvera. Takođe, može da se upotrebi za izračunavanje raznih metrika kvaliteta, kao što su proširivost, kompleksnost, održivost, hijerarhija nasleđivanja klasa i dupliciranje koda.

6. LITERATURA

[1] owasp.org, 2017, *Static Code Analysis*, [online] dostupno na: https://www.owasp.org/index.php/Static_Code_Analysis [posećeno 4 Sep. 2018]

[2] Chess, B and West, J 2007, *Secure programming with Static Analysis*, Addison-Wesley, Boston

[3] github.com, 2018, *.NET Compiler Platform ("Roslyn") Overview*, [online] dostupno na: https://github.com/dotnet/roslyn/wiki/Roslyn%20Overview [posećeno 6 Sep. 2018]

[4] Chess, B and West, J 2007, *Secure programming with Static Analysis*, Addison-Wesley, Boston

[5] Hunt, L 2007, *C# coding standard for .NET*

Kratka biografija:

Željka Aleksić rođena je u Rijeci 1994. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Sigurnost i bezbednost u sistemu pametnih brojila odbranila je 2017.god.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4:621.395
DOI: https://doi.org/10.24867/01BE38Gogic

DDOS NAPAD NA SCADA PODSISTEM SMART GRID-A

DDOS ATTACK ON SCADA SMART GRID SUBSYSTEM

Damjan Gogić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu su analizirane posledice DDoS napada na SCADA podsistem Smart Grid-a. Za potrebe testiranja razvijena je DERMS (Distributed Energi Resource Management System) aplikacija za upravljanje distribuiranim izvorima električne energije. U okviru aplikacije su razvijene tri različite arhitekture SCADA podsistema nad kojima su izvršavani DDoS napadi. Pomoću DDoS simulatora izvršava se napad na SCADA podsistem. Tokom napada su merene performanse sistema u zavisnosti od broja napadača i različitih arhitektura SCADA podsistema.*

Ključne reči: *Distributed Denial of Service, Supervisory Control and Data Acquisition, Smart Grid*

Abstract – *This paper analyzes the consequences of DDoS attacks against SCADA subsystems in Smart Grids. For the needs of testing, a small DERMS (Distributed Energi Resource Management System) application for managing distributed power sources has been developed. Within the application, three different SCADA subsystem architectures were developed over which DDoS attacks were performed. The SCADA subsystem was attacked by a DDoS simulator. During the attack, system performance was measured for various numbers of attackers and different SCADA architectures.*

Keywords: *Distributed Denial of Service, Supervisory Control and Data Acquisition, Smart Grid*

1. UVOD

Inteligentna, pametna ili mreža budućnosti (engl. *Smart Grid*), predstavlja unapređenu verziju tradicionalnih elektroenergetskih sistema dvadesetog veka. Korišćenje novih informacionih tehnologija, omogućilo je razvoj tradicionalnih elektroenergetskih sistema u cilju efikasnije, pouzdanije i brže isporuke električne energije. Zavisnost čovečanstva od električne energije je u konstantnom porastu i iz tih razloga elektroenergetski sistemi moraju da se razvijaju velikom brzinom.

Jedan od nadolazećih izazova sa kojima mora da se suoči *Smart Grid* jeste mogućnost sajber napada prouzrokovani

povećanjem površine napada usled upotrebe modernih komunikacionih i informacionih tehnologija.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Lendak Imre, red. prof.

Motivi sajber napada na pametnu mrežu mogu biti ekonomski, dokazivanje, nezadovoljstvo zaposlenih, industrijska špijunaža, terorizam itd.

Zanemarivanje informacione bezbednosti u kritičnim infrastrukturama (u koje spada i pametna mreža), može da omogući napadaču da se lako infiltrira u sistem, dobije pristup softveru, koji upravlja mrežom, i na taj način destabilizuje sistem i nanese ozbiljne financijske gubitke preduzeću ali i potrošačima. Pored financijskih gubitaka, napadač može da ugrozi i privatnost potrošača, prikupljanjem i zloupotrebom njihovih podataka.

2. TEORIJSKE OSNOVE

2.1. *Distributed Network Protocol* (DNP3)

U svetu računarskih mreža, protokoli definišu pravila na osnovu kojih uređaji komuniciraju jedni sa drugima. DNP3 predstavlja skup komunikacionih protokola koji se koriste za komunikaciju između dve tačke u sistemima za automatizaciju industrijskih postrojenja [4]. Protokol je razvijen od strane Westronic kompanije (danas u vlasništvu GE Harris kompanije) tokom 1990ih. Arhitektura DNP3 protokola obuhvata četiri sloja. To su fizički, *DataLink*, transportni i aplikacioni sloj.

Fizički sloj je odgovoran za razmenu poruka preko fizičkih medijuma kao što su radio, satelit, bakar, itd [4]. Pored toga, odnosi se i na stanje medija (slobodno ili zauzeto), kao i na sinhronizaciju između strana koje razmenjuju poruke.

***DataLink* sloj** [4] je zadužen za obezbeđivanje dvosmerne komunikacije između aplikacija i uređaja u polju.

Transportni sloj je zadužen da rastavi poruku sa aplikativnog sloja u jedinice podataka veličine DataLink *frame*-a, kako bi bili podesniji za transport.

Aplikacioni sloj specificira DNP3 poruke zahteva i odgovora, definiše uloge *master* i *outstation* uređaja [4].

2.2. Bezbednost

Ukoliko se posmatra bezbednost informacionih sistema tri osnovne komponente zaštite su poverljivost, integritet i raspoloživost. Izraz koji se najčešće upotrebljava za ove tri komponente je CIA triada (**C** – *confidentality*, **I** – *integrity* i **A** – *aviability*).

Poverljivost [5] je komponenta koja obezbeđuje da informacija nije otkrivena neautorizovanim licima, entitetima ili procesima.

Integritet (celovitost, neokrnjenost) [5] znači održavanje i osiguravanje tačnosti i celovitosti informacije tokom celog životnog ciklusa.

Raspoloživost (dostupnost) [5] se može definisati kao mogućnost da se do informacije ili resursa dođe na pouzdan način u vremenski prihvatljivom intervalu.

2.4.1. DoS/DDoS

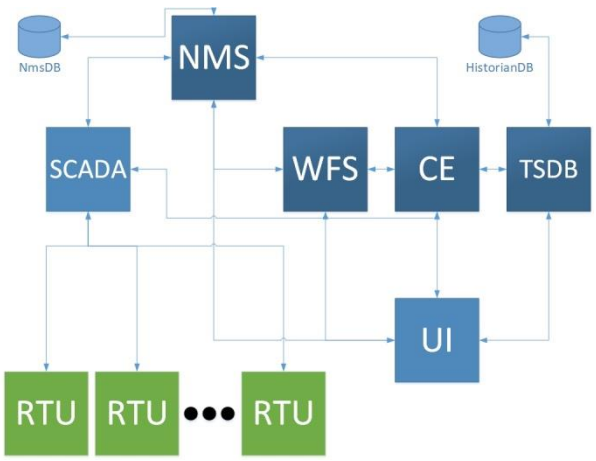
DoS (engl. *Denial of Service*) predstavlja napad koji narušava komponentu raspoloživosti. On se izvodi tako što se ciljanom sistemu šalje velika količina neželjenih poruka, koju on mora da obradi. Na taj način se iscrpljuju resursi sistema i on postaje nedostupan za sve korisnike njegovih usluga.

Ukoliko se sa jednog računara napada određeni sistem, servis ili proces radi se o jednostavnom DoS napadu, međutim, ako se napad izvodi sa više prostorno udaljenih računara tada se taj napad naziva distribuiran DoS (engl. *Distributed Denial of Service*) i često se zapisuje kao DDoS.

Prilikom primene DoS napada koristi se samo jedna IP adresa, dok se za DDoS napad koristi veliki broj zaraženih računara. Vlasnici računara sa kojih se izvodi DDoS napad često nisu svesni postojanja zlonamernog koda na njihovim računarima, koji se nazivaju *bot*-ovima a mreža takvih računara se naziva *botnet*. Zlonameran kod se širi putem internete često preko trojanaca, crva, *phishing email*-a, itd. Takav kod ne nanosi štetu računaru domaćinu, njegov zadatak je da ostane neprimetan i da služi za potrebe DDoS napada. Mrežom zaraženih računara upravlja napadač preko upravljačkog servera (engl. Command & Control – C&C server).

3. ARHITEKTURA SISTEMA

Razvijena aplikacija predstavlja *Distributed Energy Resource Management System (DERMS)* softversku platformu za nadgledanje, menadžment i regulaciju distribuiranih izvora energije. Pored toga, moguća je agregacija DER-ova po različitim kriterijumima (lokacija, tehnologija). Omogućeno je nadgledanje stanja rada svake grupe, kao i prognoza rada za svaku grupe Upravljanje radom svake grupe omogućeno je zadavanjem globalnog *setpoint*-a koji se automatski, optimalno raspoređuje na pojedinačne elemente grupe. Na slici 1 prikazana je arhitektura sistema.



Slika 1. *Arhitektura DERMS sistema*

Calculation Engine (CE) – Prikuplja analogna merenja preko SCADA komponente, generiše nadzorno upravljačke komande na zahtev korisnika sistema.

Network Model Service (NMS) - Osnovna namena NMS komponente je da ostalim komponentama u sistemu obezbedi pristup mrežnom modelu EES kroz odgovarajuće interfejse.

Remote Terminal Unit (RTU) - RTU se koristi za prikupljanje izmerenih analognih i digitalnih podataka sa uređaja u polju, i kao komandno komunikacioni kontroler za uređaje u polju.

Supervisory Control And Data Acquisition (SCADA) komponente obavlja funkcije nadzora i upravljanja fizičkom procesima realnom vremenu.

Time Series Database (TSDB) - TSDB predstavlja komponentu koji radi prikupljanje merenja očitanih sa SCADA servisa i njihovo skladištenje, pored toga radi agregaciju podataka na satnom, dnevnom, mesečnom i godišnjem nivou.

User Interface (UI) - Nadgledanje, menadžment i regulacija distribuiranih izvora električne energije omogućena je preko UI komponente.

Weather Forecast Service (WFS) – Namena WFS komponente je da ostalim komponentama omogući pristup vremenskoj prognozi. Pored trenutne vremenske prognoze servis obezbeđuje i predviđanje za sedam dana unapred sa rezolucijom od jednog sata.

4. REALIZACIJA NAPADA

Puna integracija elektroenergetskih sistema sa naprednim informacionim i komunikacionim tehnologijama, otvorila je mogućnost sajber (engl. *cyber*) napada. Od mogućih vektora napada ovde će se prikazati napad iz procesnog postrojenja.

4.1. Arhitektura napadnute komponente (SCADA)

4.1.1. SCADA komponenta bez redova

U SCADA komponenti bez redova čekanja, podaci iz svih RTU-ova se momentalno obrađuju u okviru programske niti dodeljene tom RTU-u. Svaka nit je zadužen za prihvatanje poruka, obrada, proveru alarma i obaveštavanje svih zainteresovanih strana o pristiglim promenama.

4.1.2. SCADA komponenta sa jednim redom

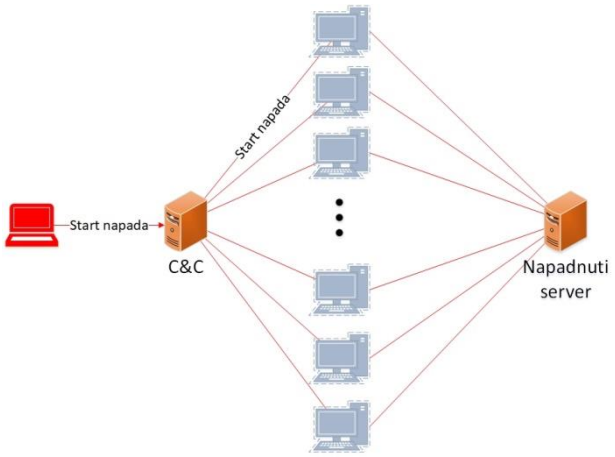
Implementacija SCADA komponente sa jednim redom zahteva da se za svaki RTU veže jedan programska nit koji prihvata podatke. Nakon što prihvati podatke, smešta ih u zajednički red čekanja i daje signal programskoj niti koji je zadužen za procesiranje podataka.

4.1.3. SCADA komponenta sa više redova

Karakteristično, za ovaj način implementacije, je da se za svaki RTU vezuje dva programske niti i jedan red čekanja. Jedna programska nit je zadužen prikupljanje podataka i njihovo smeštanje u red. Druga nit povlači i obrađuje podatke. Unapređenje se ogleda u tome što su podaci od svakog RTU-a obrađuju nezavisno jedan od drugog ali se ne može uticati na redosled izvršavanja.

4.2. Implementacija DDoS simulatora

Implementacija DDoS simulatora se oslanja na klasičnu centralizovanu klijent-server arhitekturu *botnet* mreže, gde se svaki *bot*, prilikom svoje aktivacije, javlja C&C serveru (Slika 2).



Slika 2. *Klasična klijent-server botnet arhitektura*

Komunikacije između njih je ostvarena putem WCF (engl. *Windows Communication Foundation*) *duplex* kanala. Svaki *bot* zna na kojoj se adresi nalazi C&C i prilikom svog startovanja uspostavlja WCF komunikaciju sa njim. Nakon uspostavljene komunikacije, C&C zna za tog *bot*-a i preuzima kontrolu nad njim tako što mu može komandovati da započne napad ili da ga zaustavi ukoliko je napad u toku. Prilikom komandovanja, *bot* dobija instrukciju kojom brzinom i na koliko adresa će da šalje podatke. U trenutku kada RTU izgubi konekciju sa SCADA komponentom, *bot* se aktivira i postavi na njegovo mesto ponašajući se kao regularni RTU. Poruke koje on šalje su u ispravnom DNP3 formatu, tako da SCADA ni u kom slučaju ne može da zna da li komunicira sa regularnim RTU-om ili malicioznim aplikacijom.

4.3. Scenariji i rezultati DDoS napada

U cilju da se simulira rad *Smart Grid* sistem, korišćena je *Microsoft Azure Cloud Computing Platform*-a. Na taj način dobijena je prostorna distribuiranost i nezavisnost resursa procesnog postrojenja i sistema koji njime upravlja. U okviru svakog scenarija, procesno postrojenje je simulirano sa 10 RTU-ova. Svaki RTU se izvršavao na zasebnoj virtualnoj mašini.

4.3.1. Scenario 1 – Uobičajen rad *Smart Grid* aplikacije

U okviru prvog scenarija se simulira uobičajen rad *Smart Grid* aplikacije gde SCADA komunicira se procesnim postrojenjem u kojem se ne nalazi ni jedan RTU zaražen zlonamernim kodom.

4.3.1.1. Rezultati testiranja

Rezultati testiranja izvršeni za sve tri arhitekture SCADA komponente, su identični. Zauzeće RAM memorije se je bilo statično, bez tendencije rasta, oko 42% od ukupne memorije računara na kom se sistem izvršavao. Procesor je prosečno, tokom svih pet časova rada, bio na 78% sa manjim varijacijama. Performanse sistema se mogu videti u tabeli 1.

Tabela 1. *Performanse uobičajenog rada sistema*

SCADA arhitektura	Zauzeće procesora [%]	Zauzeće RAM memorije [%]	Vreme ispada sistema
SCADA bez redova	78%	42%	-
SCADA sa jednim redom	78%	42%	-
SCADA sa više redova	78%	42%	-

4.3.2. Scenario 2 – Jedan RTU zaražen zlonamernim kodom

U okviru drugog scenarija se prikazuju se posledice po *Smart Grid* ukoliko se jedan RTU zarazi zlonamernim kodom.

4.3.2.1. Rezultati testiranja

Performanse rada sistema tokom napada prikazane su u tabeli 2. SCADA komponenta bez redova čekanja se najlošije pokazala tokom ovog testa. Jedino je u tom slučaju došlo do ispada sistema, gde je prva komponenta koja je otkazala bila *CalculationEngine*, nakon toga, otkazala je i SCADA.

Oko dva sata je bilo potrebno da se destabilizuje sistem. Međutim, na osnovu logova se moglo primetiti da, i pre nego što je otkazala, SCADA nije obrađivala sve pristigle podatke već je dolazilo do gubitka informacija.

Analizom logova, kreiranih tokom testiranja SCADA komponente sa jednim redom čekanja, može se primetiti da je već posle 10 minuta, red, u koji se smeštaju pristigli podaci, sadržao 1000 poruka koje su čekale na obradu. Taj broj se do kraja testa neprestano povećavao. Opterećenje procesora bilo oko 78%, dok se zauzeće RAM memorije blago povećavalo, što bi na duže staze dovelo do potrošnje resursa.

Implementacija SCADA komponente sa više redova pokazala je neznatno bolje rezultate u odnosu na prethodne dve arhitekture. Nakon 10 minuta rada sistema, redovi, koji prihvataju podatke sa RTU-ova sadržali su između 10 i 50 neobrađenih poruka.

Programske niti koje su zadužene za obradu podataka nisu uspevale da obrade pristigle podatke tako da se kašnjenje, na obradu, povećavalo.

Tabela 2. *Performanse sistema tokom napada*

SCADA arhitektura	Zauzeće procesora [%]	Zauzeće RAM memorije [%]	Vreme ispada sistema
SCADA bez redova	85%	39%	1h 56 min
SCADA sa jednim redom	78%	42%	-
SCADA sa više redova	70%	40%	-

4.3.3. Scenario 3 – Polovina RTU-ova zaražena zlonamernim kodom

Treći scenarijo prikazuje ponašanje SCADA komponente u slučaju kada je polovina RTU-ova, koji su pod njenim nadzorom, zaražena zlonamernim kodom.

4.3.3.1. Rezultati testiranja

Ponašanje *Smart Grid* sistema, kada se u okviru njega izvršava SCADA bez redova, je identično kao u prethodnom testu. Zauzeće procesora je iznosilo 88% dok se potrošnja memorije nije menjala. Nakon 1 sata i 50 minuta sistem je prestao sa radom.

Kada se posmatra arhitektura sistema sa jednim redom, broj zaostalih poruka, posle 10 minuta, iznosio je 5259. Posle pet sati testiranja, taj broj se povećao na 145118. Razlika u odnosu na prethodni slučaj je i ta što je nakon 2 sata došlo do prestanka rada *CalculationEngina*, međutim, on nije prouzrokovao otkaz čitavog sistema.

Tokom napada na SCADA komponentu sa više redova procesor je bio opterećen 98%. Zauzeće memorije se, do kraja testa, povećalo za 14%. Kao i u prethodnom slučaju, *CalculationEngine* je prestao sa radom nakon 2 sata od početka napada, posle toga je ručno podignut i sistem je nastavio sa radom.

Rezultati testiranja prikazani su u okviru tabele 3.

Tabela 3. *Performanse sistema tokom napada*

SCADA arhitektura	Zauzeće procesora [%]	Zauzeće RAM memorije [%]	Vreme ispada sistema
SCADA bez redova	88%	39%	1h 50 min
SCADA sa jednim redom	78%	Od 42% do 50%	2h (jedna komponenta)
SCADA sa više redova	98%	Od 40% do 54%	2h (jedna komponenta)

4.3.4. Scenario 4 – Nema regularnih RTU-ova

U okviru ovog scenarija nema ni jednog regularnog RTU-a, već su svi zaraženi zlonamernim kodom.

4.3.4.1. Rezultati testiranja

Rezultati SCADA komponente bez redova se nisu drastično menjali u odnosu na prethodne testove. Vreme kada je ceo sistem prestao sa radom je bilo oko 1 sat i 35 minuta. Zauzeće procesora u početku je iznosilo 92%.

Kada je u pitanju SCADA komponenta sa jednim redom procesor je prosečno bio opterećen 95%. Do kraja testa, potrošeno je dodatnih 38% RAM memorije. Nakon 4 sata i 16 minuta, aplikacija je prestala sa radom.

U slučaju SCADA komponente sa više redova, prosečno, 1500 poruka je bilo u svakom redu nakon 10 minuta rada. Da kraja, svaki red imao 36000 neobrađenih poruka. Procesor je konstantno radio sa 100% svojih mogućnosti dok je potrošnja RAM memorije bila u konstantnom porastu od 22%. Nakon 4h 20 minuta došlo je do ispada čitavog sistema.

Rezultati testiranja prikazani su u okviru tabele 4.

Tabela 4. *Performanse sistema tokom napada*

SCADA arhitektura	Zauzeće procesora [%]	Zauzeće RAM memorije [%]	Vreme ispada sistema
SCADA bez redova	98%	40%	1h 35min
SCADA sa jednim redom	95%	Od 39% do 77%	4h 16 min
SCADA sa više redova	100%	Od 40% do 62%	4h 10 min

3. ZAKLJUČAK

Elektroenergetski sistemi su tokom svog razvoja prešli put od tradicionalnih do naprednih (*Smart Grid*) sistema.

Jedan od osnovnih problema sa kojima se suočava pametna mreža jeste informaciona bezbednost. Kompleksnost sistema i integracija sa internetom, dvosmerna komunikacija sa velikim brojem distriburianih uređaja za prikupljanje podataka, povećali su ranjivost samog sistema.

Iz rezultata dobijeni tokom testiranja može se izvesti zaključak: bez obzira na arhitekturu SCADA komponente, dovoljan je samo jedan napadnut RTU u procesnom postrojenju kao izvor (D)DoS napada, da bi sistem postao nedostupan uz uslov da ne postoje implementirani bezbednosni mehanizmi na SCADA komponenti.

Testiranje sprovedeno u ovom radu predstavlja dobru osnovu za unapređenje bezbednosti *Smart Grid* sistema, kao i za razumevanje DDoS napada iz procesnog postrojenja preko DNP3 protokola.

4. LITERATURA

[1] J. P Arriaga., H. Rudnick, M. Rivier, „Chapter 1: Electric Energy Systems. An Overview.“, Boca Raton. CRC Press, 2009.

[2] V. Šiljkut, „Upravljanje potrošnjom u inteligentnim energetskim mrežama sa varijabilnom proizvodnjom.“ Beograd, 2014/15

[3] “Position Paper on Smart Grids”, European Regulators Group for Electricity and Gas (ERGEG) - Conclusions Paper, Jun 2010.

[4] M. Patwardhan, „DNP3: Security and scalability analysis.“ Sacramento, California State University, 2012

[5] M. Petković, „Prilog razvoju metode za detekciju napada ometanjem usluga na internetu“, Novi Sad. RS: Faculty of technical sciences, University of Novi Sad, 2018

Kratka biografija:

Damjan Gogić rođen je 1994. godine u Sanskom Mostu, Bosna i Hercegovina. Završio je srednju ekonomsku školu „Srednja ekonomska škola“ u Somboru 2013. godine. Osnovne akademske studije završio 2017. godine na Fakultet tehničih nauka u Novom Sadu.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/01BE39Batinic

ANALIZA PRIMENE AMI SISTEMA U CLOUD OKRUŽENJU

APPLICATION ANALYSIS OF AMI SYSTEM IN CLOUD ENVIRONMENT

Bratislav Batinić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisana je AMI arhitektura i predstavljen je problem rukovanja velikom količinom podataka. Nakon toga, prezentovano je soft-versko rešenje AMI sistema, gde se deo aplikacija nalazi na Cloud-u. Na kraju rada prikazani su i analizirani rezultati testiranja izabranog skupa funkcionalnosti implementiranog rešenja.*

Ključne reči: *AMI sistem, Cloud*

Abstract – *This paper describes the AMI architecture and presents the problem of handling a large amount of data. After that, the software solution of the AMI system is presented, where a part of the applications is on the Cloud. Also, the paper presents and analyzes the results of testing the selected set of functionalities of the implemented solution.*

Keywords: *AMI system, Cloud*

1. UVOD

Osnovni zadatak tradicionalnih elektroenergetskih sistema bio je da se potrošačima obezbedi isporuka električne energije, a da to čine uz što niže troškove i uz odgovarajući kvalitet. Porast broja stanovnika, njihova potreba za električnom energijom, kao i promene u načinu na koji se električna energija proizvodi, prenosi, distribuira i troši, dovele su do nastanka pametnih elektroenergetskih mreža (*Smart Grid*). *Smart Grid* uvodi dvosmernu komunikaciju između preduzeća i potrošača i donosi promenu u načinu na koji se pristupa zahtevima potrošnje, sigurnosti i unapređenjima zaštite životne sredine.

U okviru *Smart Grid*-a, nalazi se i AMI (*Advanced Metering Infrastructure* - napredna merna infrastruktura). AMI sistem čine sistemi zaduženi za merenje, prikupljanje i analizu potrošnje električne energije, kao i sistemi zaduženi za komunikaciju sa pametnim mernim uređajima (*smart meters*).

Osnovni zadatak jednog AMI sistema jeste rad sa velikom količinom podataka, gde se podaci o potrošnji električne energije prikupljaju periodično. Često se u kontekstu AMI sistema spominje i *Cloud*. *Cloud* podrazumeva pružanje neograničenih usluga i deljenje istih putem interneta, gde korisnici plaćaju iznajmljivanje usluge kako bi koristili date resurse.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Darko Čapko, vanr. prof.

Cloud čine infrastruktura (računarski resursi), centri podataka i virtuelizacija, koja omogućava raspodelu fizičkog resursa na više virtualnih resursa koji se mogu nezavisno i paralelno koristiti prema potrebama korisnika. Upotreba *Cloud* usluga u AMI sistemu doprinosi visokoj pouzdanosti i dostupnosti podataka. Imajući u vidu da količina podataka u AMI sistemu varira u zavisnosti od doba dana, *Cloud* platforme koriste se upravo u cilju dobijanja skalabilnih i fleksibilnih resursa.

2. ARHITEKTURA AMI SISTEMA

AMI predstavlja integrisanu infrastrukturu koja podrazumeva dvosmernu komunikaciju između pametnih brojila i definisanih učesnika na tržištu. Pametna brojila šalju podatke o potrošnji električne energije, dok preduzeća imaju mogućnost upravljanja. AMI uključuje komunikacionu mrežu i pruža mogućnost automatskog čitanja potrošnje električne energije, bržeg i preciznijeg identifikovanja i izolovanja napada, brže restauracije i praćenja napona **Error! Reference source not found.**.

2.1. Infrastruktura AMI sistema

AMI se tipično sastoji od 3 komponente: pametna brojila, koja prikupljaju podatke u određenim vremenskim intervalima, komunikaciona mreža, koja prenosi podatke od pametnih brojila do MDMS-a (Meter Data Management System) i MDMS, koji skladišti i obrađuje podatke i povezuje ih sa ostalim sistemima 0. AMI obuhvata i sistem upravljanja podacima kojim se vrši razmena podataka između svih učesnika u sistemu.

Pametna brojila omogućavaju merenje potrošnje transformatorskih stanica, domaćinstava, komercijalnih i industrijskih potrošača. **HAN** (*Home Area Network*) je bežična mreža u okviru manjeg prostora (kuća, kancelarija), gde je omogućeno povezivanje sa pametnim brojilima putem kućnih aparata. Pametna brojila koriste PLC (*Power Line Communication*) i RF (*Radio Frequency*) sa koncentratorima/ruterima u okviru NAN (*Neighborhood Area Network*) mreže. **NAN** mreža je ograničena na određeno geografsko područje, obično na nivou transformatorske stanice. Koncentratori/ruteri su preko **WAN** (*Wide Area Network*) mreže povezani sa AMI aplikacionim centrima, kao što su NMS, MDMS i centrom podataka.

AMM (*Automated Meter Management*) podrazumeva daljinsko očitavanje, nadzor i upravljanje drugim komponentama AMI sistema, veliku brzinu obrade podataka, vezu sa drugim sistemima i transfer podataka u MDMS. Komunikacija između AMM sistema i ostalih

sistema vrši se putem WAN. **Koncentrator podataka** predstavlja uređaj koji je povezan sa pametnim brojilima i MDMS. Koncentrator mora da obezbedi neposrednu komunikaciju sa pojedinačnim brojilom. On automatski ili na zahtev izvršava funkcije očitavanja i memorisanja podataka sa brojila i predaje istih podataka na zahtev MDMS sistemu. Podaci sa pametnih brojila se prikupljaju putem NAN mreže, a dostavljaju MDMS-u putem WAN mreže. Komunikacija sa brojilima se odvija putem PLC komunikacije, a sa MDMS preko GPRS komunikacije [1]. Koncentrator ima i funkciju upravljanja/parametrizacije kojom se menjaju parametri brojila, a može se izmeniti i softver brojila. Obrada podataka na koncentratoru podrazumeva obradu alarma i sastavljanje izveštaja o kvalitetu isporuke električne energije koji se po zahtevu šalje MDMS sistemu. **MDMS** je deo AMI sistema. Uključuje procese prikupljanja podataka od koncentratora, obrađivanja i čuvanja istih. Obrada podataka podrazumeva detaljnu analizu gde se isti tipovi podataka grupišu, kako bi se olakšao postupak dobavljanja željenih podataka. Obrada podrazumeva i obračun utrošene električne energije, kako bi se obezbedili podaci sistemu za obračun i naplatu električne energije, kao i čuvanje obrađenih podataka.

2.2. Pametna brojila

Pametna brojila mogu da mere i snimaju stvarnu potrošnju energije tokom dana u određenom vremenskom intervalu. Ona šalju prikupljene podatke do MDMS-a. Pametna brojila omogućavaju bržu detekciju otkaza, reagovanje i restauraciju, kao i bolju informisanost klijenata o statusu elektroenergetske mreže.

Funkcionalnosti pametnih brojila su [2]:

- Dinamičko određivanje cena podrazumeva način na koji se zahteva ravnomerna maksimalna tražnja. U toku trajanja maksimalne potražnje, potrošači se podstiču da smanje potrošnju energije, pošto je cena tada najveća. Potrošači mogu smanjiti račun za utrošenu električnu energiju time što će smanjiti njeno korišćenje tokom perioda maksimalne potražnje.
- Dvosmerna komunikacija, koja podrazumeva slanje podataka od pametnih brojila do MDMS-a kada se zatraži i upravljanje pametnim brojilima od strane preduzeća.
- Daljinsko upravljanje od strane preduzeća za nekoliko funkcija, kao što su daljinsko isključivanje/povezivanje potrošača, detekcija prekida napajanja, slanje upozorenja potrošaču i slanje povratnih signala potrošaču.
- Detekcija kvara, kao i mogućnost obaveštavanja preduzeća kada je napajanje obnovljeno.
- Mogućnost nadogradnje softvera. Ovo omogućava mogućnost nadogradnje pametnog brojila, dodavanje novih funkcija i ispravljanje grešaka u ranijoj verziji softvera.

2.3. Karakteristike potrošača

Karakteristike potrošača mogu da se koriste za opisivanje krivih dnevnih opterećenja. Uočavaju se sličnosti između krive opterećenja korisnika istog tipa. Na primer, maksimalno opterećenje stambenih potrošača uvek se koncentriše u jutarnjim i večernjim časovima, a maksimalno opterećenje industrijskih potrošača u toku dana 0. Razlike u potražnji za električnom energijom najčešće su primetne na sezonskoj osnovi, tokom sedmice i tokom dana. Na samu potražnju može uticati više faktora, među kojima i događaji poput popularnih emisija na televizijskim programima. Tipično, potražnja za električnom energijom je veća zimi nego leti, a takođe i niža u noćnim satima.

Što se tiče vremenskih intervala prikupljanja podataka, za domaćinstva je tipičan interval 1 sat, što je dovoljno za potrebe naplate. Za komercijalne i industrijske potrošače, preduzeća uglavnom koriste interval od 15 minuta za čitanje podataka sa pametnih brojila [2]. Preduzeća koriste podatke o potrošnji električne energije, kako bi dobili informacije o potrošačima koji su relevantni za optimizaciju njihovih programa energetske efikasnosti.

3. ANALIZA RADOVA AMI ARHITEKTURE

U radu [3], istraživan je dizajn optimalnih komunikacionih infrastruktura za AMI sistem. Tradicionalna AMI komunikaciona arhitektura podrazumeva MDMS koji je centralizovan i okružen glavnim operacijama i upravljačkim funkcijama. Potom je predložena arhitektura sa distribuiranim MDMS-om, gde se podaci šalju od pametnih brojila do koncentratora, koji dalje prosleđuju svom MDMS-u, a koji je povezan sa centralnim MDMS-om. U okviru istog rada predložena je i potpuno distribuirana AMI komunikaciona arhitektura, koja podrazumeva distribuiranje svih komponenti sistema.

U radu [7] analizira se upotreba *Cloud*-a u AMI sistemu, gde preduzeća razvijaju i održavaju servise na *Cloud*-u, kojima pristupaju pametna brojila i tako kontrolišu uređaje u vidu odgovora koji stiže sa *Cloud*-a. Ukoliko provajder servisa na *Cloud*-u ažurira ovu uslugu, pametno brojilo nije svesno ove promene, već će u porukama sa servisa dobijati drugačije odgovore.

4. OPIS I ARHITEKTURA IMPLEMENTIRANOG SISTEMA

Model podataka implementiranog rešenja je modelovan upotrebom CIM (*Common Information Model*) apstraktnog UML (*Unified Modelling Language*) modela kojim su predstavljeni elementi elektroenergetskog sistema, gde je za svakog **potrošača** vezano više analognih signala, tj. merenja koja se odnose na **aktivnu snagu** (**P**), **reaktivnu snagu** (**Q**) i **napon** (**V**), kao i **transformator** koji sadrži signal za merenje napona. Oni su grupisani na nivou **transformatorske stanice**, odnosno **geografskog područja**. Takođe, za svakog potrošača i transformator definiše se i **naponski nivo**, koji ima podatak o vrednosti nominalnog napona. Merenja imaju podatak i o adresi RTU uređaja za koji su vezani, kao i o jedinici merenja (P, Q i V). Potrošači i transformatori sadrže podatak o validnom opsegu napona, dok potrošači sadrže podatak i o tipu potrošača (domaćinstvo, tržni centar i firma) i maksimalnoj vrednosti aktivne i reaktivne snage.

Prethodno opisani objekti u sistemu predstavljaju statičke podatke sistema i nisu podložni izmeni. S obzirom na prirodu sistema, gde se merenja (P, Q i V) prikupljaju periodično, korišćena je posebna klasa koja od atributa ima referencu na potrošača, atribute koji predstavljaju

izmerene vrednosti P, Q i V, vreme kada su izmerene vrednosti, podatak da li je potrošač u alarmu i tipu alarma (previsok ili prenizak napon).

Za razvoj aplikacija u *Cloud* okruženju korišćena je *Service Fabric* platforma, koja omogućava razvoj distribuiranih aplikacija. *Service Fabric* je zasnovan na mikroservisima koji podrazumevaju softversku arhitekturu u kojoj se aplikacija sastoji od manjih, nezavisnih procesa, koji komuniciraju međusobno preko standardnih protokola. Mikroservisi se smeštaju na klasterima. **Klaster** (engl. *Cluster*) predstavlja skup čvorova, odnosno grupu fizičkih ili virtualnih mašina koji su povezani kako bi obezbedili pouzdano i dostupno okruženje za pokretanje aplikacija. Klasteri se mogu napraviti u **lokalnom okruženju**, gde je jedna fizička mašina logički podeljena na nekoliko čvorova, u okviru **Azure okruženja**, gde se može koristiti veći broj mašina, ili na nekom drugom Cloud servisu. **Čvor** (engl. *Node*) predstavlja fizičku ili virtualnu mašinu koja pripada određenom klasteru 0.

Implementirano softversko rešenje sadrži infrastrukturu za daljinsko očitavanje električne energije, daljinsko očitavanje statusa napajanja individualnog potrošača, kao i očitavanje vrednosti napona na mestu potrošača. Fokus rada bio je na arhitekturi koja omogućava akviziciju i obradu podataka. Aplikacije koje rade u lokalnom okruženju su: Klijent, SCADA (*Supervisory Control And Data Acquisition*) i simulator. Funkcionalnosti datih aplikacija, kao i ostalih komponenti sistema koje se nalaze na *Cloud*-u slede u nastavku.

Korisnik je u mogućnosti da preko **klijentske aplikacije** šalje statičke podatke sistema u vidu CIM/XML dokumenta. Ukoliko je validacija poslatih podataka prošla uspešno, korisnik u toku rada aplikacije ima uvid u trenutne vrednosti podataka koji se menjaju, uključujući potrošnju električne energije i vrednosti napona. Dati podaci odgovaraju realnim situacijama. Korisnik ima mogućnost pregleda krive potrošnje i napona po određenim kriterijumima koji se odnose na odabir tipa sezone (leto i zima), tipa dana (radni dan i vikend), tipa potrošača (domaćinstvo, tržni centar i firma) i specifičnog dana. U slučaju alarmnih situacija, korisnik će dobiti obaveštenje o vremenu generisanja alarma, nazivu potrošača i tipu alarma (previsok ili prenizak napon).

Implementirano rešenje pruža podršku za rad sa više simulatora koji u ovom slučaju predstavljaju RTU uređaje. Za komunikaciju između SCADA-e i simulatora korišćen je dnp3 protokol (*Distributed Network Protocol*). Merenja koja simulator šalje su **aktivna snaga**, **reaktivna snaga** i **napon**. Prilikom prijema merenja od simulatora, **SCADA** vrši proveru vrednosti napona i ukoliko je izvan validnog opsega označava alarm za dato merenje. Takođe, SCADA šalje sva merenja do *Calculation Engine*-a.

U procesu distribuirane transakcije omogućeno je samo dodavanje novih entiteta. Ideja **koordinatora transakcije** jeste da započne transakciju, proveri da li sva 3 servisa (NMS, SCADA i *Calculation Engine*) mogu da pređu iz jednog konzistentnog stanja u drugo i da na osnovu rezultata završi transakciju (*Commit* ili *Rollback*).

NMS opisuje statičke elemente sistema. NMS je svestan samo statičkih podataka sistema i ima zadatak da dostavlja podatke drugim servisima.

U okviru ***Calculation Engine***-a podržan je sledeći set funkcionalnosti: primanje izmerenih podataka od strane SCADA-e i slanje do klijentske aplikacije, komandovanje u slučaju detekcije alarma i agregacija mernih podataka u više nivoa. Primljeni podaci od SCADA-e su agregirani u 3 nivoa i upisuju se u minutnu, satnu i dnevnu tabelu. Agregirani podaci služe klijentu da u zavisnosti od tražene rezolucije dobije grafički prikaz potrošnje električne energije uz statističke podatke. Servis za agregaciju u unapred definisano vreme pokreće procese agregacije na pomenutim nivoima. Merenja u odnosu na koje se vrši agregacija se dobijaju od SCADA-e na svakih 30 sekundi.

5. REZULTATI IMPLEMENTIRANOG SISTEMA

Testiranje je rađeno u lokalnom i Azure okruženju. Testiranje je vršeno redom sa 1.000, 10.000, 100.000 i 1.000.000 objekata. U 1.000 objekata nalazi se 250 potrošača i 750 merenja (po 250 merenja aktivne snage, reaktivne snage i napona), i tako proporcionalno za veći broj objekata. Svako pojedinačno testiranje je vršeno sa 20 ponavljanja, pa dobijeni rezultati predstavljaju prosečnu vrednost istih.

Trajanje distribuirane transakcije obuhvata konverziju podataka u odgovarajući format i dodavanje istih u bazu podataka svake od komponenti. Rezultati testiranja prikazani su u tabeli 1.

Tabela 1: Trajanje distribuirane transakcije

Broj objekata	Lokalno okruženje	*Azure* okruženje
1,000	27.800 s	8.384 s
10,000	4 min 27 s	1 min 34 s
100,000	45 min 11 s	15 min 32 s
1,000,000	7 h 40 min	2 h 33 min

Rezultati testiranja detekcije alarma na *Calculation Engine*-u i slanje komande ka SCADA-i prikazani su u tabeli 2. Svako slanje merenja je sadržalo 10% potrošača koji su u alarmu. Za 1.000 objekata, sledi da je 25 potrošača u alarmu.

Tabela 2: Detekcija alarma i komandovanje

Broj objekata	Lokalno okruženje	*Azure* okruženje
1,000	0.302 s	1.314 s
10,000	0.922 s	3.888 s
100,000	9.848 s	31.532 s
1,000,000	1 min 41 s	5 min 19 s

Calculation Engine vrši upis novodobijenih merenja u bazu podataka, čiji rezultati su prikazani u tabeli 3.

Tabela 3: Trajanje upisa merenja u bazu podataka

Broj objekata	Lokalno okruženje	*Azure* okruženje
1,000	0.110 s	0.515 s
10,000	0.971 s	6.015 s
100,000	12.425 s	1 min 11 s
1,000,000	1 min 56 s	12 min 44 s

Testiranje je vršeno i za proces slanja merenja od strane SCADA-e, pa sve do prikaza izmenjenih vrednosti u vidu tabele na klijentskoj aplikaciji, što je prikazano u tabeli 4. Dobijena vremena uključuju i vremena dobijena testiranjem iz tabela 2 i 3. Takođe, vremena koja su dobijena ovim testiranjem uključuju i sumiranje merenja na nivou pripadnosti transformatorskoj stanici i geografskom području, kao i upisivanje rešenih alarma u bazu podataka od strane *Calculation Engine*-a.

Tabela 4: *Trajanje slanja merenja*

Broj objekata	Lokalno okruženje	*Azure* okruženje
1,000	0.584 s	3.319 s
10,000	2.851 s	13.057 s
100,000	28.789 s	1 min 50 s
1,000,000	4 min 47 s	17 min 57 s

Vremensko trajanje distribuirane transakcije pokazalo je da se povećavanjem broja redova koji se upisuje u bazu podataka svake od komponenti koja učestvuje u distribuiranoj transakciji povećava približno linearno. Sam proces distribuirane transakcije u okviru Azure okruženja brži za oko 3 puta u odnosu na isti proces u lokalnom okruženju.

Proces detekcije alarma na *CalculationEngine*-u i slanje komande na SCADA-i je za oko 4 puta brže u lokalnom okruženju, u odnosu na Azure okruženje.

Perzistencija pristiglih merenja na *CalculationEngine*-u na oba testirana okruženja ima približno linearni rast sa povećavanjem broja merenja. Samo upisivanje merenja u bazu podataka je za oko 6 puta brže u lokalnom okruženju. Imajući u vidu da sam upis u Azure okruženju vrši jedna mašina, koja ima slabije performanse od lokalnog računara, kao i činjenica da se baza podataka u lokalnom okruženju nalazi na istom računaru kao i sama aplikacija, pa iz navedenih razloga sledi zaključak zašto je ovaj proces brži u lokalnom okruženju.

Rezultati procesa slanja merenja od SCADA-e do klijentske aplikacije su pokazali da je sa manjim brojem merenja navedeni proces brži u lokalnom okruženju za oko 5 puta u odnosu na *Azure* okruženje, dok je sa većim brojem merenja proces u lokalnom okruženju brži za oko 3.5 puta. Treba imati u vidu da iako klijentska aplikacija nije deo *Cloud* okruženja, ona se u lokalnom okruženju nalazi na istom računaru kao i aplikacije koje su deo *Cloud*-a.

Rezultati koji su dobijeni pokazuju da na brzinu testiranih funkcionalnosti bitno utiče i karakteristika mašine. Na primeru distribuirane transakcije pokazano je da bitnu ulogu igra broj mašina u okviru klastera, odnosno raspodela procesa upisivanja u bazu podataka na svakoj od mašina. Zbog ograničenja u vidu performansi računara, testiranje je obuhvatilo slanje redova do veličine 1.000.000. Očekivano je da u realnim sistemima broj merenja bude i veći.

6. ZAKLJUČAK

U ovom radu predstavljen je AMI sistem uz poseban osvrt na komunikacionu infrastrukturu, pametna brojila i tipove potrošača. Analizirani su radovi iz oblasti arhitekture AMI sistema i korišćenja *Cloud* usluga u takvim sistemima, koji su ukazali na benefite i probleme svakog od predloženih rešenja. Implementirano rešenje je obuhvatilo izabrani set funkcionalnosti, uz poseban osvrt na alarme, slanje podataka prema tipu potrošača i agregaciju podataka u istorijsku bazu podataka.

Vršeno je poređenje vremenskih rezultata pojedinačnih funkcionalnosti, kao i rezultata dobijenih testiranjem u okviru izabrana dva *Cloud* okruženja, odnosno lokalnog i *Azure* okruženja.

Razmatrani pravci daljeg razvoja su testiranje implementiranog sistema u okviru *Cloud* okruženja gde bi mašine u okviru klastera imale bolje karakteristike. Uz to, moguć je i odabir drugog *Cloud* provajdera, pa bi se dobijeni rezultati mogli porediti između svih izabranih *Cloud* okruženja.

7. LITERATURA

[1] JP Elektroprivreda Srbije, "*Funkcionalni zahtevi i tehničke specifikacije AMI/MDM sistema*", 2014.

[2] G. R. Barai, S. Krishnan, and B. Venkatesh, "*Smart metering and functionalities of smart meters in smart grid - A review*", Electrical Power and Energy Conference (EPEC), 2015.

[3] J. Feng, "*An analytics of electricity consumption characteristics based on principal component analysis*", IOP Conference Series: Earth and Environmental Science, pp. 0–6, 2018.

[4] U.S. Department of Energy, "*Advanced Metering Infrastructure and Customer Systems: Results from the Smart Grid Investment Grant Program*", Off. Electr. Deliv. Energy Reliab., pp. 1–98, 2016.

[5] H. Bai, "*Programming Microsoft Azure Service Fabric*", Microsoft Press, 2016.

[6] R. Q. Hu, J. Zhou, R. Q. Hu, and S. Member, "*Scalable Distributed Communication Architectures to Support Advanced Metering Infrastructure in Smart Grid*", IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 9, pp. 1632–1642, 2012.

[7] M. Yigit, V. C. Gungor, and S. Baktir, "*Cloud Computing for Smart Grid applications*", Comput. Networks, vol. 70, pp. 312–329, 2014.

Kratka biografija:

Bratislav Batinić je rođen 18.10.1994. godine u Bačkoj Topoli. Završio je srednju ekonomsku školu "Dositej Obradović" u Bačkoj Topoli 2013. godine. Diplomirao je na Fakultetu tehničih nauka u Novom Sadu 2017. godine. Master studije na istom fakultetu je upisao školske 2017/2018. godine. Ispunio je sve obaveze i položio je sve ispite predviđene studijskim programom.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/01BE40Sedlar

AUTOMATSKA VERIFIKACIJA REPLIKACIJE U OKVIRU SCADA SISTEMA

AUTOMATIC REPLICATION VERIFICATION WITHIN A DISTRIBUTED SCADA SYSTEM

Milica Sedlar, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu je opisan postupak automatizacije testova za verifikaciju konfiguracije u okviru DMS-a (Distributed Management System). Takođe je opisan i proces testiranja verifikacije konfiguracije, kao i alati i tehnologije koje su korišćene za izradu praktičnog dela zadatka. Automatizacija testova za verifikaciju konfiguracije se sastoji u manipulaciji lokalnim i daljinskim prekidačima u produkcionom sajtu i proveri replikacije u drugim sajtovima u sistemu pomoću aplikacije.*

Ključne reči: *DMS, replikacija, verifikacija*

Abstract – *This paper describes the procedure for automation of tests to verify the configuration on DMS (Distributed Management System). It also describes the process of testing to verify the configuration, as well as the tools and technologies used to create a practical part of the task. Automation of tests to verify the configuration consists in manipulating local and remote switches in the production site and checking replication in other sites in the system using the application.*

Keywords: *DMS, replication, verification*

1. UVOD

Testiranje softvera je od izuzetne važnosti u procesu životnog ciklusa razvoja softvera. Kako bi se u što većoj meri smanjile greške koje nastaju prilikom razvoja softverskog proizvoda, proces testiranja softvera u velikoj meri utiče na smanjenje nastalih grešaka kao i na kvalitet samog proizvoda. Kako bi se proces testiranja ubrzao, a samim tim i smanjila ljudska greška prilikom testiranja, vrši se automatizacija test slučajeva.
Verifikacija softvera predstavlja jednu fazu testiranja softvera. U ovoj fazi testiranja se utvrđuje da li softver odgovara zahtevima korisnika.
Konfiguracija predstavlja instalaciju i podešavanje softvera kako bi isti bio spreman za dalje testiranje ili isporuku krajnjem korisniku.
U ovom radu će biti opisana automatizacija testova za verifikaciju konfiguracije distribuiranog sistema. Kako je distribuirani sistem kompleksan po svojoj strukturi jer se sastoji od više samostalnih računara međusobno povezanih računarskom mrežom, samim tim je i verifikacija konfiguracije kompleksna. Upravo se zbog kompleksnosti verifikacije konfiguracije distribuiranog sistema javila ideja da se ova vrsta testova automatizuje, odakle je i nastalo celokupno istraživanje kao i tema rada. Testovi za verifikaciju konfiguracije čija automatizacija će biti opisana u ovom radu jesu testovi za replikaciju.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Darko Čapko, vanr. prof.

2. TESTIRANJE SOFTVERA

Testiranje softvera [1][2] je proces analize elemenata softvera kako bi se utvrdile razlike između postojećeg stanja i zahteva korisnika, ali i kako bi se ustanovile karakteristike softvera.
To je jedan od najviše korišćenih metoda za upravljanje rizikom, a sve u cilju verifikacije ispunjenja funkcionalnih zahteva.
Testiranje softvera predstavlja petu fazu u procesu razvoja softvera po metodologiji „Vodopad“ čija aktivnost je integrisanje svih komponenti, verifikacija, validacija, instalacija i obuke, a čiji je rezultat dobijanje funkcionalnog sistema.

3. KORIŠĆENE TEHNOLOGIJE I ALATI

U toku procesa istraživanja i implementiranja aplikacije za automatsko testiranje verifikacije konfiguracije sistema korišćeno je više različitih alata i tehnologija koje će biti opisane u nastavku.

3.1. Microsoft Visual Studio

Microsoft Visual Studio [3] je integrisano razvojno okruženje koje se koristi za razvoj računarskih programa, kao i web stranica, web aplikacija, web servisa i mobilnih aplikacija.
Visual Studio koristi razvojne platforme kao što su Windows API, Windows Forms, Windows Presentation Fundation kao i mnoge druge platforma. Integrisani debager funkcioniše jednako dobro kako na nivou debagovanja izvornog koda tako i na mašinskom nivou debagovanja.
Microsoft Visual Studio podržava trideset šest različitih programskih jezika. Ugrađeni programski jezici su: C, C++, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML i CSS. Podrška za ostale programske jezike, kao što su Python, Ruby, Node.js kao i mnogi drugi, je potrebno instalirati kao posebne dodatke.

3.2. Extensible Markup Lenguage (XML)

XML [4] predstavlja markup jezik koji definiše skup pravila za enkodiranje dokumenata u format koji je čitljiv i za čoveka i za mašinu. Definisan je od strane World Wide Web Consortium-a (W3C) kroz XML 1.0 specifikaciju i predstavlja besplatan i otvoren standard. Predstavlja tekstualni format podataka sa podrškom Unicode karaktera za veliki broj govornih jezika.
Kao dodatak tome što je XML dobro formatiran, može se proveriti validnost XML dokumenta. To znači da je moguće izvršiti proveru da li su svi elementi u dokumentu opisani u skladu sa sintaksnim pravilima određenim XML šemom. XML procesori koji služe za obradu podataka mogu voditi računa o validnosti samog dokumenta, ali ukoliko vode, moraju imati mehanizam za prijavu grešaka ukoliko je dokument nevalidan.

3.3. Windows Power Shell

Windows PowerShell je [5] skript jezik za pisanje željenih komandi koje je potrebno izvršiti. Dizajniran je posebno za potrebe sistemske administracije radi što brže automatske administracije operativnih sistema kao što su Linux, macOS, Unix i Windows.
Windows PowerShell podržava command-line i skript okruženja eliminišući kopleksne probleme i dodajući nove karakteristike. Baziran je na objektima, a ne na tekstu, tako da se na izlazu i dobijaju objekti, a ne tekst. Izlazni objekat jedne komande je moguće poslati kao ulazni objekat drugoj komandi.
Windows PowerShell olakšava proces tranzicije od pisanja komandi interaktivno do kreiranja i pokretanja skripti.

3.4. C# Programski jezik

Programski jezik C# je dizajniran da bude upotpunosti kompatibilan sa .NET kontrolisanim programskim okruženjem. Dizajniran je da bude platformski nezavisan. Kako je to objektno-orijentisani programski jezik, spada u programske jezike višeg reda i predstavlja deo .NET kontrolisanog programskog okruženja.
C# programski jezik ne podržava višestruko nasleđivanje, pa se kao rešenje ovog problema koriste interfejsi. Takođe, ovaj programski jezik ima još jednu korisnu karakteristiku, a to je „garbage collector“. To znači da nije potrebno praviti destruktor za svaku klasu. Moguće je i direktno pristupiti memoriji upotrebom pokazivača, ali pokazivače neće „počistiti“ garbage collector sve dok se to posebno ne navede.

3.5. Active Directory Services Interface Edit (Adsi Edit)

Active Directory Services [6] je repozitorijum mreže i aplikacija koje koriste više korisnika ili više nekih drugih aplikacija. Alat za pristupanje servisu aktivnog direktorijuma je Adsi Edit. Pomoću ADSI Edit alata je moguće na jednostavan način manipulisati objektima koji se nalaze na servisu direktorijuma. Kao što je moguća jednostavna manipulacija objektima, na isti način je moguća i jednostavna manipulacija atributima tih objekata.
ADSI Editi dolazi u sklopu instalacije Windows Servera.
Za pristupanje objektima servis direktorijuma se koristi LDAP (Lightweight Directory Access Protocol) standard koji definiše kako je taj servis direktorijum zaista implementiran i dostupan.

3.6. Domain Name System (DNS)

DNS [6] predstavlja najveću digitalnu bazu podataka na svetu. Web pretraživači komuniciraju putem internet protokol (IP) adresa. DNS prevodi imena domena, imena mašina i imena servisa u IP adrese, takođe IP adrese prevodi u imena mašina, ali i imena mašina prevodi u alternativna imena (aliase). Svaki uređaj koji je povezan na internet ima jedinstvenu IP adresu koju druge mašine koriste kako bi pronašle uređaj. Za mapiranje podataka koji govore DNS serveru sa kojom IP adresom se povezuje svaki domen, kao i kako se rukovodi sa zahtevima koji se šalju svakom domenu, koriste se DNS rekordi.

4. AUTOMATIZACIJA TESTOVA ZA VERIFIKACIJU KONFIGURACIJU REPLIKACIJE U DISTRIBUIRANOM SISTEMU

Distribuirani sistem je zbog njegove složenosti veoma teško konfigurisati i taj proces se najčešće izvodi sporo. Kako je konfiguracija ovakvog sistema ispraćena velikim brojem konfiguracionih koraka, te konfiguracione korake je potrebno i verifikovati. Verifikacija konfiguracije [7] se svodi na test slučajeve koje je potrebno izvršiti kako bi se utvrdilo da li je sistem dobro konfigurisan i da li je spreman za dalju upotrebu. Verifikacija konfiguracije distribuiranog sistema je od velikog značaja iz razloga što loše konfigurisan sistem može dovesti do otkaza sistema. S obzirom na to da je konfiguracija kompleksna, i verifikacija konfiguracije je takođe kompleksna i spora. Kako bi se ubrzao proces i otklonila mogućnost greške prilikom verifikacije konfiguracije ovakvih sistema, automatizacija testova ove vrste se našla kao moguće rešenje.
Osim povećanja tačnosti, automatizacija ove vrste testova doprinosi i smanjenju vremena koje je potrebno utrošiti da bi se konfiguracija verifikovala.
Verifikacija konfiguracije replikacije među sajtovima je od izuzetne važnosti za upravljanje distribuiranim sistemima kao što je DMS jer i najmanja konfiguraciona greška može da ima katastrofalne posledice. Zbog posledica koje konfiguracione greške mogu da uzrokuju, javila se potreba da se istraži mogućnost detektovanja grešaka na vreme, sa velikom pouzdanošću i u što kraćem vremenskom periodu, još u fazi verifikacije. Upravo je ta potreba stvorila ideju za razvoj aplikacije koja ubrzava i povećava pouzdanost verifikacije.

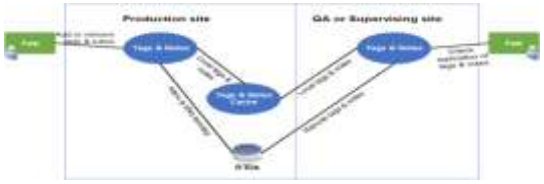
4.1. Opis automatizacije testova za verifikaciju konfiguraciju replikacije

Kako verifikacija konfiguracije replikacije u DMS sistemu traje dugo, a pritom je mogućnost ljudske greške neizbežna, razvijena je aplikacija kako bi se ovaj proces ubrzao i kako bi se greške svele na minimum.
Automatizovani su verifikacioni testovi za proveru konfiguracije replikacije u distribuiranom sistemu koji

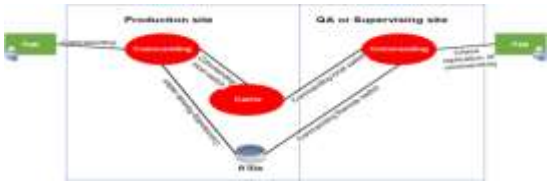
ima tri sajta. Test slučajevi verifikuju replikaciju u DMS sistemu tako što se vrši manipulacija prekidačima (komandovanje (otvaranje i zatvaranje) lokalnog i daljinskog prekidača kao i dodavanje i brisanje oznaka i beleški na prekidačima). Komandovanje i dodavanje oznaka i beleški se vrši u produkcionom sajtu, dok se provera replikacije vrši u sajtovima za kontrolu kvaliteta i nadgledanje.

Replikacija između sajtova u distribuiranom sistemu kao što je DMS je od velike važnosti zbog praćenja eventualnih otkaza na distributivnoj mreži, kao i planiranja i testiranja preključivanja na istoj. Ukoliko nema replikacije između sajtova, može doći do velikih problema koje mogu da izazovu katastrofalne posledice čak i po život ljudi.

Svaki sajt ima računarski resurs na kome se nalazi servis koji je neophodno da bude startovan kako bi replikacija između sajtova bila moguća. Informacije o manipulaciji lokalnim prekidačima se čuvaju u keš memoriji koja je zajednička za sve sajtove u sistemu, dok se informacije o manipulaciji daljinskim prekidačima čuvaju u RealTime bazi podataka. Slika 4.1. prikazuje replikaciju oznaka i beleški, dok slika 4.2. prikazuje repikaciju komandovanja prekidačima.



Slika 4.1. *Prikaz replikacije oznaka i beleški*

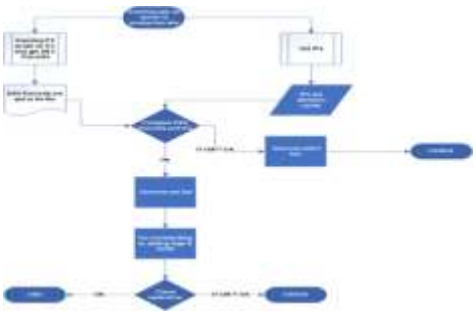


Slika 4.2. *Prikaz replikacije komandovanja prekidačima*

4.2. Opis rešenja automatizacije testova za verifikaciju konfiguracije replikacije

U toku procesa automatizacije testova za verifikaciju replikacije obuhvaćena su četiri testa, a to su komandovanje (otvaranje i zatvaranje) lokalnim prekidačima, komandovanje (otvaranje i zatvaranje) daljinskim prekidačima, dodavanje i brisanje oznaka na lokalnim i daljinskim prekidačima i dodavanje i brisanje beleški na lokalnim i daljinskim prekidačima. Ovim testovima se verifikuje replikacija između sajtova u DMS sistemu, a tok rada aplikacije za verifikaciju je prikazan na slici 4.3.

Aplikacija se pokreće u produkcionom sajtu na jednom računarskom resursu. Kada se aplikacija pokrene, vrši se provera da li je određen servis, koji je potreban za replikaciju između sajtova, startovan. Ta provera se vrši tako što se izvrši Power Shell skripta na DC-u sistema koja pročita DNS rekorde servisa i upiše u tekstualni fajl, nakon čega se ti DNS rekordi iz tekstualnog fajl uporede sa IP adresama na računarskom resursu.



Slika 4.3. *Prikaz toka rada aplikacije za verifikaciju replikacije*

Nakon što se proveri da li je servis startovan ili ne, vrši se komandovanje slučajno odabranim lokalnim i daljinskim prekidačima, kao i dodavanje oznaka i beleški na iste. Kada se završi manipulacija slučajno odabranim prekidačima, rezultat se upisuje u xml fajl.

Taj xml fajl se distribuira na računarske resurse na kojima se takođe nalazi određen servis koji je potreban za replikaciju, a koji se nalaze u drugim sajtovima. Računarski resursi na kojima se nalazi taj servis se čitaju iz ADSI Edita, a aplikacija uspe da pristupi ADSI Editu i pročita te informacije na osnovu LDAP putanje.

Replikacija se verifikuje u sajtu za nadgledanje i sajtu za kontrolu kvaliteta. Verifikacija konfiguracije replikacije se vrši tako što se čitaju podaci iz xml fajla, koji su dobijeni iz produkcionog sajta, i upoređuju sa podacima iz keš memorije, ako se manipuliše lokalnim prekidačima. Ukoliko se manipuliše daljinskim prekidačima, verifikacija konfiguracije se vrši poređenjem podataka iz xml fajla sa podacima iz RealTime baze.

Nakon što se replikacija verifikuje, potrebno je vratiti sistem u početno stanje. Vraćanje sistema u početno stanje se vrši u produkcionom sajtu tako što se predhodno dodate oznake i beleške obrišu, a lokalni i daljinski prekidači se otvore, odnosno zatvore u zavisnosti od predhodne akcije

5. REZULTATI TESTIRANJA

Testiranje aplikacije je vršeno u testnom okruženju koje se sastoji od tri sajta (produkcioni, sajt za nadgledanje i sajt za kontrolu kvaliteta) sa po jednim računarskim resursom sa određenim servisom u svakom sajtu.

U toku testiranja poređeno je vreme trajanja ručno rađene verifikacije konfiguracije replikacije sa vremenom koje je potrebno da se verifikacija uradi pokretanjem aplikacije.

Tabela 1. prikazuje vreme trajanja verifikacije konfiguracije replikacije rađene ručno, dok tabela 2. prikazuje vreme trajanja verifikacije konfiguracije replikacije rađen automatski, puštanjem aplikacije.

Prvi korak koji se vrši kada se verifikuje konfiguracija replikacije jest provera da li je startovan servis. Ručna provera ovog koraka zahteva da se određeni servis pronađe, detektuje i utvrdi njegovo stanje, dok se automatski (puštanjem aplikacije) stanje servisa utvrđuje iz programskog koda što skraćuje izvršavanje ovog koraka za 120s.

Tabela 1. *Vreme trajanja verifikacije konfiguracije replikacije rađene ručno*

Aktivnost za izvršavanje	Vreme trajanja izvršavanja aktivnosti [s]
Provera da li je servis startovan	300 s
Komandovanje lokalnim prekidačima	30 s
Komandovanje daljinskim prekidačima	30 s
Dodavanje oznaka	40 s
Dodavanje beleški	40 s
Provera replikacije u sajtu za nadzor	180 s
Provera replikacije u sajtu za kontrolu kvaliteta	180 s
Vraćanje lokalnog prekidača na početno stanje	30 s
Vraćanje daljinskog prekidača na početno stanje	30 s
Brisanje oznaka	30 s
Brisanje beleški	30 s
Ukupno vreme trajanja verifikacije	920 s

Tabela 2. *Vreme trajanja verifikacije konfiguracije replikacije rađene automatski, puštanjem aplikacije*

Aktivnost za izvršavanje	Vreme trajanja izvršavanja aktivnosti [s]
Provera da li je servis startovan	20 s
Komandovanje lokalnim prekidačima	15 s
Komandovanje daljinskim prekidačima	15 s
Dodavanje oznaka	15 s
Dodavanje beleški	15 s
Provera replikacije u sajtu za nadzor	15 s
Provera replikacije u sajtu za kontrolu kvaliteta	15 s
Vraćanje lokalnog prekidača na početno stanje	15 s
Vraćanje daljinskog prekidača na početno stanje	15 s
Brisanje oznaka	15 s
Brisanje beleški	15 s
Ukupno vreme trajanja verifikacije	170 s

Komandovanje lokalnim ili daljinskim prekidačima zahteva da se na šemi nađe željeni lokalni ili daljinski prekidač, a zatim je potrebno ručno promeniti vrednost prekidača. Automatsko komandovanje lokalnim ili daljinskim prekidačem se vrši tako što se iz keš memorije nasumično odabere lokalni ili daljinski prekidač i promeni se njegova vrednost, bez potrebe traženja prekidača na šemi. Ručno komandovanje lokalnim i daljinskim prekidačima ukupno traje 60s, dok ja ze ovaj korak automatski potrebno polovina vremena (30s).

Za dodavanje oznaka i beleški je automatski potrebno ukupno 30s. Ručno izvršavanje zahteva odabir željenog prekidača na šemi, zatim je potrebno zadati komandu da se doda oznaka ili beleška i na kraju je dodati. Dodavanje ručno samo oznake ili samo beleške traje 40s, što je za 10s više nego što je potrebno za izvršavanje oba ova koraka automatski. Ovde je vidno značajno smanjenje vremena trajanja verifikacije.

Replikacija se provarava tako što se ode u sajt za nadzor ili u sajt za kontrolu kvaliteta i na šemi se pronađe lokalni ili daljinski prekidač kome je promenjena vrednost ili na koji je dodata oznaka ili beleška. Provera replikacije ručno po sajtu traje 180s. Automatski se replikacija proveri za 15s po sajtu jer se provera radi pristupanjem keš memorije, bez potrebe pokretanja šeme i traženja željenih prekidača. Provera konfiguracije replikacije automatski je znatno smanjena, čak za 165s po sajtu.
Automatizacijom vraćanja lokalnog i daljinskog prekidača na početno stanje, kao i brisanja oznaka i beleški je smanjeno vreme izvršavanja sa ukupno 120 s na ukupno 60 s, što je znatno uticalo na uštedu vremena koje je bilo potrebna za izvršavanje ovih koraka.

Ukupno vreme trajanja verifikacije konfiguracije replikacije je smanjeno sa 920s na 170s.

U tabelama se jasno vidi da je automatizacijom ova četiri testa za verifikaciju konfiguracije replikacije ušteda vremena velika.

6. ZAKLJUČAK

Cilj rada je bila automatizacija testova za verifikaciju konfiguracije replikacije u distribuiranom sistemu.

U prvom delu je bilo potrebno realizovati automatsko čitanje DNS rekorda sa DC-a, automatsko čitanje IP adresa sa računarskog resursa gde je potrebno izvršiti verifikaciju, kao i upoređivanje DNS rekorda i IP adresa radi utvrđivanja stanja servisa. Automatsko čitanje DNS rekorda je urađeno u skript jeziku Power Shell, dok je automatsko čitanje IP adresa i upoređivanje sa DNS rekordima urađeno u programskom jeziku C#.

U drugom delu je bilo potrebno automatizovati čitanje svih računarskih resursa sa određenim servisom sa svih sajtova iz Adsi Edita, dok je u trećem delu bilo potrebno realizovati manipulisanje lokalnim i daljinskim prekidačima i proveru replikacije. Drugi i treći deo su realizovani upotrebom programskog jezika C#.

Automatizacijom testova za verifikaciju konfiguracije je ubrzan proces verifikacije konfiguracije u distribuiranom sistemu, ali je povećana i pouzdanost verifikacije.

7. LITERATURA

[1] Elfriede Dustin, Jaff Rashka, Johan Pau, Automated Software Testing, Introduction, management and performance, New York, 2008.
[2] Myers:, Glendford J., The art of software testing., New Jersy, 2004.
[3] De, Alan, Visual SourceSafe: Microsoft's Source Destruction System. Highprogrammer.com, 2009.
[4] Pilgrim, Mark, The history of draconian error handling in XML, 2004.
[5] https://docs.microsoft.com/en-us/powershell/scripting/powershell-scripting?view=powershell-6 , datum pristupa 26.05.2018.
[6] Brian Desmond, Joe Richards, Robbie Allen, Alistair G. Lowe-Norris, Active Directory: Designing, Deploying, and Running Active Directory, 2010.
[7] David R. Bourgeois, James A. Ryan, Subhash C. Varshney, Data processing system with self testing and configuration mapping capability, USA, 1982.

Kratka biografija:

Milica Sedlar je rođena 1992. godine u Novom Sadu. Srednju školu je završila u Inđiji 2011. godine. Fakultet Tehničkih Nauka u Novom Sadu je upisala 2011. godine. Ispunila je sve obaveze i položila sve ispite predviđene studijskim programom na smeru Računarstvo i automatika, usmerenju Računarske nauke i informatika. Odmah nakon toga je upisala master akademske studije na smeru Primenjeno softversko inženjerstvo i ispunila sve svoje obaveze i položila sve ispite predviđene studijskim smerom.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.316
DOI: https://doi.org/10.24867/01BE43Petkovic

DINAMIČKA RASPODJELA MIKROSERVISA U DISTRIBUIRANOM SISTEMU

DYNAMIC DISTRIBUTION OF MICROSERVICE IN A DISTRIBUTED SYSTEM

Nebojša Petković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu predstavljena je mogućnost dinamičkog kreiranja mikroservisa u zavisnosti od potreba sistema, tj. od količine podataka koja je neophodana da se obradi. Ideja ovog rada je da se dinamički kreirani servisi opterete sličnom ili približno sličnom količinom posla, kako bi došlo do maksimalne iskorišćenosti resursa u okviru mikroservisne arhitekture, a samim tim i povećanja performansi. U okviru rada dat je detaljan opis mikroservisne arhitekture i navedene su prednosti i nedostaci mikroservisne arhitekturu u odnosu na aplikacije kreirane pomoću monolitnog pristupa. Nakon teorijskog uvoda, dat je opis predloženog rješenja. Na kraju rada predstavljeni su rezultati poređenja izvršavanja testne aplikacije kada je ona kreirana kao monolitna u odnosu na vrijeme izvršavanja kada je ona kreirana pomoću mikroservisne arhitekture.*

Ključne reči: *Cloud, Mikroservisna arhitektura, dinamičko kreiranje mikroeservisa*

Abstract – *The possibility of creating a dynamic microservice based on the system needs (the amount of data needing to be processed) will be shown in this paper. The idea of this assignment is for dynamically created systems to be loaded with similar or equal amount of work, so you would use the maximum amount of resources in the microservice arhitecture, and with that, amount to better performance. This paper contains detail description of the microservice arhitecture, as well as the pros and cons of the microservice arhitecture compared to applications created using a monolithic system. After the theoretical introduction, a description of a possible solution is given. The results of the comparasion between the execution of the test application when it's created as a monolithic compared to the time of execution when it's created using microservice arhitecture.*

Ključne reči: *Cloud, Microservice architecture, dynamic microservice design*

1. UVOD

Tradicionalni način razvijanja softvera podrazumijeva da kompanije posjeduju sopstvenu infrastrukturu. Porastom korinisčkog softvera neophodna je i investicija u proširenje hardvera što predstavlja veliku manu ovakvog pristupa. Razvojem računarskih mreža i konstantim rastom brzine Interneta stekli su se preduslovi za reogranizaciju računarskih sistema na ekonomičniji način. Stvorena je mogućnost za isporuku računarskih usluga putem Interneta.

Kompanije koje svoje usluge nude na ovaj način nazivaju se *Cloud* provajderi i obično svoje računarske usluge naplaćuju na osnovu potrošnje računarskih resursa, slično kao što elektrodistribucije naplaćuju potrošnju struje. Neki od najpoznatijih provajdera su: Amazon, Google, Microsoft, IBM, Alibaba, itd. *Microsoft Azure* je Microsoft-ova *Cloud* computing platforma, koja pruža širok spektar usluga koje se mogu koristiti bez kupovine i pružanja sopstvenog hardvera. *Azure*-ovi računarski, skladišni, mrežni i aplikacioni servisi omogućavaju korisniku da se fokusira na izgradnju odličnih rješenja bez potrebe da vodi računa o tome kako se fizička infrastruktura sastavlja [1].

Izraz "*Microservice Architecture*" se pojavio u poslednjih nekoliko godina kako bi opisao određeni način dizajniranja softverskih aplikacija kao skup nezavisno razvijenih servisa, koji se izvršavaju na *Cloud*-u. Mikroservisi predstavljaju način razbijanja velikih softverskih rješenja u manje, nezavisne i labavo povezane servise. Jedna od glavnih benefita mikroservisne arhitekture jeste prevazilaženje ograničene skalabilnost takvih monolitnih arhitektura [2].

2. ZADATAK RADA

Zadatak ovog rada je implementacija softvera koji se zasniva na mikroservisnoj arhitekturi, softver određuje broj instanci mikroservisa koji će biti podignuti, kao i da na osnovu algoritma za raspodjelu opterećenja određuje težinu posla koja će biti izvršena na novokreiranim instancama mikroservisa, tako da se računarski resursi najefikasnije iskoriste.

Za testiranje implementiranog rješenja korišćen je algoritam za izračunavanje topološke analize pomoću tehnike rijetkih matrica.

Topološka analiza ima zadatak da napravi povezan model elektrodistributivne mreže u kojem se nalaze samo veze između elemenata koji su direktno povezani ili posredstvom zatvorenih polja. Kao rezultat topološke analize dobijaju se struktuirana mreža po slojevima i vektori koji omogućavaju efikasno kretanje kroz mrežu.

3. TEORIJSKE OSNOVE

Azure Service Fabric je distribuirana sistemska platforma koja omogućava lakše pakovanje, *deploy*-ovanje i upravljanje skalabilnim i pouzdanim mikroservisima i kontejnerima. *Service Fabric* se bavi značajnim izazovima u razvoju i upravljanju aplikacijama u *Cloud*-u,

---

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Srđan Vukmirović, vanr.prof.

čime omogućava korisnicima da ne vode računa o kompleksnim infrastrukturalnim problemima, već da svoj fokus stave na implementaciju softvera. *Service Fabric* predstavlja platformu budućnosti za izgradnju i upravljanje aplikacijama u *Cloud*-u. Da bi shvatili *Service Fabric* klaster neophodno je definisati dva koncepta na kojima se zasniva *Service Fabric*, a to su. čvorovi i klasteri. U tipičnom *Service Fabric*-u jedan čvor predstavlja jednu mašini (fizičku ili virtuelnu). Klaster predstavlja kolekciju čvorova koji su međusobno povezani kako bi kreirali visoko dostupno i pouzdano okruženje za izvršavanje aplikacija i serivsa. Softverski sistemi koji nisu pisani na osnovu nekog softverskog obrasca posjeduju arhitekturu u kojoj su elementi pretežno čvrsto povezani i jako teški za mijenjanje. Kada se priča o mikroservisnoj arhitekturi govori se o kolekciji manjih servisa, ali veličina servisa ne bi trebala biti najvažnija karakteristika. Umjesto toga, najvažnija karakteristika bi trebala biti ta da taj servis bude potpuno nezavisan, da ima autonomiju razvoja, primjene i obima. Sama ideja prelaska na mikroservisnu arhitekturu, zasniva se na dekomponovanju servisa na skup manjih servisa, dokle god nema previše direktnih zavisnosti sa drugim mikroservisima. Prednosti mikroservisne arhitekture*:*

- **Lakše je graditi i odžavati softver.** Ključni princip mikroservisa je jednostavnost. Softver postaje lakši za izgradnju i održavanje kada se podijeli u set manjih servisa.
- **Komponente su jednostavnije.** Značajnu ulogu igra i kompleksnost softvera, s obzirom da je ideja da servisi budu mali i laki za razumijevanje.
- **Poboljšana produktivnost i brzina** Arhitektura mikroservisa se bavi problemom produktivnosti i brzine. Različiti timovi mogu raditi na različitim komponentama istovremeno bez potrebe da čekaju da jedan tim završi dijelove softvera prije započinjanja njihovog rada.
- **Fleksibilnost u izboru tehnologija i skalabilnosti.** Arhitektura mikroservisa omogućava pisanje servisa na različitim programskim jezicima koji mogu nesmetano poslovati sa drugim servisima preko API-ja.
- **Testiranje.** Svaki servis može biti testiran pojedinačno i mogu se testirati komponente koje su već razvijene, dok softver inženjeri rade na drugim.

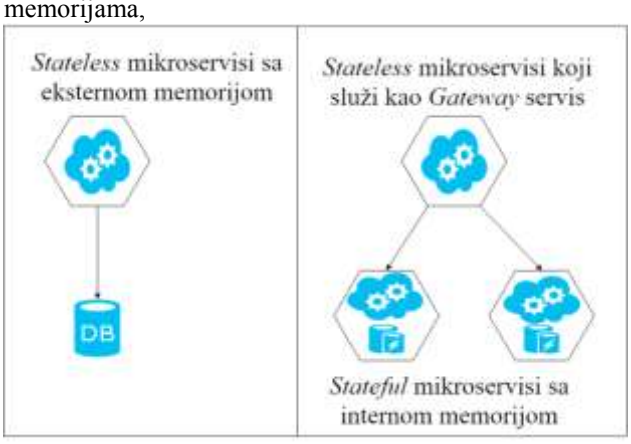
Nedostaci mikroservisne arhitekture:

- **Skuplje održavanje.** Izbor različitih tehnologija prilikom izgradnje komponenti, dovodi do problema nejednakog dizajna softvera što može prouzrokovati povećanje troškova održavanja na duži vremenski rok.
- **Povećana upotreba resursa**. Početne investicije za pokretanje ovih softvera su velike, jer se sve komponente nezavisno pokreću što prouzrokuje potrebu za sopstvenim kontejnerima za rad sa više memorije i CPU-a.
- **Povećana mrežna komunikacija**. Nezavisno pokrenute komponente međusobno komuniciraju preko mreže. Takvi sistemi zahtjevaju pouzdane i brze mrežne veze.
- **Različita interpretacija podataka**. Svaki servis može imati svoju interpretaciju podataka što povećava zahtjeve za obradu podataka.
- **Mrežna sigurnost**. Osnova mikroservisene arhitekture predstavlja komunikacija između servisa, što ovakve softvere čini sigurnosno ranjivim.

Reliable Services je jedan od programskih modela dostupnih na *Service Fabric*-u. *Reliable Services* korisnicima pruža snažan model koji je jednostavan za programiranje, takođe predstavlja lako uklopljiv model koji podržava sve vrste komunikacije HTTP, TCP ili bilo koji drugi protokol. *Reliable Services* se razilikuje od dosadašnjih modela servisa sa kojim se susretala većina korisnika, oni obezbjeđuju [3]

- **Pouzdanost.** Servis ostaje dostupan čak i u nepouzdanom okruženju.
- **Dostupnost**. Servis je konstantno dostupan. *Service Fabric* održava željeni broj instanci.
- **Skalabilnost**. Servisi su odvojeni od specifičnog hardvera i mogu se povećavati ili smanjivati po potrebi dodavanjem ili uklanjanjem hardvera ili drugih resursa. Servisi se mogu kreirati i brisati dinamički putem koda ili komandne linije, omogućavajući da se više instanci pokrene po potrebi, recimo u odgovoru na zahtjeve klijenata.
- **Konzistentnost**. Garantuje se konzistentnost svih podataka koji se čuvaju u servisu.

Postoje dvije vrste servisa: servisi koji čuvaju podatke (eng. *Stateful Reliable Services)* na eksternim memorijma i kolekcijama unutar servisa poput *ReliableDictionary*-a i servisi koji ne čuvaju podatke (eng. *Stateless Reliable Services)* na eksternim memorijama,



Slika 1. *Vrste Reliable Services-a*

Stateless i *Stateful* servisi su komplementarni. Na primer, na slici 1. na desnom dijagramu može se primjetiti da se *stateful* servis može podijeliti na više particija, da bi se pristupilo tim particijama može se koristiti *stateless* servis koji djeluje kao ulazna tačka (engl. *gateway)* tj. servis koji zna kako adresirati svaku particiju na osnovu ključa particije.

4. OPIS APLIKATIVNOG RJEŠENJA

Tema ovog rada jeste skalabilna mikroservisna arhitektura, što obuhvata i sam prelazak monolitnih

aplikacija na mikroservisnu arhitekturu, distibuirano izvršavanje aplikacija i ravnomjernu raspodjelu opterećenja kako bi se računarski resursi iskoristili na najefikasniji način. Efikasno korišćenje računarskih resursa dovodi do znatno boljih rezultata u slučaju prevelikih opterećenja.

4.1 Skalabilnost mikroservisa

Neke komponente mikroservisne arhitekture se češće koriste ili zahtjevaju više resursa od ostalih komponenti. Stoga je neophodno da se te komponente grade tako da budu skalabilne.

Efikasnost je od najveće važnosti u arhitekturi velikih distribuiranih sistema. Kod monolitnih aplikacija mnogo je lakše kvantifikovati efikasnost sistema, dok je procjena i postizanje veće efikasnosti u velikom sistemu mikroservisa, gdje se zadaci dijele na veliki broj malih servisa, znatno teže.

Da bi se osigurala skalabilnost i performantnost mikroservisa neophodno je adekvatno i efikasno koristiti hardverske resusrse, biti svjestan uskih grla sistema, obezbjediti procesuiranje zadataka na skalabilan i performatan način. Skaliranje u okviru *Service Fabric* klastera se postiže na nekoliko različitih načina [7]:

- kreiranjem ili uklanjanjem servisa,
- kreiranjem ili uklanjanjem novih aplikacija,
- pomoću particioniranisanja servisa,
- dodavanjem i uklanjanjem čvorova iz klastera,
- pomoću metrike menadžera resursa klastera.

Implementirano programsko rješenje skaliranje u okviru *Service Fabric*-a vrši dodavanjem i uklanjanjem servisa iz već pokrenute aplikacije u okviru klastera, ostali vidovi skaliranja nisu predmet ovog rada.

4.1.1 Skaliranje mikroservisa dodavanjem i uklanjanjem servisa

Novi servisi mogu se kreirati (ili ukloniti) pošto servisi postaju manje ili više zauzeti. Ovo omogućava da se zahjtevi šalju na više servisnih instanci, što prouzrokuje smanjenje opterećenja postojećih servisa. Kada se kreira servis, Service *Fabric Cluster Resource Manager* stavlja servis u klaster [7].
Novopodignuti servisi mogu biti obrisani nakon izvršenog zahtjeva ili ostavljeni u stanju pripravnosti u iščekivanju novih zahtjeva.

4.2 Arhitektura programskog rješenja

Na temu prelaska sa monolitnih aplikacijana mikroservisnu arhitekturu ispisan je veliki broj radova, jedan od njih jeste [4]. U ovom radu objašnjena je razliku između monolitne i mikroservisne arhitekture, kao i razlozi za prelazak na mikroservisnu arhitekturu. Mikroservise opisuju kao nezavisne procese koji se sami pokreću i koji posjeduju sopstvenu bazu podataka. Takođe, dat je osvrt i na mogućnost mikroservisa da se sami skaliraju.

Osim toga opisana su i neka moćna svojstva mikroservisa koja prouzrokuju visoke troškove i nadvladavaju njihove prednosti kod malih i srednjih softverskih projekata.

Ideja ravnomjernog opterećenja klastera može da se sagleda iz dva ugla:

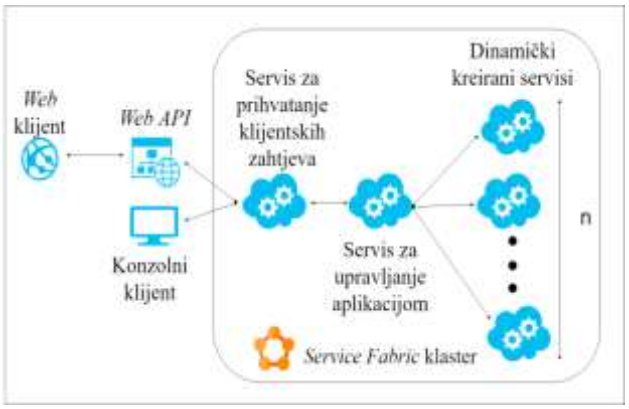
- ravnomjerno opterećenje čvorova, i
- ravnomjerno opterećenje servisa.

Za ravnomjerno opterećenje čvorova zadužena je posebna funkcionalnost u okviru *Service Fabric*-a, koja u slučaju detekcije većeg broja servisa na jednom čvoru u odnosu na ostale čvorove u okviru klastera, pokreće balansiranje opterećenja u okviru klastera, nakon kog se na svakom čvoru u okviru klastera izvršava isti ili približno isti broj servisa.

Predmet ovog rada jeste ravnomjerno opterećenje servisa pomoću algoritma za ravnomjernu raspodjelu, koji jednako ili približno jednako raspoređuje opterećenje na *N* podignutih servisa. Skaliranje mikroservisa na način opisan u okviru poglavlja 4.1.1, predstavlja način na koji se preopterećeni servis podiže dinamički na *N* instanci.

Na slici 2. prikazana je softverska arhitektura implementiranog rješenja. Kao što se može primjetiti arhitektura rješenja je pravljena prema mikroservsnom obrascu topologije centralizovanog upravljanja.

Web klijent se konektuje na *web* API koji predstavlja jedan kontroler koji služu za proslijeđivanje zahtjeva pristiglih od klijenta na mikroservis zadužen za prijem zahtjeva na *Cloud*-u. Kontroler kao i konzolni klijent sa ovim mikroservisom komuniciraju preko WCF komunikacije.



Slika 1. *Arhitektura rješenja*

Servis za prijem zahtjeva na slici 2. prikazan kao *Request* servis služi samo za prenos poruka ka mikroservisu za upravljanje aplikacijom na slici 2. predstavljen kao *Manager* servis. Ideja ovog servisa je da se pomoću njega servis za upravljanje rastereti, ovakav pristup dovodi do značajnog poboljšanja performansi.

U okviru studija [5] objašnjena je upotreba posrednika između klijenta i servisa, kao i njihov doprinos efikasnijoj upotrebi sistema, povećanju pouzdanosti i posebno performansu sistema. Studija pokazuje da dodavanje posrednika u već postojeći distribuirani sistem neće zahtjevati previše truda u poređenju sa poboljšanjem kvaliteta sistema koji on donosi.

Servis za upravljanje aplikacijom predstavlja najvažniju tačku softvera jer posjeduje veliki broj zaduženja. Osnovno zaduženje jeste podizanje servisa koji vrše

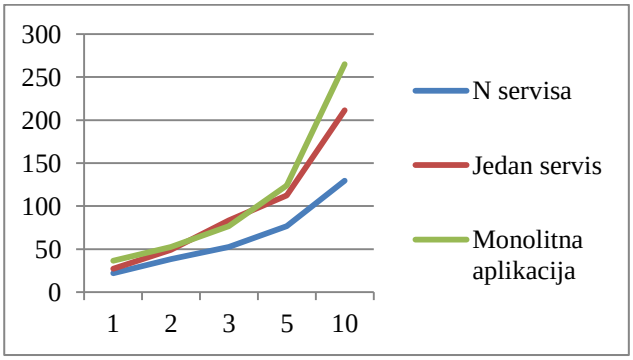
topološku analizu (ili nekog drugog testnog servisa), kao i izvršavanje *Number-partitioning* algoritma.

4.3 *Number-partitioning* algoritam

Problem ravnomjernog opterećenja resursa je prisutan u skoro svim softverima koje vode računa o performansama, tj. da se njihovi resursi pravilno koriste. Apstraktno, ovaj problem se može predstaviti kao problem raspodijele brojeva na *n* podskupova tako da je zbir svakog podskupa približno jednak. Većina algoritama za probleme optimizacije mogu se podijeliti u dvije grupe: algoritmi koji garantuju optimalno rješenje na kraju, ali to se dešava u eksponencijalnom vremenu i algoritmi koji samo donose aproksimativna rešenja ali u polinomijalnom vremenu. Između ove dvije grupe su *anytime* algoritmi [6], koji uglavnom pronalaze bolja rješenja što se duže izvršavaju. U okviru ovog rada rješavan je problem particionisanja brojeva i odnosi se direktno na problem sumiranja podgrupa. Ovaj problem je opširnije opisan u studiji [6]. Prilikom implementacije razmatra se sledeći vrlo jednostavan problem planiranja. Uzimajući u obzir da se naš softver izvšava na *n* identičnih mašina, niz zadataka koje je potrebno obraditi potrebno je rasporediti tako da se na svakoj mašini izvršava ista količina posla tako da se izvši u najkraćem mogućem vremenu.

5. EKSPERIMENTI

Prilikom izrade implementiranog rješenja korišćen je lokalni *Service Fabric Cluster*, koji pruža mogućnost simulacije virtuelnih mašina na *Cloud*-u. Lokalni *Service Fabric Cluster* prilikom testiranja je konfigurisan na pet čorova, što simulira virtuelno *Cloud* okruženje sa pet virtuelnih mašina. Ideja opterećivanja implementiranog rješenja sa deset klijenata koji istovremeno zahtjevaju topološku analizu jeste da se preko ovog testnog slučaja dokaže da je vrijeme izvršavanja topološke analize preko dinamičkih mikroservisa znatno manje u slučaju velikih opterećenja. Grafički prikaz rezltata dat je na slici 3.



Slika 1. *Grafički prikaz rezultata izvršavanja testne aplikacije*

6. ZAKLJUČAK

Računarski svijet se zauvijek promijenio sa pojavom *Cloud*-a, koji omogućava programerima pristup infrastrukturi odmah, jeftino i u skoro beskonačnim skalama. Agilnost *cloud*-a i visoka dostupnost, kao i stalni zahtevi agilnosti savremenog poslovanja su nagomilavali monolitne arhitekture i rezultirali rastom aplikacija zasnovanih na mikroserviserima. Uz sveobuhvatnu platformu za mikroservise, programeri mogu da kreiraju aplikacije koje podržavaju masovnu razmjenu sa visokim performansama, visokom dostupnošću, ekonomičnom efektivnošću i nezavisnim upravljanjem životnog ciklusa, kako preko javnih *cloud*-a, tako i preko privatnih.

Na osnovu grafika sa slike 3. može se zaključiti da implementirano softversko rješenje daje znatno bolje rezultate prilikom izvršavanja velikog broja zahtjeva sa velikim podacima što i jeste bio cilj ovog rada. Dokazano je da se korišćenjem dinamički podignutih servisa, kao i algoritma za raspodjelu opterećenja vrijeme izvršavanja znatno smanjuje.

Jedan od načina da se unapredi implementirano rješenje jeste da se u okviru servisa koji upravlja aplikacijom, prilikom pokretanja kreira skup servisa koji vrše obradu. Ovim bi se postiglo brže rukovanje zahtjevima jer ne bi bilo neophodno čekati da se servisi podignu, iako je vrijeme podizanja servisa malo (mjeri se u milisekundama) ovakav način imlementacije bi smanjio vrijeme izvršavanja zahtjeva

7. LITERATURA

[1] Fundamentals of Azure Second Edition Microsoft Azure Essentials, Michael Collier Robin Shahan

[2] S. Newman, Building Microservices. O'Reilly, 2015

[3] https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-reliable-services-introduction, Pristupljeno 04.09.2018.

[4] Arne Koschel Irina Astrova Jeremias Dötterl, 2017 International Conference on Information Society (i-Society)

[5] Software Brokers for Quality of Services in Service-Oriented Distributed Systems Jiangyun Xu Weichang Du, Faculty of Computer Science, University of New Brunswick Fredericton

[6] A Complete Anytime Algorithm for Number Partitioning, Richard E. Korf ,Computer Science Department University of California, Los Angeles, 1997.

[7] https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-concepts-scalability, Pristupljeno 04.09.2017

Kratka biografija:

Nebojsa Petković rođen je 1994. godine u Ilidži. Osnovnu školu „Jovan Dučić" u Bijeljini završio je 2009. godine, nakon čega upisuje Srednju tehničku školu „Mihajlo Pupin" u Bijeljini, koju završava 2013. Godine i iste godine upisuje Fakultet tehničih nauka u Novom Sadu, smjer Elektroenergetski softverski inženjering. Osnovne studije je završio 2017. godine nakon čega je upisao master studije, smjer Primenjeno softversko inženjerstvo. Ispunio je sve obaveze i položio je sve ispite predviđene studijskim programom.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42:004.738.12
DOI: https://doi.org/10.24867/01BE44Pavicic

POREĐENJE ARHITEKTURA KLIJENTSKIH WEB APLIKACIJA

COMPARISON OF WEB FRONT-END ARCHITECTURES

Dalibor Pavičić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – Primenjene računarske nauke i informatika

Kratak sadržaj –*Izbor odgovarajuće arhitekture i radnog okvira za novi ili postojeći projekat može da predstavlja izazov zbog velikog broja faktora koje je potrebno uzeti u obzir. U ovom radu je izvršeno poređenje poznatih arhitektura za klijentske web aplikacije. Definisan je konceptualni model evaluacije klijentskih radnih okvira koji je potom primenjen na evaluaciju tri popularna radna okvira: Angular, React i Elm. U svakom od navedenih radnih okvira implementirana je i testna aplikacija. Na kraju su date smernice za odabir najadekvatnije arhitekture i radnog okvira u zavisnosti od konteksta.*

Ključne reči: *Evaluacija, Poređenje, Klijentski radni okvir , JavaScript, SPA*

Abstract *–Choosing the right architecture and framework for a new or existing project may be a challenge due to the many factors that one has to consider. This paper contains a comparison of popular architectures for client web applications. The conceptual model of the front-end frameworks evaluation was defined and applied to the evaluation of three popular options: Angular, React and Elm. A demo application has been implemented by using each of them. The paper also provides guidelines for selecting the most suitable architecture and framework for the given context.*

Keywords: *Evaluation, Comparison, Front-end framework, JavaScript, SPA*

1. UVOD

U poslednjih nekoliko godina došlo je do naglog razvoja klijentskih web aplikacija i prelaska sa tradicionalnog višestraničnog modela (*multi-page*) na jednostranični model (*single-page*). Jedan od glavnih nedostataka višestraničnog modela jeste loš odziv web stranica. SPA (*single page applications*) aplikacije u kombinaciji sa AJAX tehnologijom predstavljaju web aplikacije koje staju na jednu stranicu i imaju za cilj da obezbede fluidnije korisničko iskustvo.

Porast popularnosti SPA aplikacija doveo je do toga da se više procesiranja odvija na klijentskoj strani što je uslovilo razvoj programskih jezika za pisanje klijentskih aplikacija, prvenstveno JavaScript-a.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio doc. Milan Segedinac.

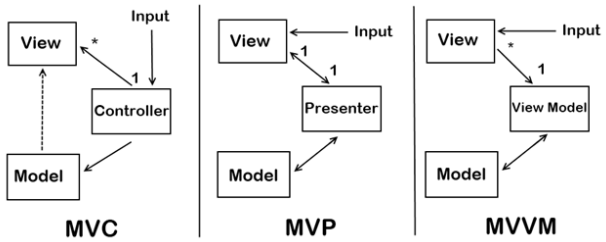
Nastali su novi radni okviri sa ciljem da olakšaju implementaciju i strukturiranje aplikacija i da obezbede gotova optimizovana rešenja za razvoj kompleksnih funkcionalnosti. Međutim, intenzivan razvoj JavaScript radnih okvira doveo je do tzv. „JavaScript zamora“ koji se javlja kada je od mnoštva alata potrebno izabrati jedan koji bi bio najpogodniji za specifičan projekat. Odatle i motivacija za ovaj rad koji ima za cilj da definiše smernice i olakša izbor odgovarajuće arhitekture i radnog okvira uzimajući u obzir sve relevantne faktore.

2. ARHITEKTURA KLIJENTSKIH WEB APLIKACIJA

Pod pojmom arhitektura klijentske web aplikacije podrazumeva se izbor suštinskih elemenata koji čine jednu klijentsku web aplikaciju. Ovde će biti reči o sledećim arhitekturama: MV* (*Model View**), FLUX i MVU (*Model View Update*).

2.1. MV*

U MV* arhitekture spadaju: MVC, zatim MVP (*Model View Presenter*) i MVVM (*Model View ViewModel*) (Slika 2.1).



Slika 2.1 *Šematski prikaz ključnih razlika između MVC, MVP i MVVM arhitektura*

MVC

MVC je verovatno najpopularnija arhitektura u poslednjih 30 godina koju koriste milioni programera u različitim projektima, nezavisno od programskog jezika. Glavna inovacija koju je MVC uneo u razvoj softvera jeste konačno razdvajanje podataka od njihove vizuelne reprezentacije što je do tada još uvek bio nedovoljno istražen koncept.

MVC definiše sledeće činioce:

- model (*Model*) – sadrži stanje aplikacije kao i domenske podatke
- prikaz (*View*) – predstavlja korisnički interfejs sa kojim korisnici vrše interakciju
- kontroler (*Controller*) – predstavlja vezu između modela i prikaza i obično je nadležan za orkestraciju toka komunikacije u aplikaciji

MVP

Ova arhitektura dolazi do izražaja u situacijama kada je potrebno koristiti iste prikaze ili ponašanja u različitim delovima iste aplikacije ili u potpuno različitim projektima. Naročito je korisna kod aplikacija koje treba da podrže različite platforme pri čemu je potrebno koristiti iste podatke, komunikacioni sloj i ponašanja ili u slučajevima kada se prikaz želi dinamički zameniti u toku izvršavanja ili kompajliranja aplikacije. Ključne razlike u odnosu na MVC sadržane su u sledećem:

- Umesto kontrolera se koristi prezenter.
- Veza između prezentera i prikaza nije 1 na prema više, kao što je slučaj sa kontrolerom kod MVC-a, nego je uvek 1 na prema 1.
- Najkorisnija implementacija MVP-a je u slučaju kada su prikazi pasivni jer je tada zamena prikaza jednostavna dokle god je ispoštovan ugovor između prezentera i prikaza.

MVVM

MVVM zadržava razdvajanje modela i prikaza kao i MVP, ali donosi i određene razlike u odnosu na druge dve MV* arhitekture. MVVM je sličan MVP-u s tim što za razliku od njega prikaz više nije pasivan.

Prva veća razlika u odnosu na MVP je to što umesto prezentera imamo model za prikaz (*ViewModel*). To je objekat koji služi kao most između podataka u modelu i prikaza. Ovaj objekat je u suštini odgovoran za izlaganje podataka iz modela na prikaz i pripremu podataka na način kako prikaz očekuje pa je logika modela za prikaz tesno povezana s onim što prikaz treba da prikaže.

Druga važna karakteristika modela za prikaz je činjenica da je veza između modela za prikaz i samih prikaza 1 na prema više. Dakle, jedan model za prikaz može da se koristi na više prikaza ili komponenti.

2.2. FLUX

Flux je arhitektura za klijentske web aplikacije koja je nastala u Facebook-u. Flux arhitektura definiše sledeće činioce: dispečer (*dispatcher*), skladišta (*stores*) i prikaze (*views*). Postoje i kontroleri, ali je njihova uloga nešto drugačija u odnosu na MVC. Kontroleri se obično nalaze na vrhu hijerarhije prikaza. Oni dobavljaju podatke iz skladišta i prosleđuju ih komponentama koje su ispod njih u hijerarhiji. Umesto MVC-a, Flux favorizuje jednosmerni tok podataka.

2.3. MVU

MVU (*Model View Update*), poznatiji kao "Elm arhitektura" predstavlja jednostavan, ali moćan šablon za strukturiranje web aplikacija. To je čisto funkcionalna arhitektura koja se vezuje za Elm, funkcionalni programski jezik za web, ali su ideje i koncepti koje ova arhitektura donosi primenljivi i u drugim programskim jezicima. Činioci koje definiše ova arhitektura su:

- model (*Model*) – sadrži stanje aplikacije kao i domenske podatke (slično MVC-u)
- prikaz (*View*) – funkcije koje generišu prikaz na osnovu modela
- akcije (*Actions*) – događaji koji nastaju kao posledica korisničkih interakcija
- ažuriranje (*Update*) – funkcije koje rukuju korisničkim interakcijama i generišu novo stanje aplikacije na osnovu prethodnog stanja i akcije

Novije arhitekture poput FLUX-a i MVU-a donose nove ideje i koncepte, ali se takođe oslanjaju na već poznate i dokazane principe iz prošlosti koji se susreću u MV* arhitekturama. Ove arhitekture su ugrađene u mnoge popularne radne okvire i biblioteke te stoga poznavanje ovih arhitektura može biti od velike koristi na realnim projektima, naročito prilikom donošenja odluke o tome koju arhitekturu odabrati za konkretan projekat.

Ne postoji univerzalno pravilo koje definiše kako treba da izgleda struktura projekta, ali svaka od opisanih arhitektura ima svoju namenu, kao i svoje prednosti i mane. "Arhitekturu softvera treba više shvatiti kao putovanje nego kao odredište, više kao tekući proces nego kao krut predmet za upotrebu" [**1**].

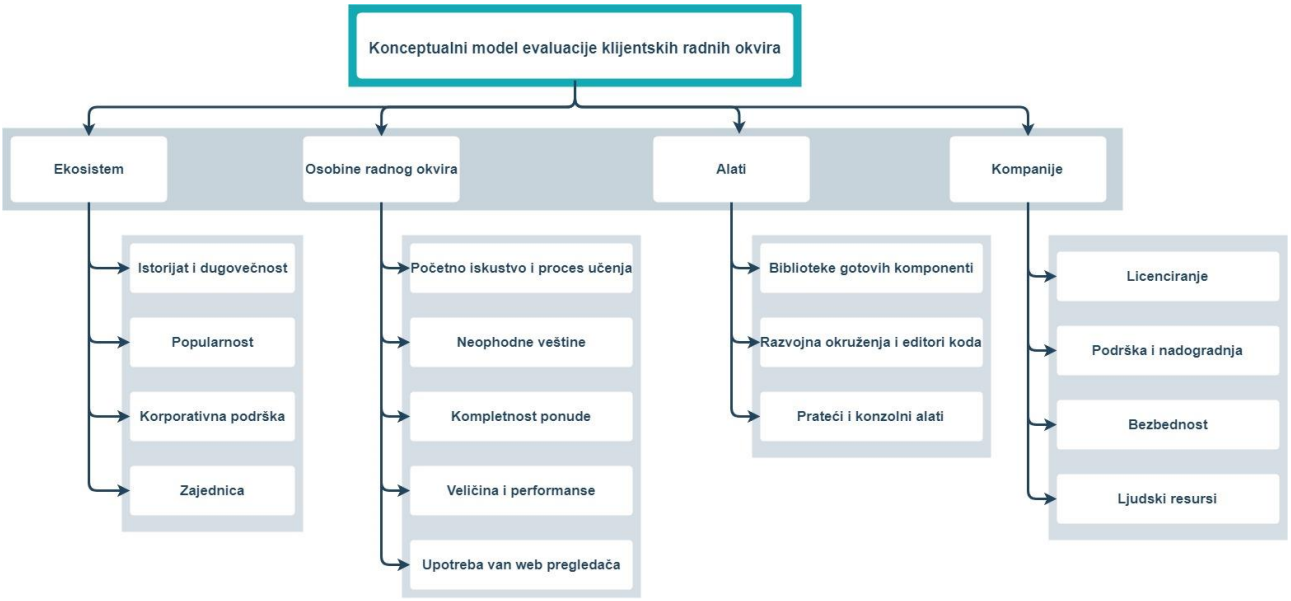
3. EVALUACIJA IZABRANIH BIBLIOTEKA I RADNIH OKVIRA

Bilo da je u pitanju potpuno novi projekat ili je potrebno migrirati ili modernizovati postojeću web aplikaciju, web programerima se nameće potreba za evaluacijom radnih okvira za web. Od mnoštva dostupnih i kvalitetnih radnih okvira ovde će se evaluacija ograničiti na sledeća tri:

- Angular
- React
- Elm

Postoji mnogo studija i članaka koji se bave neformalnim poređenjem klijentskih radnih okvira od kojih se većina, poput [**2**] i [**3**], bazira na subjektivnom mišljenju autora. Nešto formalnije poređenje, ali ipak ograničeno na specifičan domen je izvršeno u [**4**]. Ipak, postoje i pokušaji da se definiše formalna metodologija i opštiji kriterijumi koji su primenljivi u širem kontekstu kao što je slučaj sa [**5**] i [**6**]. U ovom radu su iskorišćeni, sistematizovani i sumirani neki od rezultata prethodno navedenih studija i istraživanja i na osnovu njih je definisan konceptualni model evaluacije klijentskih radnih okvira (Slika 3.1). Predloženi model je primenjen na analizu tri popularna radna okvira, Angular-a, React-a i Elm-a, uz pokušaj da se da odgovor na pitanje koji od njih koristiti za postizanje najboljih rezultata u razvoju date web aplikacije koju određena kompanija ili programer želi da napravi.

Prikazani konceptualni model se satoji od faktora i kriterijuma koji se mogu podeliti na kvantitativne u koje spadaju: popularnost, zajednica, ekosistem, veličina i performanse i kvalitativne kao što su: lakoća učenja, korporativna podrška, podrška za razvojna okruženja, licenciranje i bezbednost. Predloženi model ne daje numeričke ocene evaluiranim radnim okvirima jer je težinu i važnost pojedinačnih faktora kao i izbor kriterijuma potrebno prilagoditi jedinstvenom kontekstu kompanije ili razvojnog tima kao i projektu za koji se radi evaluacija.



Slika 3.1 *Konceptualni model evaluacije klijentskih radnih okvira*

3.1. Ekosistem

Ekosistem određene biblioteke uključuje sledeće faktore: istorijat razvoja, zajednicu, popularnost, uključenost korporacija kao i poznate aplikacije koje koriste tu biblioteku ili su napravljene uz pomoć nje [**7**].

Istorija i starost su korisni faktori jer daju jasniju sliku o tome kako je biblioteka započeta i kolike su šanse da će opstati u doglednoj budućnosti. Angular je najstariji među evaluiranim kandidatima ako se računa prva verzija koja je izašla 2010. godine pod nazivom AngularJS. Međutim, 2016. godine je izašla nova verzija koja je u potpunosti promenjena i iznova napisana. Migracija sa starije na noviju verziju gotovo da zahteva pisanje aplikacija ispočetka što je jedna od velikih zamerki Angular-u. Prva verzija React-a je izašla 2013. godine i od tada nije bilo značajnijih izmena API-ja što ukazuje na stabilnost i pouzdanost. Slična je situacija i sa Elm-om koji je nastao 2012. godine.

Kvantitativne mere poput broja zvezdica na GitHub platformi, broja skidanja sa npm registra biblioteka kao i zastupljenost na StackOverflow-u (Tabela 1) ukazuju na popularnost analiziranih radnih okvira. Iako daju uvid u to koliko se svaki od njih koristi, ovi brojevi ne prave razliku između stvarne upotrebe u praksi i upotrebe u svrhu učenja ili isprobavanja pa ih treba razmotriti u kombinaciji s primerima konkretnih aplikacija koje ih koriste u praksi.

Tabela 1 *Popularnost biblioteka izražena kvantitativnim merama*

	GitHub	npm	StackOverflow
A	~31000	~1,93 miliona	~87000
R	~76000	~5.85 miliona	~67000
E	~4500	~65000	~1200

Sve tri biblioteke imaju neki oblik korporativne podrške. Zaposleni u Google-u aktivno nadziru i održavaju Angular, dok je s React-om slična situacija sa zaposlenima u Facebook-u. Iza Elm-a ne stoji velika korporacija kao što je slučaj sa druge dve biblioteke. Međutim, projekat su podržale kompanije Prezi i NoRedInk, a uspostavljena je i fondacija za Elm softver (*Elm Software Foundation*) čija je misija da promoviše, zaštiti i unapredi programski jezik Elm.

Iako je teško kvantitavno izmeriti veličinu zajednice, mogu se razmotriti određeni faktori koji govore o aktivnosti ekosistema oko određene biblioteke. Jedan od tih faktora je broj npm biblioteka i paketa dostupnih programerima (Tabela 2).

Tabela 2 *Broj dostupnih paketa za svaku od izabranih biblioteka*

Biblioteka	Broj paketa
Angular 2+	~18,322 npm
React	~8,174 npm
Elm	~992

3.2. Osobine radnog okvira

Proces učenja svakog od analiziranih radnih okvira u velkoj meri zavisi od prethodnog iskustva. Angular zahteva učenje TypeScript-a i specifične sintakse koja se koristi u UI šablonima. Većina dostupnog React koda koristi ES6 sintaksu i JSX koje je neophodno savladati za ozbiljniji rad sa React-om. Elm je zaseban jezik koji se kompajlira u JavaScript, a pri tom je i čisto funkcionalan pa je pored specifične sintakse potrebno savladati i koncepte funkcionalnog programiranja što može dodatno da oteža početno iskustvo u radu s Elm-om.

Angular u osnovnom paketu sadrži neke od neophodnih alata za pravljenje složenih aplikacija: upravljanje korisničkim interfejsom, upravljanje stanjem aplikacije, rutiranje i testiranje i zahteva manje dodatnih biblioteka u odnosu na React i Elm. To može da bude prednost jer je početni razvoj znatno brži i zahteva manje odluka, ali isto tako i mana jer je tada i fleksibilnost znatno manja.

Kada su performanse u pitanju nema značajnijih odstupanja između kandidata. Neki *benchmark* testovi [**8**] pokazuju da je Angular najsporiji, React nešto brži i Elm najbrži, ali razlike nisu velike.

U poslednje vreme klijenti sve češće očekuju i zahtevaju da aplikacije koje rade u web pregledačima rade i na mobilnim uređajima pa čak i nativno na desktopu. Za svakog od kandidata postoje gotova rešenja ili makar pokušaji za rešavanje opisanog problema. Dve popularne opcije za Angular su: NativeScript i Ionic. Za React su to React Native i react-native-renderer. Postoji i projekat pod nazivom elm-native-ui kao pokušaj da se Elm koristi za mobilne aplikacije.

3.3. Alati

Za sva tri evaluirana radna okvira postoje odlične biblioteke UI komponenti kao što su Angular-Material, Material-UI, React-Bootstrap, elm-bootstrap kao i veliki broj gotovih individualnih komponenti.

Od razvojnih okruženja treba istaći Visual Studio Code i WebStorm koji pružaju odličnu podršku za Angular i React uključujući isečke i šablone koda kao i moćne alate za refaktorisanje dok je podrška za Elm prisutna nešto slabija. Pored razvojnih okruženja postoje i dodatni alati za automatizaciju zadataka kao što su: kreiranje skeleta projekta, provera stila koda i pokretanje jediničnih testova, server za lokalni razvoj uz automatsko osvežavanje aplikacije tokom pisanja koda, alati za pakovanje i stavljanje koda u produkciju itd. Ovde se blaga prednost može dati Angular-u, ali i druga dva kandidata raspolažu solidnim alatima za navedene zadatke.

3.4. Kompanije

Poslednja grupa evaluacionih faktora tiče se direktno kompanija. Tu spadaju: licenciranje, podrška, bezbednost i mogućnost zapošljavanja programera sa odgovarajućim veštinama [**7**]. Ovi faktori se mogu primeniti i na manje organizacije, ali oni u slučaju kompanija mogu biti presudni prilikom donošenja odluke o biblioteci ili proizvodu koji ispunjava sve ostale kriterijume.

Kompanije koje koriste alate otvorenog koda za pravljenje softvera treba da pažljivo analiziraju licence pod kojima se ti alati održavaju i da se osiguraju da se te licence ne kose sa upotrebom alata u kompaniji. Evaluirani radni okviri su pod MIT licencom [**9**] sa izuzetkom React-a pre verzije 16 (BSD + Patenti [**10**]) koja može uzrokovati potencijalne zakonske probleme *startup* kompanijama.

Ni za jedan od analiziranih radnih okvira nisu prijavljeni veći bezbednosni problemi, a ako ih je i bilo uglavnom su brzo rešavani i distribuirani. To je važan podatak za kompanije koje se odluče za neki od njih s obzirom na svakodnevni porast poslova koji se obavljaju na mreži.

Kada se bira web radni okvir za pravljenje preduzetničkih aplikacija u softverskim kompanijama potrebno je razmotriti i ljudski faktor i osigurati da trenutno zaposleni programeri mogu da ga nauče kao i to da je moguće angažovati dodatne ljudske resurse ukoliko se za to javi potreba. Aktuelne statistike [**8**] pokazuju da je React trenutno najpopularniji dok popularnost Angular-a opada. Interesovanje za Elm raste, ali je daleko manje u odnosu na druga dva kandidata.

4. STUDIJA SLUČAJA

U svakom od analiziranih radnih okvira, Angular-u, React-u i Elm-u razvijena je istovetna CRUD aplikacija koja predstavlja registar kompanija i njihovih zaposlenih. Ova aplikacija sadrži većinu elemenata složene klijentske web aplikacije uključujući dobavljanje podataka preko HTTP-a, rutiranje, generičke komponente za tabelarne prikaze, forme i validaciju podataka. Razvoj testne aplikacije je omogoućio i praktičnu proveru zaključaka i tvrdnji vezanih za arhitekture i tehnologije koji su istaknuti u ovom radu.

4.1 Komponente

Tipične komponente koje su implementirane u testnoj aplikaciji su: tabela koja prikazuje podatke uz mogućnost filtriranja, sortiranja i straničenja, navigacioni meni, tekstualno polje i dugme za akcije. React i Elm su se pokazali odličnim za kreiranje malih pasivnih komponenti koje nemaju interno stanje već se definišu kao klasične funkcije koje dobijaju ulazne podatke i vraćaju HTML. Kreiranje komponenti u Angular-u je nešto složenije zbog sintakse, ali i komplikovanijeg mehanzima za komunikaciju sa klijentskim komponentama.

4.2 Sintaksa

Implementacija svake od testnih aplikacija zahtevala je korišćenje specifične sintakse. Kod Angular-a su to TypeScript i direktive, kod React ES6+ i JSX a kod Elm-a sintaksa i koncepti samog jezika. Moglo bi se reći da se trud uložen u učenje ES6 sintakse verovatno najviše isplati s obzirom da se stečeno znanje može upotrebiti u različitim kontekstima, čak i za razvoj serverskih aplikacija.

4.3 UI šabloni

Kod pisanja UI šablona uočena je sličnost između React-a i Elm-a koji su uneli inovacije i doveli do preispitivanja dugogodišnjih „najboljih“ praksi. JSX sintaksa koju koristi React kao i Elm-ove *view* funkcije ponovo uvode mešanje UI šablona sa programskom logikom iako su programeri decenijama ranije pokušavali da ih razdvoje smatrajući takav pristup lošom praksom. Zagovornici ovakvog pristupa smatraju da je razdvajanje šablona od logike samo razdvajanje tehnologija, ali ne i nadležnosti. Stoga savetuju da programeri treba da razvijaju komponente umesto UI šablona jer su to jedinice koje se mogu ponovo upotrebiti, kombinovati i testirati u izolaciji [**11**]. Angular je negde između ova dva pristupa jer su HTML šabloni i dalje prisutni s tim što su obogaćeni direktivama.

4.4 Upravljanje stanjem i povezivanje podataka

Pored komponenti, stanje predstavlja važan koncept u izgradnji web korisničkih interfejsa. Komponente opisuju stanje korisničkog interfejsa u bilo kom trenutku vremena. Posmatranje korisničkog interfejsa kao “funkcije” stanja je u manjoj ili većoj meri zastupljeno u sve tri implementacije testne aplikacije. Tok podataka kroz React i Elm aplikacije je jednosmeran, dok je kod

Angular-a dvosmeran. Pristup koji koriste prva dva radna okvira jeste nešto opširniji za pisanje, ali omogućava bolju kontrolu, lakše upravljanje stanjem, pa čak i bolje performanse. Nasuprot tome, kod Angular-ovog dvosmernog povezivanja sam radni okvir interno menja stanje modela kada se neki element poput polja za unos ažurira što rezultuje čistijim kodom ali je u kompleksnijim slučajevima teže ispratiti tok podataka.

4.5 Fleksibilnost

Kada je u pitanju fleksibilnost, prednost se može dati React-u. React se može koristiti kao standardna JavaScript biblioteka jednostavnim uključivanjem *script* taga na stranicu. Pored toga, React je više biblioteka nego radni okvir i zagovara suprotnu filizofiju u odnosu na Elm i Angular koji nude sveobuhvatan radni okvir. React omogućava zamenu malih delova aplikacije boljim bibliotekama, umesto čekanja da sam radni okvir evoluira i uvede novine.

5. ZAKLJUČAK

U ovom radu su opisane neke od popularnih arhitektura za klijentske web aplikacije. Definisan je konceptualni model evaluacije klijentskih radnih okvira koji je primenjen na evaluaciju tri popularna radna okvira po svakom od uspostavljenih kriterijuma. Implementirana je i testna aplikacija u cilju istraživanja i sticanja praktičnog iskustva sa svakim od analiziranih radnih okvira.

Odgovor na pitanje koju arhitekturu i radni okvir ili biblioteku izabrati za konkretan projekat nije jednostavan i umnogome zavisi od konteksta i brojnih faktora na koje je ovaj rad nastojao da ukaže. Što se arhitektura tiče, blaga prednost se može dati novijim arhitekturama poput FLUX-a i „Elm arhitekture“ kao i njihovim varijacijama jer sadrže ugrađeno prethodno znanje i iskustvo, a ujedno donose i nove ideje i koncepte koji su prilagođeni potrebama savremenog web razvoja. O analiziranim radnim okvirima su izvedeni sledeći zaključci:

- Ukoliko je potrebno rešenje koje ima ugrađeno sve što je potrebno za pravljenje kompleksnih aplikacija, Angular je najbolji izbor.
- Ako se cilja na biblioteku iza koje stoje veliki korporativni sponzori, Angular i React su dobre opcije.
- Razvojni timovi koji dobro poznaju Java i C# programske jezike i preferiraju sisteme sa statičkim tipovima će se najlakše snaći s Angular-om.
- Za razvojne timove sastavljene od iskusnih JavaScript “gurua” i ljubitelja funkcionalnog programiranja React i Elm su odlične opcije.
- Organizacije koje su u razvoju i žele da unajme programere koji su motivisani za rad sa najnovijim tehnologijama, React i Elm su takođe dobar izbor.

6. LITERATURA

[1] Robert Cecil Martin, *Clean Architecture*.: Prentice Hall, 2017.

[2] Jens Neuhaus. Medium. [Online]. https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176

[3] Cory House. freeCodeCamp. [Online]. https://medium.freecodecamp.org/angular-2-versus-react-there-will-be-blood-66595faafd51

[4] Jaakko Voutilainen, "Evaluation of Front-end JavaScript Frameworks for Master Data Management Application Development," Metropolia University of Applied Sciences, Bachelor’s Thesis 2017.

[5] Julia Plekhanova, "Evaluating web development frameworks: Django, Ruby on Rails and CakePHP," Temple University, Philadelphia, IBIT Report 2009.

[6] Changpil Lee, "An Evaluation Model for Application Development Frameworks for Web Applications," The Ohio State University, Columbus, Thesis 2012.

[7] Brandon Satrom. (2018, Jan.) Telerik. [Online]. https://www.telerik.com/whitepapers/kendo-ui/choosing-the-right-javascript-framework-for-your-next-web-application

[8] Stefan Krause. (2017) Results for js web frameworks benchmark – round 7. [Online]. http://www.stefankrause.net/js-frameworks-benchmark7/table.html

[9] Wikipedia. [Online]. https://en.wikipedia.org/wiki/MIT_License

[10] Open Source Initiative. [Online]. https://opensource.org/licenses/BSDplusPatent

[11] Pete Hunt. (2013) YouTube. [Online]. https://www.youtube.com/watch?v=x7cQ3mrcKaY&t=11s

[12] The State Of JavaScript. [Online]. https://stateofjs.com/2017/front-end/results/

Kratka biografija:

Dalibor Pavičić rođen je u Novom Sadu 1992. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Primenjene računarske nauke i informatika odbranio je 2018.god.
kontakt: dalibor.pavicic92@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/01BE45Maric

AUTOMATIZACIJA INSTRUMENTALIZACIJE KODA POMOĆU *ROSLYN* GENERATORA KODA

AUTOMATION OF INSTRUMETALIZATION OF CODE USING *ROSLYN* CODE GENERATOR

Luka Marić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Automatizovana instrumentalizacija koda predstavlja proces modifikacije koda kako bi se omogućilo vođenje evidencije o događajima koji su se desili u toku izvršavanja softvera od interesa. Motivacija za implementaciju predstavlja ušteda vremena koja može da se dobije ukoliko se čitav proces automatizuje. Za implementaciju rešenja korišćena je Roslyn platforma za pisanje korisnički definisanih kompajlera. Razlog upotrebe Roslyn platforme leži u objektnom modelu koji pruža i intuitivnom načinu za generisanje novog koda što predstavlja centralni deo rada koji će biti opisan.*

Ključne reči: *Roslyn, kod generacija, C#, kompajler*

Abstract – *Automated instrumentation of code is a process of code modification to allow it to keep records of events that occurred during the execution of software of interest. The motivation for implementation is the time savings that can be gained if the whole process is automated. To implement the solution, Roslyn platform for writing user-defined compilers is used. The reason for using the Roslyn platform lies in an object model that it provides and an intuitive way of code generation, which is a central part of the work that will be described.*

Keywords: *Roslyn, code generation, C#, compiler*

1. UVOD

Još od samog početka informatičke revolucije, pa do danas prodor tehnologije u sve sfere ljudskog života nezaustavljivo raste. Tako danas, retko koja grana industrije nije zavisna od raznovrsnih tehnoloških proizvoda. Kako bi ti proizvodi mogli da funkcionišu na predviđen način, uz što manje uplitanja ljudi, neophodno je da njegov rad bude kontrolisan od strane softvera koji je pisan za njega.

Sa napretkom tehnologije, cena potrebnih uređaja je drastično padala, što je prouzrokovalo dalji rast industrija zavisnih od istih. Napredak tehnologije nije samo uticao na cenu uređaja, nego i na njihovu raznolikost, u zavisnosti od njihove namene.

Naravno, kao što je ranije rešeno, kako bi se što optimalnije koristili uređaji, morao biti napisan i softver za njihovo upravljanje.

Pošto je gotovo nemoguće da se u prvom pokušaju, ili uopšte napiše "savršen" softver, od presudnog značaja za profit je vreme provedeno u procesu otkrivanja izvora problema kao i načina za njegovo rešavanje. Nekad je, u slučaju obimnih programskih sistema, veoma teško pronaći izvor problema dok njegovo rešavanje može da bude trivijalno. Pored toga, postoji šansa da se napiše takav softver čije će performanse biti narušene zbog malog broja funkcionalnosti koje nisu dovoljno optimizovane. U tom slučaju, procesor bi gubio dragoceno vreme na izvršavanje tih funkcionalnosti, umesto da to isto vreme posveti nečemu drugom. Zbog te činjenice, proces instrumentalizacija izvornog koda softvera predstavlja proces od izuzetne važnosti.

U kontekstu razvoja softvera, termin instrumentalizacija se odnosi na mogućnost nadzora ili merenja nivoa performansi proizvoda kao i dijagnostifikovanja različitih problema i čuvanja informacija kako bi se problem što lakše mogao lokalizovati i odstraniti. Instrumentalizacija može biti ostvarena različitim tehnikama vođenja evidencije u toku izvršavanja softvera [3]. Takođe, veoma korisno može da bude i zapisivanje u kom trenutku se događaji od važnosti dešavaju.

2. ROSLYN

Roslyn predstavlja jednu od novijih platformi za razvoj kompajlera. Suštinski *Roslyn* je grupa *open-source* kompajlera i API-ja namenjenih za vršenje analize izvornog koda napisanog u C# i *Visual Basic* programskim jezicima.

Motivacija za njegov razvoj leži u tome što je pre njega proces kompajliranja predstavljao crnu kutiju. Termin crna kutija se odnosi na veoma malo informacija o tome šta se zapravo događa u toku kompajliranja, i da su poznati samo ulazi, u vidu izvornog koda, i izlazi u obliku objektnog fajla. Način na koji kompajler u toku prevođenja programa stiče znanja o samom izvornom kodu je dugo vremena bio nepoznat. Nije bilo moguće da se bilo koji način prekine proces kompajliranja kako bi se mogli proučiti međukoraci. Kako kompleksnost razvijanog softvera raste, raste i potreba za razumevanjem načina na koji kompajler stiče potrebne informacije kako bi što bolje preveo izvorni kod. To je omogućeno *Roslyn* platformom.

2.1. Sintaksa

Najosnovnija struktura podataka izložena od strane kompajlerskih API-ja je sintaksno stablo. Svrha ovih

---

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Branislav Atlagić, docent.

sintaksnih stabala je da grafički predstave leksičku I sintaksnu strukturu izvornog koda. Razlozi za to su :

1. Omogućavanje različitim alatima (razvojno okruženje, alati za analizu koda itd.) lakše kretanje kroz posmatrani kod, njegovo bolje razumevanje kao i lakše procesuiranje izvornog koda.
2. Omogućavanje alatima, kao što su alati za refaktoring ili razvojna okruženja, da modifikuju ili kreiraju novi izvorni kod bez potrebe da se koristi tekstualni editor.

2.2. Sintaksno stablo

Sintaksno stablo ima tri ključne osobine. Prva osobina je ta da svako sintaksno stablo sadrži sve informacije o izvornom kodu: svaku gramatičku konstrukciju, leksički token, čak i delove koda koji predstavljaju beline, komentare i preprocesorske direktive.
Mogućnost da se kreirano stablo ponovo transformiše u tekstualni format predstavlja drugu od tri ključne osobine. Od svakog sintaksnog čvora, moguće je dobiti tekstualnu reprezentaciju podstabla koje kao koren ima izabrani sintaksni čvor. Zahvaljujući ovoj osobini, moguće je vršiti potrebne modifikacije koje će se kasnije odraziti na izvorni kod.
Treću osobinu sintaksnog stabla predstavlja činjenica da je to *immutable* struktura podataka, a to znači da njeno eventualno menjanje neće uticati na posmatrani objekat, već će biti kreiran potpuno novi objekat koji će se od originala razlikovati u načinjenoj izmeni. Takođe, ove strukture mogu da se koriste od strane više različitih niti, koje se izvršavaju u sklopu programa, bez potrebe za proverom da li je neka druga nit vršila modifikacije na određenim elementom.

2.3. Sintaksni čvor

Sintaksni čvor predstavlja osnovni činilac sintaksnog stabla. Reprezentuje sintaksnu konstrukciju kao što su deklaracije, iskazi i izrazi. Osnovna klasa u objektnom modelu koja predstavlja sintaksni čvor je *SyntaxNode* klasa, iz koje se dalje izvode klase koje predstavljaju čvorove specifične kategorije.

2.4. Sintaksni token

Sintaksni token predstavlja terminalni element gramatike jezika i najmanji sintaksni deo izvornog koda. U sklopu stabla se nikada ne nalaze u poziciji da predstavljaju roditeljski čvor nekom drugom čvoru. Sintaksni token može biti član stabla koji predstavlja identifikator, ključnu reč, broj i interpunkcijski znak. Za razliku od prethodno opisanih sintaksnih čvorova, u sklopu modela podataka Roslyn platforme jedna klasa je zadužena za predstavljanje svih tipova tokena, gde vrednosti svojstava klase variraju i zavise od tipa tokena kojeg modeluju.

2.5. Sintaksna trivija

Sintaksne trivije predstavljaju delove stabla koji nisu od značaja za dobro razumevanje samog koda. Tako sintaksne trivije mogu reprezentovati beline, komentare ili preprocesorske direktive. Sintaksne trivije se ne smatraju kao deca čvorovi drugih čvorova u stablu iz razloga što ne predstavljaju deo sintakse odabranog jezika i mogu da se nađu između bilo koja dva tokena u stablu. Uprkos tome, one su našle mesto u stablu zbog njihove važnosti u procesu refaktoringa i kako bi se osiguralo potpuno poklapanje sintaksnog stabla i izvornog koda na osnovu kog je ono kreirano.
Postoje dve vrste trivija koje se mogu naći u sintaksnom stablu: trivije koje prethode tokenu (Leading Trivia) i trivije koje slede nakon tokena (Trailing Trivia).

2.6. Greške

Važno je istaći da će sintaksno stablo biti kreirano i u slučaju da se u izvornom kodu detektuje greška. Pod greškom se smatra bilo koje neslaganje izvornog koda sa definisanom sintaksom izabranog programskog jezika. U tom slučaju parser može da konstruiše stablo na dva načina:

- Ukoliko parser očekuje token na određenom mestu u stablu i to nije slučaj, parser će ubaciti poseban token koji reprezentuje token koji nedostaje. Token će biti ubačen na tačno ono mesto gde je očekivan originalni token. Ubačeni token se po tipu ni na koji način ne razlikuje od očekivanog, jedino mu je svojstvo *IsMissing* postavljeno na vrednost *true*.
- Parser će preskočiti sve dok ne naiđe na token od kog može da nastavi proces parsiranja. Svi preskočeni tokeni se smeštaju u kolekciju koja predstavlja triviju tipa *SkippedTokens*.

2.7. Generisanje koda pomoću *Roslyn* platforme

Ranije su strategije vezane za generisanje koda bile bazirane na strukturiranju metapodataka po šablonu T4 ili u objekte klasa sadržanih u bibliotekama poput *CodeDom*. Nešto više o pomenutim strategijama biće rečeno u nastavku poglavlja. Još pre upotrebe šablona T4 i biblioteke *CodeDom*, inženjeri su bili primorani da direktno koriste *StringBuilder* kako bi generisali kod.
T4 šabloni predstavljaju mešavinu tekstualnih blokova i kontrolne logike za generisanje tekstualnih datoteka. Kontrolna logika može biti pisana u programskom jeziku C# ili *Visual Basic*. Generisani tekst može biti internet stranica, resursna datoteka ili izvorni kod pisan u željenom programskom jeziku [2].
Biblioteka *CodeDom* pruža tipove podataka koji u najvećem broju slučajeva pokrivaju opšte tipove koji se javljaju u izvornom kodu. Generisanje koda se vrši kreiranjem objektnog grafa koji sadrži elemente biblioteke *CodeDom*.
Roslyn pruža bolju kontrolu nad kodom u poređenju sa *CodeDom* bibliotekom. Generisanje izvornog koda u bilo kom programskom jeziku je takođe podržano. Kako *Roslyn* radi sa objektnim modelom, moguće je da se prati stanje i tok rada napisanog alata kako bi moglo da se vidi na koji način *Roslyn* rukuje generisanim kodom [1].

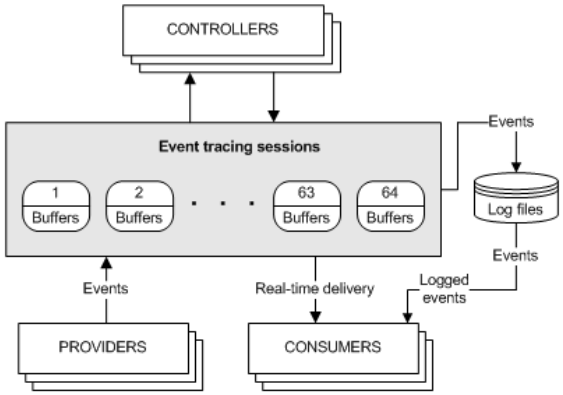
3. EVENT TRACE FOR WINDOWS

Event tracing for windows predstavlja efikasan mehanizam za vođenje evidencije na nivou kernela, koji korisniku omogućava da zapisuje željene informacije o događajima koji su se dogodili u toku rada *Windows* operativnog sistema ili korisnički definisanog programa koji podržava ovakav način vođenja evidencije. Zapisane

informacije se kasnije mogu koristiti kako bi se otkrili problemi u kodu kao i da bi se izmerile performanse izvršavanja pisanog koda.
Event Tracing API podaljen je u tri odvojene komponente:

- kontroleri – zaduženi za startovanje i stopiranje procesa evidentiranja
- provajderi – zaduženi za objavljivanje događaja
- korisnici – alati koji koriste zapisane podatke

Način na koji radi *ETW* prikazan je na slici 3.1.



Slika 3.1 *Evetn Tracing for Windows* mehanizam

3.1. Kontroleri

Kontroleri su programi koje:

- definišu veličinu i lokaciju datoteka u kojoj će se vršiti perzistencija evidentiranih informacija,
- započinju i prekidaju sesije vođenja evidencije,
- omogućavaju provajderima da vrše evidenciju,
- rukuju veličinom *buffer*-a i
- vode statistiku koja uključuje broj korišćenih *buffer*-a, kao i izgubljenih događaja i *buffer*-a

3.2. Provajderi

Provajderi su programi koje su instrumentalisani. Kako bi provajder mogao da vodi evidenciju događaja, prvo je potrebno da se registruje na operativni sistem kako bi kontroler mogao da mu dozvoli vođenje evidencije. Od provajdera zavisi kako će se ponašati u slučaju da mu je dozvoljeno vođenje evidencije i u slučaju da nije.

3.3. Korisnici

Korisnici predstavljaju aplikacije koje se pretplaćuju na željene događaje iz odgovarajućih sesija. Ukoliko je neki od provajdera aktivan, korisnik informacije o njegovim događajima može dobiti iz predefinisanih datoteka ili direktno iz sesije u realnom vremenu.

4. OPIS REŠENJA PROBLEMA

Rešenje problema je implementirano kao dodatak Visual Studio integrisanom razvojnom okruženju koje je moguće u projekat uključiti kao *NuGet* paket.
Pomoćne klase koje su implementirane radi lakšeg implementiranja potrebne logike su:

- *MethodInfo* – klasa koja modeluje pojedinačnu metodu u klasi
- *ClassInfo* – klasa koja modeluje pojedinačnu klasu u dokumentu
- *DocumentInfo* – klasa koja modeluje pojedinčni dokument u sklopu odabranog projekta
- *ProjecInfoHelper* – klasa koja modeluje odabrani projekat

4.1. *MethodInfo*

Klasa *MethodInfo* se nalazi na najnižoj poziciji u hijerarhiji pomoćnih klasa i ona služi za perzistenciju informacija o detektovanim metodama u sklopu izvornog koda, koje su od značaja.

4.2. *ClassInfo*

Klasa *ClassInfo* služi za enkapsuliranje informacije, o detektovanim klasama u sklopu odabranog projekta za instrumentalizaciju, koje će kasnije biti od značaja.

4.3. *DocumentInfo*

U hijerarhiji, klasa *DocumentInfo* se nalazi odmah ispod *ProjectInfoHelper* klase. Ova klasa modeluje čitave .cs datoteke koje su uključene u posmatrani projekat nezavisno od toga da li u sklopu projekta postoji neka hijerarhija foldera.

4.4. *ProjectInfoHelper*

ProjectInfoHelper klasa se nalazi na vrhu opisane hijerarhije, pored zaduženja za skladištenje korisnih informacija o projektu od interesa, zadužena je i za čuvanje napravljenih izmena na originalnim projektom.

4.5. Klase koje vrše generaciju koda

U okviru implementacije rešenja problema, generisanje koda je podeljeno na tri klase. Takva podela je napravljena pošto je bilo potrebno na tri različita mesta vršiti generisanje. Tako klase koje su zadužene za generisanje koda su:

- *InstrumentationGenerator* – klasa zadužena za modifikovanje postojećeg koda,
- *EventSourceEventGenerator* – klasa zadužema za generisanje potrebnih događaja i
- *EventSourceWriterGenerator* – klasa zadužena za generisanje metoda koje upisuju događaje

4.5.1. *InstrumentationGenerator*

InstrumentationGenerator klasa, kao što je rečeno, ima ulogu da modifikuje postojeći kod. Rešenje problema je zamišljeno tako da su za svako metodu od interesa bitna četiri događaja: početak i kraj izvršavanja metode, slučaj kada je došlo do problema prilikom izvršavanja kao i beleženje povratne vrednosti metode. Kako bi se jasno mogli razdvojiti događaji koji obeležavaju kraj metode, događaji koji vraćaju informaciju o povratnoj vrednosti metode kao i događaji u slučaju neuspešnog izvršavanja

metode, čitava metoda je podeljena u tri posebne celine. Tako postoje *try*, *catch* i *finally* segmenti svake instrumentalizovane metode.

4.5.2. *EventSourceEventGenerator*

U skladu sa predefinisanim šablonom koji određuje kako generisana klasa treba da bude formatirana, njeno generisanje u okviru *EventSourceEventGenerator* klase podeljeno je u četiri dela:

1. Generisanje polja potrebnih za konfigurisanje događaja
2. Generisanje samih događaja
3. Generisanje jedinstvenih identifikatora događaja
4. Generisanje jedinstvenih identifikatora metoda na osnovu kojih će događaji biti generisani

Kako je bilo potrebno podržati mogućnost pokretanja procesa instrumentalizacije i u slučaju da je određeni deo koda već instrumentalisan, bilo je potrebno implementirati funkcionalnost kojom će se izvršiti modifikacija postojeće klase koja sadrži generisane događaje. Modifikacije koje je potrebno izvršiti su dodavanje skupa događaja zasnovanih na obrađenim metodama, kao i njihovih identifikatora.

Pošto je produkt rada *EventSourceEventGenerator* klase novi dokument koji u tom trenutku nije ni na koji način vezan za izabrani projekat, bilo je potrebno implementirati logiku za njegovo čuvanje i uključivanje u projekat.

4.5.3 *EventSourceWriterGenerator*

Pošto postoji gotovo neograničeni broj kombinacija skupova parametara, metode koje su zadužene za zapisivanje događaja koje se nalaze u okviru klase *EventSource*, koju nasleđuje generisana klasa zadužena za zapis događaja, nisu dovoljne kako bi se pokrio svaki pojedinačan slučaj. To je uslovilo potrebu za generisanjem posebne klase koja sadrži korisnički definisane metode za zapis događaja na osnovu metode iz izvornog koda nad kojima se vrši proces instrumentalizacije.

Upravo za to služi *EventSourceWriterGenerator* klasa.

Šablon koji klasa treba da poštuje manje je kompleksan u odnosu na šablon *EventSourceEventGenerator* klase tako da se njeno generisanje može podeliti u dva dela. Prvi deo je zadužen za generisanje logike koja odlučuje u kom trenutku će se zapisivati tačno vreme dešavanja događaja, dok se u drugom delu generišu potrebne metode koje će vršiti zapisivanje događaja.

Kako nije redak slučaj da različite metode mogu imati jednake skupove parametara po pitanju broja i tipova, potrebno je obezbediti samo jednu generisanu metodu za zapisivanje događaja. To je neophodno kako se ne bi generisao problematičan kod koji će u daljem radu onemogućiti proces kompajliranja.

5. ZAKLJUČAK

Kada je reč o prednostima i manama implementiranog rešenja, potrebno je adresirati je jedno i drugo. Jedna od najvećih prednosti i glavnih razloga automatizacije postupka instrumentalizacije jeste vreme koje je potrebno da bi se postupak izvršio. Za ručnu instrumentalizaciju projekta prosečne kompleksnosti i veličine vreme koje je potrebno izdvojiti može biti reda veličine nekoliko dana ili čak nedelja. Kada je automatizovani način u pitanju, uz dobro definisani šablon koje bi generisane klase poštovale, postupak instrumentalizacije bi mogao da se izvrši za vreme reda veličine nekoliko minuta. Međutim, uprkos velikoj brzini generisanja koda, problem predstavlja njegova količina.

Za projekte prosečne veličine broj generisanih linija lako može dostići i nekoliko hiljada, što može biti veoma teško za održavanje i proveru validnosti. Osim količine koda, problem predstavljaju različite navike pisanja koda koje inženjeri imaju. Kako bi se pokrili svi mogući slučajevi, potrebno je da se rešenje testira ne velikom broju projekata, ali ni tada se ne bi moglo reći sa sigurnošću da su svi slučajevi pokriveni. Dodatno, implementirano rešenje predstavlja alat koji je zadužen samo za generisanje koda, ali ne i za njegovo validiranje, tako da ukoliko sam izvorni kod nije napisan dobro, instrumentalizacije neće biti generisana na onako kako je predviđeno.

Kako opisano rešenje predstavlja početak istraživanja na polju vođenja evidencije pomoću *ETW* mehanizma i generisanja koda pomoću *Roslyn* platforme za pisanje kompajlera, ima dosta mesta za optimizaciju i napredak u razvoju alata. Jedan od budućih zadataka će biti da se pokriju sve situacije koje nisu pokrivene inicijalnim rešenjem. Primer takvih situacija su metode koje su pitane u programskom jeziku *C++* i *C* (*unsafe*), kao i privatne metode. Zatim bi od velike koristi bilo implementirati korisnički definisan alat za obradu i prikazivanje događaja koji su se desili, kako bi se mogle dobiti funkcionalnosti koje su potrebne korisniku a nisu implementirane u sklopu već postojećih alata.

6. LITERATURA

[1] drdobbs.com, 2011, Commenting, Testing, and Instrumenting Code, [online] dostupno na: http://www.drdobbs.com/architecture-and-design/commenting-testing-and-instrumenting-cod/229300224 [posećeno 07.09.2018]

[2] docs.microsoft.com, *Code Generation and T4 Text Templates*, [online] dostupno na: https://docs.microsoft.com/en-us/visualstudio/modeling/code-generation-and-t4-text-templates?view=vs-2017 [posećeno 10.09.2018]

[3] Alfred, V. Aho 2007, Compilers: Principles, Techniques, and Tools, 2nd edn, AddisonWesley, Boston

Kratka biografija:

Luka Marić rođen je u Novom Sadu 1994. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Automatizacija instrumentalizacije koda pomoću *Roslyn* kod generatora odbranio je 2018.god.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.31
DOI: https://doi.org/10.24867/01BE46Vukeljic

VREMENSKO KAŠNJENJE SIGNALA U DISTRIBUTIVNOJ MREŽI

TIME DELAY OF SIGNAL IN THE DISTRIBUTION NETWORK

Bojan Vukeljić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu se razmatra putanja signala prilikom komandovanja telemetrisanom tačkom i izvršavanja VVO CL funkcije. Razmatraju se tipovi vremenskog kašnjenja kao i samo vremensko kašnjenje koje se javlja prilikom komandovanja i izvršavanja VVO CL funkcije. Takođe, razmatra se i kako i na koji način vremensko kašnjenje utiče na proračune, nadzor sistema i komandovanje. Proračun vremenskog kašnjenja je izvršen na osnovu prikupljenih vremenskih kašnjenja.*

Ključne reči: *vremensko kašnjenje, VVO CL, regulacioni resursi.*

Abstract – *In this paper the signal trajectory during the command of a telemetric point and VVO CL function execution was discussed. The types of time delay as well as the time delay that occurs during the command and execution of the VVO CL function are considered. Also taken into account is how and in what way the time delay affect computation and system oversight and commanding does. The calculation of time delays were conducted based on the acquired time delays.*

Ključne reči: *time delay, VVO CL, regulation resources.*

1. UVOD

U distributivnim mrežama (DM) tajming u komunikaciji i razmeni podataka je veoma kritičan. Zaista, ovo je najvažnija razlika između prenosa podataka u DM i prenosa podataka u većini drugih mreža. Neke tipove podataka između električnih uređaja korisne su samo u unapred definisanom vremenskom periodu. Ako kašnjenje u komunikaciji tj. u prenosu podataka prelazi vremenski period, podaci ne služe njenoj nameni. Informacije dobijene iz polja (status rasklopne opreme, merenje napona, struje itd.) mogu biti nekonzistentne kao posledica ne samo loših merenja već i vremenskog kašnjenje prilikom prenosa podataka (koji u međuvremenu mogu da se promene). Posledice kašnjenja mogu biti loši rezultati proračuna, ali i šteta na opremi, nepouzdan rad relejne zaštite (npr. kasni otvaranje prekidač, prekidač mora odmah otvoriti ako napon ili struja u zaštitnom uređaju premaši definisani prag [1]).

Distribution Management System (DMS) sistemi za kontrolu i upravljanje DM-om u realnom vremenu se primenjuju kao distribuirani sistemi, gde su petlje kontrole i upravljanja zatvorene preko komunikacione mreže.

Komunikacijsku mrežu dele razni procesori, od kojih svaki ima različite prioritete i računarska opterećenja.

U takvim sistemima neizbežno je da u komunikaciji između različitih jedinica postoji vremensko kašnjenje [2]. Vremensko kašnjenje u komunikaciji se javlja uglavnom prilikom prenosa signala putem optičkih vlakana, putem mikrotalasa, konverzije analognog signala u digitalni (A/D konverzija) i obrnuto, kao i prilikom vremenske sinhronizacije signala pomoću GPS-a [3].

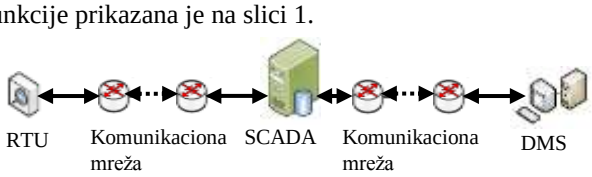
Pored vremenskog kašnjenja u komunikacijama postojaće i računarsko vremensko kašnjenje (vreme kašnjenja servisa i protokola). Dužina vremenskog kašnjenja je teška tačno predvideti i zato se najčešće modeluje kao slučajna veličina [2]. Konačno, uticaj kašnjenja može da se razmatra kao uticaj na kontrolu upravljanja i uticaj na performanse sistema [3].

Tipovi vremenskog kašnjenja, kao i analiza njegovog uticaja na upravljanje DM-om predstavlja osnovne teme ovog rada. Pritom, obrađuju se dva aspekta, prenosa podataka iz polja do kontrolnog centra, a to su kašnjenje podataka i tipovi kašnjenja koji se pojavljuju prilikom prenosa podataka. Oba aspekta obrađuju se na dva konkretna primera, prvi primer komandovanje tačkom u polju iz DMS-a i drugi izvršavanje VVO Closed Loop (VVO CL) funkcije.

Nakon uvoda, u drugoj glavi je ukratko prikazana putanja signala iz polja do DMS-a kao i tipovi kašnjenja prilikom tog prenosa podataka. U trećoj glavi je definisan i objašnjen matematički model sistema koji se razmatra. U četvrtoj glavi su obrađeni konkretni primeri kašnjenja komandovanja telemetrisanim tačkama i izvršavanja VVO CL funkcije. Zaključak je prikazan u petoj glavi, a u šestoj glavi je referentno navedena korišćena literatura.

2. KAŠNJENJE

Kašnjenje se definiše kao vreme proteklo od promene stanja do vremena kada je neka aplikacija na to stanje reagovala. Svaka aplikacija ima svoj sopstveni zahtev za kašnjenje, što zavisi od odgovora sa kojim sistem radi [4,5]. Primer putanje signala prilikom komandovanja telemetrisanom tačkom u polju i izvršavanjem VVO CL funkcije prikazana je na slici 1.



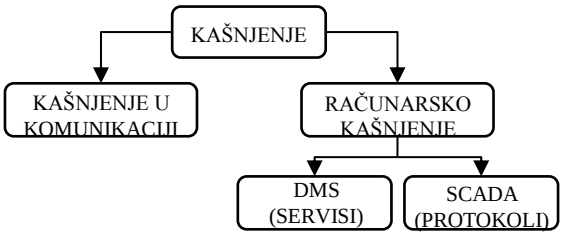
Slika 1 – Putanja signala prilikom komandovanja i izvršavanja VVO CL funkcije

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Goran Švenda, red.prof.

Neminovno je da će prilikom prenosa podataka postojati određeno vremensko kašnjenje, čiji uzrok zavisi od mnogo faktora.

Tipovi kašnjenja koje se razmatraju u ovom radu su:

- Kašnjenje u komunikaciji (telekomunikacije);
- Računarsko kašnjenje (servisi/protokoli);
- Mehaničko kašnjenje.



Slika 2 – Stablo vremenskog kašnjenja signala

Na slici 2 prikazana je podela vremenskog kašnjenja signala, kašnjenje signala je podeljeno na kašnjenje u komunikaciji i računarsko kašnjenje. Mehaničko vremensko kašnjenje se ne odnosi na kašnjenje signala nego je to jednostavno kašnjenje mehaničkih sklopova distributivne opreme u ovom slučaju teretnog menjača RTr-a i prekidača.

2.1 Kašnjenje u komunikaciji

U komunikacionoj mreži, pretpostavlja se da se podaci prenose u formi paketa. Vremensko kašnjenje u komunikaciji obuhvata nekoliko različitih kašnjenja koja se događaju u komunikacionim sistemima. Obično ova kašnjenja su: serijska kašnjenja, „međupaketska" serijska kašnjenja, routing kašnjenja, kašnjenje u propagaciji [3].

Zavisno od medijuma koji se koristi za prenos podataka, postojaće različita vremenska kašnjenja. U tabeli 1 prikazana su vremenska kašnjenja u zavisnosti od medijuma koji se koristi za prenos podataka.

2.2 Računarsko kašnjenje

Računarsko kašnjenje je teško predvideti i najčešće se modeluje kao slučajna varijabla. U ovom radu računarsko kašnjenje je predstavljeno kao kašnjenje servisa i protokola, odnosno potrebno vreme procesiranja servisa i protokola.

U tabeli 2 data su vremena procesiranja komandi za DMS i SCADA-u.

2.3 Mehaničko kašnjenje

Prilikom slanja komande iz DMS-a u cilju promene statusa prekidača, ili promena pozicije teretnog menjača na RTr, pored vremena koje je potrebno da se komanda prenese do uređaja potrebno je određeno vreme i za mehaniku prekidača i teretnog menjača.

Pod mehanikom prekidača smatra se otvaranje i zatvaranje kontakata, gašenje luka i vreme za prekidanje strujnog kruga. Dok pod mehanikom RTr smatra se pronalaženje finalne pozicije i vreme između dve uzastopne promene pozicije teretnog menjača.

Dakle, pri analizi vremenskog kašnjenja, u toku izvršavanja VVO CL funkcije i komandovanje telemetrisanom tačkom važno je uzeti u obzir i mehaničko kašnjenje. U tabelama 3 i 4 data su vremena vezana za mehaniku prekidača i RTr-a.

Tabela 1 – Vremenska kašnjenja u zavisnosti od medijuma za prenos podataka [1,3,5,6]

Tip medijuma za prenos podataka	Vremensko kašnjenje [ms]
Broadband Power Line (BPL)	~ 24
General Packet Radio Service (GPRS)	~ 100
WiMAX	~ 10 ÷ 100
Optička vlakna	~ 38 ÷ 100
Mikrotalasi	~ 80 ÷ 150
Power Line Carrier (PLC)	~ 150 ÷ 350

Tabela 2 – Vreme procesiranja komandi DMS-a i SCADA-e

	Akcija koja se procesira	Vreme procesiranja [ms]
DMS	Procesiranje VVO CL funkcije	~ 1989 ÷ 9765
	Procesiranje signala za komandovanja	~ 636 ÷ 937
SCADA	Procesiranje 1000 komandi jednu po jednu	~ 700
	Procesiranje 1000 komandi u istom trenutku	~ 440
	Procesiranje 1000 komandi u grupi od po 100 komandi	~ 270

Tabela 3 – Kataloški podaci prekidača ISM15_LD_1 [7]

Akcije	Vreme [ms]
Otvaranje kontakata	~ 15 ÷ 45
Zatvaranje kontakata	~ 30 ÷ 60
Gašenje luka	~ 10 ÷ 20
Prekidanje strujnog kruga	~ 25 ÷ 60

Tabela 4 – Kataloški podaci RTr-a [8]

Akcije	Vreme [s]
Pronalaženje finalne pozicije	~ 45 ÷ 75
Vreme između dve uzastopne promene pozicija	~ 1 ÷ 5

3. MATEMATIČKI MODEL

Upotrebom pametnih mreža, količina ključnih informacija koje putuju između komponenata u polju i IT infrastrukture se drastično povećava, što oslikava nužnost da distribucije počnu da investiraju u sisteme i telekomunikacije. Ova investicija je ključna da bi se pružila robusna, sigurna i efikasna infrastruktura koja je u stanju da podrži različitu opremu, protokole, procese i tokove podataka koji rezultuju iz tehnologije pametne mreže [5].

Na slici 3 prikazan je protok informacija i vremensko kašnjenje u sistemu.

Kašnjenje u sistemu se može podeliti na tri dela: računarsko kašnjenje, kašnjenje u komunikaciji i mehaničko kašnjenje. Neka t_{ukupno} označava ukupno vremensko kašnjenje u sistemu:

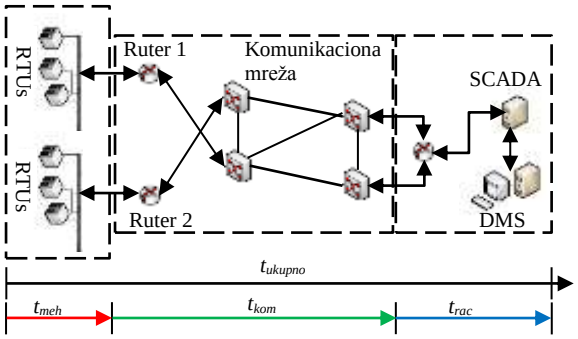
$$t_{ukupno} = t_{meh} + t_{kom} + t_{rac} \quad (1)$$

gde je:

t_{meh} – vremensko kašnjenje prilikom promene pozicije RTr-a ili prilikom manipulacije prekidača;

t_{kom} – vremensko kašnjenje u komunikaciji;

t_{rac} – računarsko vremensko kašnjenje.



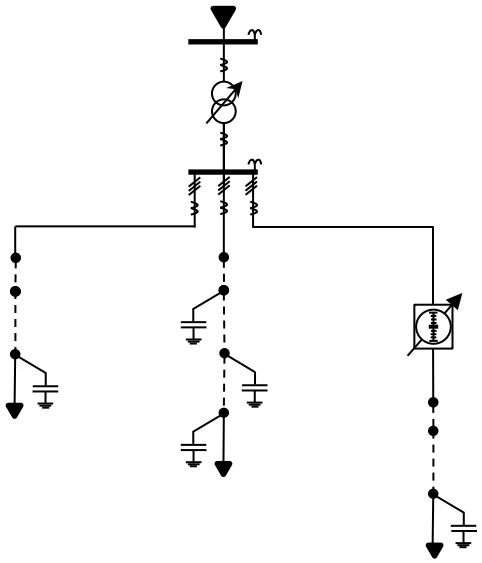
Slika 3 – Tok informacija i vremensko kašnjenje [9]

4. VERIFIKACIJA MATEMATIČKOG MODELA

Verifikacija matematičkog modela se vrši merenjem vremenskog kašnjenja izvršavanja VVO CL funkcije i komandovanja telemetrisanom tačkom.

4.1 Test DM

Test DM na kojoj se izvršava proračun VVO CL funkcije prikazana je na slici 4.



Slika 4 – Test DM [10]

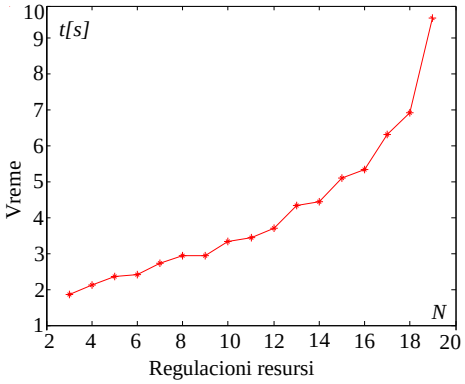
Mreža ima 2600 čvorova i sastoji se od sledećih elemenata: trofazni RTr sa regulacijom napona pod opterećenjem, 3 SN trofazna izvoda, nadzemnih i kablovskih trofaznih vodova, 18 kondenzatorskih baterija, 1 regulator napona, i 3992 monofaznih i trofaznih potrošača.

4.2 Vremensko kašnjenje izvršavanja VVO CL funkcije

Postupak proračuna VVO CL čini 17 koraka (ciklusa) u kojima su izvršene izmene na test DM koja je objašnjena u poglavlju 4.1. Ciklusi se sastoje od izvršavanja proračuna VVO CL funkcije, provere njenih rezultata i beleženja vremena potrebnog za izvršavanje jednog ciklusa.

Izmene rađene na test DM su promena broja regulacionih resursa u mreži, što uzrokuje i promenu vremena potrebno za proračun funkcije. Vrednost vremena potrebnog za proračun VVO CL funkcije, zavisi od raspoloživih regulacionih resursa u mreži, što se može videti na slici 5. Što je broj regulacionih resursa veći i vreme za proračun je veće što dodatno utiče na ukupno vremensko kašnjenje. Ukupno vremensko kašnjenje je proračunato sabiranjem svih vremenskih kašnjenja i krajnji rezultat prikazan u tabeli 5. U tabeli se izdvajaju dva karakteristična vremenska kašnjenja za treći i trinaesti ciklus proračuna, razlog tome je što je vremensko kašnjenje u trećem ciklusu najmanje, a u trinaestom najveće.

Vremensko kašnjenje dato u tabeli 5 je kada se kao komunikacioni medijum koristi PLC tehnologija, za druge komunikacione medijume kašnjenje se dobija odabirom drugog vremenskog kašnjenja iz tabele 1.



Slika 5 – Zavisnost vremena proračuna VVO CL funkcije od broja regulacionih resursa

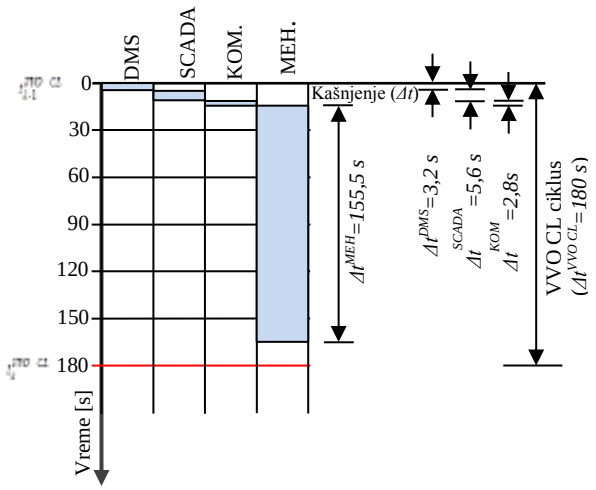
Tabela 5 – Vremensko kašnjenje izvršavanja VVO CL funkcije

Ciklusi proračuna	Broj instrukcija za optimizaciju	Vreme izvršavanja ciklusa VVO CL funkcije (max) [s]	Vreme izvršavanja ciklusa VVO CL funkcije (min) [s]
1.	10	34,06	20,71
2.	9	27,92	15,90
3.	8	25,10	14,42
4.	8	99,10	58,42
5.	7	96,64	57,29
6.	6	93,81	55,8
7.	6	93,40	55,39
8.	5	90,39	53,71
9.	5	90,19	53,52
10.	4	87,72	52,38
11.	3	84,96	50,95
12.	2	82,64	49,97
13.	4	167,09	97,75
14.	3	84,46	50,46
15.	2	82,02	49,35
16.	2	81,82	49,15
17.	2	81,56	48,89

Prilikom poređenja vremenskog kašnjenja u trećem i trinaestom ciklusu može se uvideti da vremensko kašnjenje u trećem ciklusu ima 8 instrukcija za izvršavanja, dok u trinaestom ciklusu ima 4 instrukcije, pa bi očekivano bilo da je vremensko kašnjenje veće u slučaju kada imamo 8 instrukcija za izvršavanje.

Međutim, veoma važnu ulogu u ukupnom vremenskom kašnjenju pored broja instrukcija ima i tip instrukcije koja je predložena za izvršavanje.

Vremensko kašnjenje u trinaestom ciklusu je najveće iz razloga jer postoje dve instrukcije za promenu pozicije RTr-a, to znatno utiče na mehaničko vremensko kašnjenje što se može videti i na slici 6.



Slika 6 – Ukupno vremensko kašnjenje za trinaesti ciklus proračuna VVO CL funkcije [11]

Na osnovu dobijenih rezultata kašnjenja, vidimo da za izvršavanje jednog ciklusa VVO CL funkcije potrebo 167,1 s, tako da vreme ponovnog pokretanja VVO CL funkcije može da se odredi na osnovu toga (npr. možemo da postavimo na 168 s). Međutim, zbog bezbednosnih razloga podešavamo ga na 180 s i kažemo da za jedan ciklus izvršavanja VVO CL funkcije potrebno je 180 s.

4.3 Vremensko kašnjenje komandovanja telemetrisanom tačkom

Postupak proračuna vremenskog kašnjenja komandovanja telemetrisanom tačkom čini 4 koraka (ciklusa) u kojima su izvršene promene položaja kontakata prekidača. Ciklusi se sastoje od komandovanja operatera prekidačem iz kontrolnog centra, povratne informacije da je prekidač izvršio akciju i beleženja vremena potrebnog za izvršavanje komande.

Vrednost ukupnog vremenskog kašnjenja izvršavanja komande dobijena je sabiranjem vremenskih kašnjenja, dobijeni rezultati prikazani su u tabeli 6. Vremensko kašnjenje dato u tabeli 6 je kada se kao komunikacioni medijum koristi WiMAX, za druge komunikacione medijume kašnjenje se dobija odabirom drugog vremenskog kašnjenja iz tabele 1.

Tabela 6 – Vremensko kašnjenje komandovanja telemetrisanom tačkom

Ciklus proračuna	Vreme izvršavanja komande (max) [s]	Vreme izvršavanja komande (min) [s]
1.	2,66	1,55
2.	2,45	1,34
3.	2,43	1,31
4.	2,38	1,26

5. ZAKLJUČAK

Iz gore navedenog jasno je da će buduće pametne mreže zahtevati obimnu informacionu i komunikacionu infrastrukturu koja podržava efikasnu i sigurnu proizvodnju, prenos i distribucija električne energije. Međutim, moguća kašnjenja u komunikaciji i prenosu podataka mogu da utiču na performanse upravljačkih sistema, što dovodi do gubitaka snage i eventualno oštećenja opreme.

U ovom radu je pre svega postavljen problem vremenskog kašnjenja koje se javlja prilikom prenosa informacija. Definisani su osnovni pojmovi, predstavljena su izmerena vremenska kašnjenja i način na koji kašnjenje utiče na sistem. Na osnovu svega toga kada se predoči DM sa svojim određenim karakteristikama može se dati okvirno vremensko kašnjenje koje će postojati prilikom prenosa informacija.

6. LITERATURA

1. H.Ali, D.Dasgupta: Effects of Time Delays in the Electric Power Grid. Jonathan Butts; Sujeet Shenoi. 6th International Conference on Critical Infrastructure Protection (ICCIP), Washington, DC, United States, Mar 2012, pp.139-154.
2. J.Nilsson and B.Bernhardsson: Analysis of Real-Time Control Systems with Time Delays: Department of Automatic Control Lund Institute of Technology Box 118, S-221 00 Lund, Sweden.
3. J.W.Stahlhut, T.J.Browne, G.T.Heydt, V.V.Fellow: Latency Viewed as a Stochastic Process and its Impact on Wide Area Power System Control Signals: IEEE Transactions on power system, Vol. 23, No. 1, February 2008.
4. P.Kansal, A.Bose: Bandwidth and Latency Requirements for Smart Transmission Grid Applications: IEEE Trans. on Smart Grid, Vol. 3, No. 3, September 2012, pp. 1344-1352.
5. Smart Grids on the Distribution Level – Hype or Vision?: CIRED Working Group on Smart Grids, Germany, 2013, pp. 37-57
6. B.Naduvathuparambil, M.C.Valenti, A.Feliachi: Communication Delays in Wide Area Measurement System: Proceedings of the Thirty-Fourth Southeastern Symposium on System Theory (Cat. No.02EX540), Huntsville, AL, USA, 19-19 March 2002.
7. Technical specification of ISM15_LD_1, ISM15_LD_6: Vacuum circuit breakers LD and Shell type, Table 23.
8. G.Chao: Voltage Control in Distribution Networks using On-Load Tap Changer Transformers, PhD, University of Bath, UK, 2013.
9. F.Zhang, Y.Sun, L.Cheng, X. Li, J.H. Chow, W. Zhao: Measurement and Modeling of Delays in Wide-Area Closed-Loop Control Systems: IEEE Transactions on Power Systems, China, September 2014, pp. 2426-2433.
10. N.Vučićević: Spremnost modela elektrodistributivne mreže primenom naprednih DMS energetskih funkcija, magistarska teza, Fakultet tehničkih nauka, Novi Sad April 2016.
11. A.M.Stanković, V.Švenda, A.T.Sarić: Hybrid Power System State Estimation with Irregular Sampling: IEEE Power & Energy Society General Meeting, Chicago, IL, USA, 16-20 July 2017.

Kratka biografija:

Bojan Vukeljić rođen je u Sanskom Mostu 1993. godine. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Elektroenergetski sistemi odbranio je 2016. god. Iste godine upisao se na master studije na istom smeru.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.85
DOI: https://doi.org/10.24867/01BE47Radosavljevic

PRIMENA UČENJA USLOVLJAVANJEM NA OBUČAVANJE AGENTA ZA IGRANJE VIDEO IGRE ROAD FIGHTER

TRAINING A REINFORCEMENT LEARNING AGENT TO PLAY ROAD FIGHTER

Ivan Radosavljević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu dat je primer primene učenja uslovljavanjem agenta za igranje video igre Road Fighter. Agent je implementiran kao Duboka Q-mreža i obučavan metodom Dubokog Q-Učenja. Obučavanje je vršeno samo na osnovu slika ekrana video igre. Rezultati postignuti nakon 10000 epizoda obučavanja bili su slični rezultatima postignutim u srodnim radovima pod uslovom sličnih hardverskih ograničenja.*

Ključne reči: *Učenje uslovaljavanjem, neuronske mreže, duboka q mreža, duboko q učenje, video igra*

Abstract – *This paper presents the application of reinforcement learning for the purpose of training an agent capable of playing the video game Road Fighter. The agent is implemented as a Deep Q-Network and is trained via the Deep Q-Learning algorithm. The only data used for training the agent were screenshots of the game. After 10000 episodes of training the performance of the agent was comparable to that achieved in related approaches under similar hardware constraints.*

Keywords: *Reinforcement learning, neural networks, deep q network, deep q learning, video game*

1. UVOD

Metode mašinskog učenja koje spadaju u metode nadgledanog i nenadgledanog učenja su se pokazale kao veoma dobre u rešavanju problema klasifikacije i otkrivanja šablona u podacima, međutim uspešnost ovih metoda najčešće zavisi od toga koliko su dobro pripremljeni podaci nad kojima se primenjuju. Priprema podataka je najčešće mukotrpan proces koji vrše ljudski eksperti i podrazumeva uklanjanje šuma, rešavanje problema nedostajućih vrednosti, u slučaju nadgledanog učenja obeležavanje podataka, itd. Za razliku od ovih metoda, učenje uslovljavanjem, se ne oslanja na prethodno pripremljene podatke, umesto toga podaci za obučavanje se sakupljaju direktno u toku procesa obučavanja agenta za rešavanje problema u nekom okruženju. Ovo čini učenje uslovljavanjem veoma primamljivim za rešavanje problema za koje nije praktično vršiti ručnu pripremu podataka. Jedan takav problem predstavlja obučavanje agenata za igranje igara, jer dovoljno složene igre mogu generisati velike količine podataka koje je u većini slučajevima nemoguće ručno pripremiti.

Iz tog razloga je agent koji će biti predstavljen u ovom radu realizovan metodom učenja uslovljavanjem, predstavljenoj u radu [1], prilagođenoj za obučavanje agenata za igranje video igara. Agent je obučavan za igranje video igre Road Fighter koja predstavlja jednostavnu simulaciju trke izdatu za Nintendo Entertainment System konzolu. Cilj ovog rada bio je da se metoda iz rada [1] isproba na prosečnom hardveru i nešto složenijoj video igri u odnosu na video igre za Atari konzolu. Po uzoru na rad [1] obučavanje je vršeno samo na osnovu slike ekrana video igre, bez dodatnih podataka. bilo internih ili dostavljenih od strane ljudskog eksperta. Glavna razlika između rešenja predstavljenog u ovom radu i rešenja iz rada [1] jeste to što se u ovom rešenju za obučavanje agenta koristilo dve neuronske mreže, a ne jedna. Takođe za razliku od rada [1] u kojem je dobavljanje slika ekrana video igre i slanje akcija za upravljanje igrom vršeno direktno preko Atari emulatora, a u ovom radu je to izvedeno indirektno preko odvojene softverske komponente napravljene sepcijalno za tu svrhu. Rezultati postignuti ovakvim pristupm bili su u okvirima očekivanih rezultata za primenu metode iz rada [1] pod ograničenjem upotrebe hardvera prosečnih računskih mogućnosti. Nakon 10000 epizoda obučavanja agent je bio sposoban da u proseku u video igri bez greške upravlja automobilom dve sekunde, dok ljudski igrač u proseku uspeva da bez greške igra video igru oko 14 sekundi.

Rad je podeljen u pet sekcija. U narednoj sekciji dat je kratak pregled postojećih rešenja. U trećoj sekciji predstavljena je primenjena metodologija. Četvrta sekcija opisuje postupak evaluacije rešenja i sadrži komentare na postignute rezultate. Konačno, u petoj sekciji, izložen je zaključak ovog rada.

2. POSTOJEĆA REŠENJA

Veliki broj radova na temu učenja uslovljavanjem tiče se primene ove metode mašinskog učenja na obučavanje agenata za igranje igara poput šaha [2], igre go [3] i sl., međutim tek u poslednje vreme su počeli da se pojavljuju radovi na temu obučavanja agenata za igranje video igara. Dva reprezentativna rada na ovu temu su [1, 4].

U radu [4] autori su predstavili implementaciju agenta za igranje video igre Dota 2. Agent je implementiran kao duboka neuronska mreža koja na osnovu podataka sa bot API-ja video igre donose odluke o akcijama koje treba da sprovede u igri. Podaci sa bot API-ja predstavljaju veoma složenu reprezentaciju stanja igre, sačinjenu od 20000 atributa. Agent se obučava igrajući partije igre protiv samog sebe. Nakon perioda od nekoliko meseci obuke

---

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kovačević, vanredni prof.

agent je došao u stanje da može da pobedi igrače početnike, što je uzimajući u obzir složenost video igre za koju se obučava veliki uspeh.

Autori rada [1] predstavili su novu metdu za obučavanje agenata za video igre koja se ne oslanja na složene reprezentacije stanja. Metoda predstavljena u njihovom radu podrazumeva samo upotrebu slika ekrana video igre za obučavanje i eksploataciju agenta. Agent je implementiran kao duboka konvolutivna neuronska mreža koja na ulaz prima slike ekrana video igre a na izlaz daje aproksimiranu vrednost optimalne funkcije vrednosti stanja. Autori ovu mrežu nazivaju duboka Q mreža (eng. *Deep Q-Network*, DQN), dok metod obučavanje ove mreže nazivaju duboko q učenje (eng. *Deep Q Learning*). Primenom ove metode pri obučavanju agenata za igranje Atari video igara autori su uspeli da za neke video igre čak nadmaše performanse ljudskih eksperata. Ipak treba imati u vidu da su igre nad kojima je ova metoda primenjena uspešno dosta jednostavnije od Dota 2 video igre.

3. METODOLOGIJA

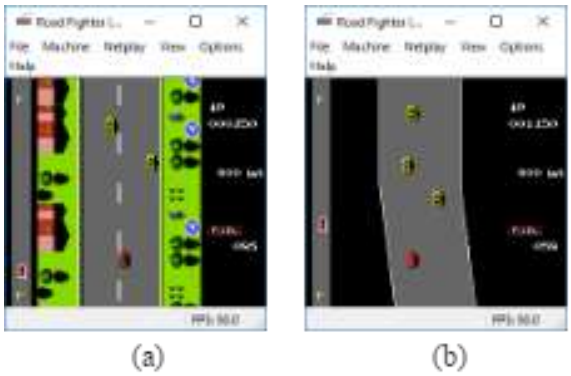
U ovoj sekciji biće predstavljeni detalji vezani za metodologiju primenjenu prilikom obučavanja agenta za igranje video igre Road Fighter. Prvo će biti opisana softverska komponenta napravljena za interakciju sa video igrom. Nakon toga će biti predstavljlena arhitektura i implementacija neuronske mreže upotrebljene za izvedbu agenta. Na samom kraju biće izneti detalji vezani za obuku agenta.

3.1. Upravljanje video igrom i dobavljanje podataka

Video igra Road Fighter izdata je 1994. godine kao video igra za Nintendo Entertainment System (NES) konzolu, i kao takva nije izvršiva na modernom računarskom hardveru. Kako bi se igra mogla pokrenuti neophodno je obezbediti emulator za NES konzolu. Za potrebe ovog rada, usled jednostavnosti upotrebe i preciznosti simulacije NES hardvera, odabran je Nestopia emulator [5]. Mana ovog emulatora ogleda se u nepostojanju ugrađenog interfejsa za povezivanje drugih programa sa emulatorom u cilju upravljanja emulatorom i dobavljanja podataka iz emulatora. Iz ovog razloga bilo je neophodno napraviti softversku komponentu koja će moći da obavlja ove zadatke. Ova komponenta implementirana je u programskom jeziku Python i sačinjena je od jedne klase pod nazivom GameControl. Klasa GameControl implementirana je uz oslonac na Windows API [6] i predstavlja apstrakciju za pristup prozoru video igre, odnosno emulatora. Upotreba Windows API-ja omogućuje pristup i upravljanje svim internim stanjima bilo kog prozora pod Windows operativnim sistemom. Ovo znači da se dobavljanje slike sadržaja prozora može vršiti veoma efikasno kopiranjem podataka iz grafičkog bafera prozora. Takođe slanje komandi u vidu pritisaka tastera tastature ili klikova miša se svodi na na pozive funkcija Windows API-ja za slanje poruka o događajima izazvanim upotrebom ulaznih uređaja. Za potrebe upravljanja emulatorom klasa GameControl implementira nekoliko metoda od kojih su najbitnije:

- *grab_screen()* - vrši dobavljanje slike sadržaja prozora emulatora, odnosno slike ekrana video igre.
- *get_current_state()* - vrši dobavljanje podataka o trenutnom stanju igre. Podaci se dobavljaju kao torka (slika ekrana, procenat pređene staze, preostala količina goriva, brzina). Pri tome slika ekrana predstavlja binarnu sliku samo dela ekrana igre na kojem su vidljivi samo staza za trkanje i automobili na stazi. Ova slika se dobija transformacijom slike dobijene pozivom metode grab_screen() u binarnu sliku i isecanjem dela koji sadrži stazu za trkanje. Poslednja tri podatka iz torke se dobavljaju izdvajanjem i interpretacijom delova slike dobijene metodom grab_screen() na kojima su prikazani ovi podaci.
- *execute_action(action_code, delay)* - izvršava zadatu akciju i blokira preuzimanje podataka ili slanje akcija u trajanju specificiranom argumentom delay. Blokiranje se vrši kako bi emulator imao vremena da obradi primljenu akciju. Dozvoljene akcije su skretanje u levo i skretanje u desno.
- *restart_game()* - vrši ponovno pokretanje igre sa početka staze.

Efikasno dobavljanje podataka o gorivu, brzini i poziciji sa slike ekrana video igre bilo je moguće zahvaljujući jednostavnom grafičkom dizajnu delova ekrana koji služe za prikaz tih podataka. Međutim izdvajanje dela ekrana na kojem se odvija sama trka i njegovo pretvaranje u binarnu sliku zahtevalo je izmenu same video igre u vidu uklanjanja mnogih ukrasnih elemenata koji se prikazuju na tom delu ekrana. Prikaz ekrana video igre pre i posle uklanjanja ovih elemenata dat je na slici 1.



Slika 1. *a) Izgled video igre pre uklanjanja ukrasnih elemenata, b) Izgled video igre nakon uklanjanja ukrasnih elemeanta*

3.2. Arhitektura i implementacija neuronske mreže

Cilj ovog rada bilo je obučavanje agenta koji bi na osnovu tri uzastopne slike ekrana video igre donosio odluke o akcijama koje treba da sprovede. Upotreba tri slike bila je neophodna kako bi se mogli izdvojiti podaci o brzinama kretanja objekata na ekranu. Iz ovog razloga odabrana je arhitektura koja predstavlja konvolutivnu neuronsku mrežu sačinjenu od dva 3d konvolutivna sloja i tri potpuno povezana sloja. Ulaz u neuronsku mrežu sačinjen je od tri binarne slike ekrana video igre pripremljena na način opisan u sekciji 3.1. Izlaz iz ove neuronske mreže je vektor koji sadrži dva elementa čije vrednosti ukazuju na povoljnost izbora akcija za skretanje levo ili desno. Prvi sloj u neuronskoj mreži je 3d konvolutivni sloj sa filtorm dimenzija 5x5x2 koji se pomera za po dva piksela po x i y osi i za po jedan po z osi. Izlaz iz ovog sloja predstavlja 16 feature mapa sačinjenih od po dve slike dimenzija duplo manjih od ulaznih slika. Nad ovim feature mapama

se potom primenjuje 3d batch normalizacija čiji se rezultat prosleđuje ReLU aktivacionoj funkciji. Sloj za 3d batch normalizaciju uveden je kako bi se postigla stabilnija konvergencija tokom obučavanja [7]. Ovako transformisane feature mape se zatim prosleđuju drugom 3d konvolutivnom sloju čiji filter ima dimenzije 3x3x2. Korak ovog filtera isti je kao i korak filtera u prvom konvolutivnom sloju.

Izlaz iz ovog sadrži 32 feature mape koje se sastoje od po jedne slike duplo manjih dimenzija od ulaznih feature mapa. Takođe i nad ovim izlazom se vrši 3d batch normalizacija i primenjuje ReLU aktivaciona funkcija.

Nakon toga dobijene feature mape se pretvaraju u vektor od 37696 elemenata koji se prosleđuje prvom potpuno povezanom sloju. Izlaz iz ovog sloja je vektor od 32 elementa. Ovaj vektor se prosleđujeu narednom potpuno povezanom sloju koji kao izlaz daje vektor od 16 elemanata koji se potom prosleđuje konačnom potpuno povezanom sloju. Izlaz iz konačnog sloja predstavljaju vrednosti povoljnosti akcija.

Odabir predstavljene arhitekture posledica je testiranja različitih arhitektura za rešavanje problema predstavljenog u ovom radu. Testiranjem je ustanovljeno da jednostavnije arhitekture nisu davale dovoljno dobre rezultate, dok su složenije arhitekture zahtevale dosta duže vreme obučavanja. Kako je ova arhitektura pokazala zadovoljavajući odnos između performansi i brzine obučavanja bila je izabrana za potrebe rešavanja ovog problema.

Predstavljena neuronska mreža implementirana je kao Python klasa koja nasleđuje klasu Modul iz biblioteke PyTorch [8], modula torch.nn. Ova klasa redefiniše metodu *forward()* čija namena je specificiranje slojeva i veza između slojeva neuronske mreže kao i toka podataka kroz neuronsku mrežu. Povratna vrednost ove metode predstavlja izlaz iz neuronske mreže.

3.3. Obučavanje agenta

Proces obučavanja upotrebljen u ovom radu predstavlja kombinaciju metoda obučavanja predstavljenih u radovima [1, 9]. Prvi korak u obučavanju jeste povezivanje sa emulatorom preko instance GameControl klase. Nakon toga vrši se instanciranje dve neuronske mreže π_{net} i V_{net}, od kojih prva služi za odabir optimalne politike (eng. *policy network*) dok druga služi za vršenje procene vrednosti stanja (eng. *value network*). Težine obe mreže se inicijalizuju istim nasumično izabranim vrednostima. Potom se inicijalizuje i replay memorija kapaciteta od 100000 stanja. Sledeći korak je pokretanje glavne petlje obučavanja. Ova petlja se izvršava dok broj iteracija petlje ne dostigne zadati broj epizoda. Na početku svake epizode vrši se inicijalizacija reda F_{queue} koji može da skladišti do tri poslednje slike ekrana video igre. Nakon ovoga vrši se restartovanje igre i otpočinjanje petlje epizode. U svakoj iteraciji petlje epizode vrši se izbor akcije. Akcija se može izabrati nasumično ili na osnovu izlaza neuronske mreže π_{net}. Da li se akcija bira nasumično ili pomoću π_{net} određeno je ϵ-pohlepnom politikom. Verovatnoća nasumičnog izbora akcije računata je po jednačini 1 [10].

$$\epsilon = \epsilon_{end} + (\epsilon_{start} - \epsilon_{end})e^{-\frac{n}{\epsilon_{decay}}} \quad (1)$$

Za parametre ϵ_{start}, ϵ_{end} i ϵ_{decay} uzete su vrednosti 0.95, 0.25, 120000. Izbor ovih vrednosti ustanovljen je empirijski. Nakon izbora akcije dobavlja se novo stanje video igre i pravi kopija reda F_{queue}, F'_{queue}. Ova kopija služi za čuvanje slika prethodnog stanja video igre. Zatim se vrši provera kraja epizode i popunjavanje replay memorije. Smatra se da je do kraja epizode došlo ukoliko je automobil usporio. Ukoliko je došlo do kraja epizode u replay memoriju se upisuje torka (F'_{queue}, $akcija$, $None$, 1), gde je $None$ oznaka za terminalno stanje, a 1 visina nagrade, i potom se izlazi iz petlje epizode. U suprotnom u replay memoriju se upisuje torka (F'_{queue}, $akcija$, F_{queue}, 1) i petlja epizode se nastavlja. Po završetku ažuriranja replay memorije i nakon svakih pet epizoda 100 puta se vrši ažuriranje težina mreže π_{net}. Ažuriranje težina π_{net} se vrši tako što se prvo iz replay memorije nasumično izabere 128 uzoraka za koje π_{net} izračuna predikcije vrednosti V. Potom se uz pomoć V_{net} računaju očekivane nagrade V' za izabrane uzorke po jednačini 2. Gde v predstavlja vrednost trenutnog stanja, γ faktor za koji se umanjuje vrednost posmatranog stanja s_t.

$$V' = v + \gamma \max(V_{net}(s_t)) \quad (2)$$

Vrednosti V i V' se zatim prosleđuju Huberovoj funkciji greške, jednačine 3 i 4 [11], koja se koristi u procesu minimizacije razlike između vrednosti V i V'.

$$E(V', V) = \frac{1}{n}\sum_{i}^{n} z_i \quad (3)$$

$$z_i = \begin{cases} 0.5(V' - V)^2, & |V' - V| < 1 \\ |V' - V| - 0.5, & |V' - V| \geq 1 \end{cases} \quad (4)$$

Dobijena vrednost greške se potom propagira niz π_{net} nakon čega se pokreće Adam algoritam za ažuriranje težina [12]. Težine neuronske mreže V_{net} ažuriraju se samo ukoliko je od njihovog poslednjeg ažuriranja proteklo 100 epizoda. Pri tome ažuriranje V_{net} podrazumeva kopiranje parametara mreže π_{net} u V_{net}. Nakon opcionog ažuriranja težina otpočinje se sledeća iteracija glavne petlje obučavanja.

4. REZULTATI

Evaluacija performansi ovog rešenja vršena je poređenjem rezultata postignutih eksploatacijom obučenog agetna i rezultata postignutih nasumičnim izborom akcija. Rezultati su izraženi u prosečnom broju akcija izvršenih do terminalnog stanja u 100 epizoda. Prikupljanje rezultata postignutih od strane agenta vršeno je u tri ključna trenutka obučavanja, nakon 0 epizoda, nakon 3000 epizoda i nakon 10000 epizoda obučavanja. Tabela 1 prikazuje ostvarene performanse agenta u navedenim trenucima.

Tabela 1. *Performanse agenta u ključnim trenucima obučavanja*

Broj epizoda obučavanja	Prosečan broj akcija izvršenih do terminalnog stanja
0	74.87
3000	116.77
10000	123.94
Nasumičan izbor akcija	109.55

Rezultat dobijen nakon 0 epizoda obučavanja posledica je nasumično inicijalizovanih težina neuronske mreže koje uzrokuju da agent uvek bira istu akciju što u proseku nakon 74 akcije dovodi do silaženja automobila sa puta i završetka epizode. Iako rezultat ostvaren nasumičnim izborom deluje prilično blizu rezultatu ostvarenom nakon 3000 epizoda obučavanja, prava razlika između ova dva rezultata ogleda se u tome što agent nakon 3000 epizoda počinje da ciljano izbegava silaženje sa puta. Nakon 10000 epizoda agent počinje da uči obilaženje drugih automobila. Međutim kako je obilaženje složenije od izbegavanja ivice puta agent često ne uspeva da obiđe druge automobile pa je iz tog razloga rezultat približan rezultatu ostvarenom nakon 3000 epizoda obučavanja. Iako ostvareni rezultati deluju ispod prosečnih, oni potvrđuju da se metode predstavljene u radovima [1, 9] mogu primeniti na obučavanje agenata za igranje video igara složenijih od Atari igara, ali i da se za postizanje dovoljlno dobrih performansi mora potrošiti puno vremena na obučavanje.

5. ZAKLJUČAK

U ovom radu predstavljena je primena učenja uslovljavanjem za obučavanje agenta za igranje video igre Road Fighter. Obučavanje agenta je vršeno, po uzoru na rad [1], samo uz pomoć slika ekrana video igre. Metoda primenjena za obučavanje agenta predstavlja modifikovane metode iz radova [1, 9]. Rezultati postignuti ovom metodom, iako lošiji od rezultata koje ostvaruje prosečan ljudski igrač, pokazuju da je motda primenljiva ali da zahteva velike količine hardverskih resursa i vremena za obučavanje.

U cilju daljeg poboljšanja performansi agenta moguće je izvršiti izmene postojećeg rešenja poput: povećanja kapaciteta replay memorije, povećanja broja slika koje se koriste za opis stanja video igre, upotrebe ϵ politike takve da se tokom obučavanja daje prednost nasumičnim akcijama na neistraženim delovima igre i upotreba ekspertskog znanja u ranijim fazama obučavanja.

6. LITERATURA

[1] V. Mnih *et al.*, "Playing Atari with Deep Reinforcement Learning," p. 9.

[2] D. Silver *et al.*, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," *arXiv:1712.01815 [cs]*, Dec. 2017.

[3] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[4] "OpenAI Five," *OpenAI Blog*, Jun-2018. Dostupno na: https://blog.openai.com/openai-five/.

[5] "Nestopia - NES/Famicom Emulator." Dostupno na: http://nestopia.sourceforge.net/.

[6] M. Satran, "Programming reference for Windows API.". Dostupno na: https://docs.microsoft.com/en-us/windows/desktop/api/.

[7] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167 [cs]*, Feb. 2015.

[8] "PyTorch dokumentacija." Dostupno na: https://pytorch.org/docs/stable/index.html.

[9] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," p. 7.

[10] A. Paszke, "Reinforcement Learning (DQN) Tutorial — PyTorch Tutorials." Dostupno na: https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

[11] P. J. Huber, "Robust Estimation of a Location Parameter," *Ann. Math. Statist.*, vol. 35, no. 1, pp. 73–101, Mar. 1964.

[12] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Dec. 2014.

Kratka biografija:

Ivan Radosavljević je rođen 20. 11. 1993. godine u Loznici, Republika Srbija. Osnovnu školu „Vuk Karadžić" završio 2008. godine. Iste godine se upisao u Srednju tehničku školu u Loznici, smer elektrotehničar računara. Srednju školu završio 2012. godine. Godinu dana kasnije upisuje se na Fakultet tehničkih nauka u Novom Sadu, odsek Elektrotehnika i računarstvo, smer Softversko inženjerstvo i informacione tehnologije u izdvojenom odeljenju u Loznici. Godine 2017. završio osnovne akademske sutdije i upisao master akademske studije na Fakultetu tehničkih nauka u Novom Sadu, odsek Elektrotehnika i računarstvo, smer Softversko inženjerstvo i informacione tehnologije, modul Inteligentni sistemi. Položio sve ispite predviđene planom i programom.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.85
DOI: https://doi.org/10.24867/01BE48Vidovic

PREDIKCIJA CENE NEKRETNINA NA OSNOVU PODATAKA IZ OGLASA

REAL ESTATE PRICE PREDICTION USING ADVERTISEMENT DATA

Mladen Vidović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu je predstavljen model za predikciju cene nekretnina na osnovu podataka iz oglasa. Iz oglasa, preuzetih sa veb stranice za oglašavanje, su izdvojene tehničke specifikacije nekretnine, slike nekretnine i, ukoliko su dostupne, geografske koordinate nekretnine. Geografske koordinate su upotrebljene za formiranje ocene kvaliteta lokacije. Slike su upotrebljene za obučavanje neuronske mreže za detekciju značajnih objekata na slikama. Formirana su tri skupa podataka za obučavanje prediktivnih modela. Prvi skup sadrži samo tehničke specifikacije nekretnina, drugi skup ima dodatu ocenu lokacije, a treći skup ima i ocenu lokacije i detektovane objekte na slikama iz oglasa. Za svaki skup je obučeno nekoliko regresionih modela za predikciju cene i njihove performanse su poređene. Performanse ovih prediktivnih modela, izražene kao R^2, su poređene. Najbolje performanse je imao GBT (Gradient Boosted Trees) model na skupu sa slikama i ocenom lokacije sa ostvarenom R^2 vrednošću od 0.856.*

Ključne reči: *Istraživanje i analiza podataka, mašinsko učenje, regresija, neuronske mreže.*

Abstract – *This paper presents a real estate price prediction model using advertisement data. Real estate technical specifications, images and, if available, geographical coordinates are extracted from advertisements, acquired from a real estate advertising website. The coordinates are used to form location ratings. The images are used to train a neural network for the detection of a real estate's equipment on images.Three separate datasets for training the price prediction models were formed. The first dataset contains only technical specifications of the real estates, the second dataset also contains location ratings, while the third dataset contains location ratings and objects detected on images. These datasets were used to train different regression models for predicting real estate prices. The performances of the models, represented by their achieved R^2 scores, were compared in order to establish which model had the best performances.. The best performing model was the GBT (Gradient Boosted Trees) model on the dataset with location ratings and detected objects, with an achieved R^2 score of 0.856.*

Keywords: *Data analysis, machine learning, regression analysis, neural networks.*

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kovačević, vanr.prof.

1. UVOD

Kupovina nekretnina za većinu ljudi predstavlja veliku i retku investiciju. Zbog toga je bitno pri prodaji odrediti adekvatnu cenu za nekretninu. U opštem slučaju, cenu nekretnine određuje njen vlasnik, ili agent za prodaju nekretnina. Njihova procena je zasnovana na ličnom iskustvu i interesima, i zbog toga nije objektivna i egzaktna. U ovom radu je predstavljen model za objektivno određivanje cene nekretnine na osnovu njenih karakteristika, kao i kvaliteta lokacije na kojoj se nalazi. Ulaz u model su podaci o nekretninama i njihovim lokacijama ekstraktovani iz njihovih oglasa. Oglasi su preuzeti sa veb stranice za oglašavanje. Model za predikciju cene je realizovan kao regresioni model, pri čemu je za regresiju primenjeno nekoliko različitih algoritama. Model je takođe primenjen na različite skupove podataka, sa različitim atributima, u cilju ispitivanja uticaja podataka o lokaciji, kao i slika iz oglasa, na konačne performanse modela.

Rad se sastoji iz 5 sekcija. U narednoj sekciji je dat pregled literature relevantne za ovaj rad. U trećoj sekciji su prikazani koraci realizacije rešenja predstavljenog u ovom radu. Četvrta sekcija se bavi evaluacijom rešenja i poređenjem performansi različitih prediktivnih modela. Peta sekcija predstavlja sumarizaciju rada i pravce daljeg razvoja.

2. PREGLED SLIČNIH RADOVA

Postoji veliki broj radova na temu predikcije cene nekretnine. Većina ovih radova koristi samo struktuirane tehničke specifikacije nekretnine za obučavanje prediktivnog modela. Radovi koji u skup podataka uključuju lokaciju nekretnine, ili vizualne podatke, odnosno slike, se obično fokusiraju na samo jedan od ovih skupova podataka. U radu [1], autori predstavljaju model za predikciju cene kuće na osnovu tehničke specifikacije i slika. Nad slikama su vršili ekstrakciju svojstava koristeći SURF (*Speeded Up Robust Features*) i ekstraktovana svojstva zajedno sa tehničkim karakteristikama nekretnine prosledili neuronskoj mreži za predikciju cene. Poredili su performanse mreže sa ranijim rešenjem za isti skup podataka [2] i pokazali da je dodavanje slika u obučavajući skup znatno poboljšalo performanse prediktivnog modela. Autori rada [3] su pokazali da lokacija nekretnine, relativno u odnosu na neke značajne centre, poput centra grada ili centara zaposlenja, utiče na cenu nekretnine. Zaključili su da je cena nekretnine inverzno proporcionalna njenoj udaljenosti od ovih značajnih centara. Autori rada [4] pokušavaju da pokažu koliko vrede objekti u okolini nekretnine, odnosno koliko utiču na njenu cenu. Koristeći

GIS (*Geographic Information System*), prikupili su podatke o zelenim površinama, poput parkova u radijusu od 500 metara oko nekretnine i uključili ih u skup podataka. Pokazali su da svaki tip ovakve zone pozitivno utiče na cenu nekretnine, ali da preveliki broj ovakvih zona može da dovede do prezasićenja, i negativno da utiče na cenu nekretnine. Takođe su potvrdili da objekti koji zagađuju okolinu, poput industrijskih zona negativno utiču na cenu nekretnine.

3. METODOLOGIJA

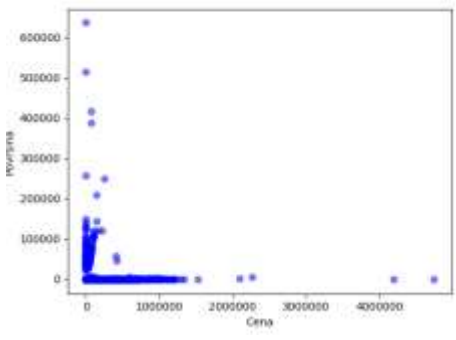
U ovoj sekciji su prikazani koraci implementacije rešenja predstavljenog u ovom radu. Dati su načini dobavljanja podataka, formiranja skupova podataka, kao i formiranje prediktivnih modela i njihova optimizacija.

3.1. Dobavljanje oglasa

Oglasi, iz kojih su izdvojeni podaci o nekretninama, su dobavljeni sa veb stranice za oglašavanje nekretnina, nekretnine.rs [5], upotrebom Scrapy biblioteke za Python programski jezik [6]. Pri prikupljanju, izdvojeni su samo oglasi za prodaju stanova u Novom Sadu i Beogradu. Dobavljeni oglasi su zapisani u datoteku u JSON formatu. Ukupno je dobavljleno 95942 oglasa i 662396 URL-ova slika, koji su kasnije upotrebljeni za dobavljanje slika.

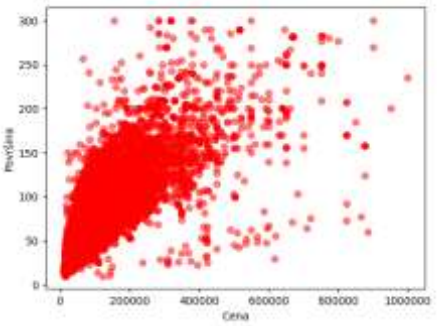
3.1. Pretprocesiranje podataka

Eksplorativnom analizom podataka uočeno je da neki oglasi nemaju navedenu cenu. Oni su neupotrebljivi za formiranje sistema, pa su uklonjeni. U nekim oglasima nije naveden sprat na kojem se stan nalazi. U ranijim radovima je ustanovljeno da sprat ima značajan uticaj na cenu stana [7], pa su oglasi bez navedenog sprata uklonjeni iz skupa podataka. Podaci o datumu objavljivanja i ažuriranja oglasa, kao i tipu oglašavača su uklonjeni iz skupa podataka, jer su vezani za sam oglas, a ne za nekretninu, pa ne bi trebalo da utiču na njenu cenu. Svaki oglas takođe sadrži komentar u obliku slobodnog teksta, koji u opštem slučaju sadrži tekstualni opis nekretnine i kontakt podatke oglašavača. Ekstrakcija podataka iz ovih komentara bi zahtevala primenu *text mining*-a, što izlazi van opsega ovog rada, tako da su slobodni komentari uklonjeni iz skupa podataka. Sledeći korak je bilo određivanje i uklanjanje šuma. Za to su pre svega posmatrane površine i cene nekretnina u oglasima, prikazane na grafiku 1.



Grafik 1. *Podaci pre uklanjanja šuma*

Uočeno je da postoje oglasi za stanove sa navedenom površinom u redu nekoliko hiljada kvadratnih metara, što nije realistično. Takođe ima veliki broj stanova sa cenom preko 1000000 evra, kao i veliki broj stanova sa veoma malom cenom, ispod 1000 evra. Čak 92 stana su imali navedenu cenu od 1 evro. Kao gornja granica za cenu, postavljeno je 1000000 evra, a donja granica je bila 10000 evra. Za površinu, gornja granica je 300m^2 a donja granica 10m^2. Na grafiku 2 se mogu videti podaci nakon uklanjanja šuma.



Grafik 2. *Podaci nakon uklanjanja šuma*

Nakon pretprocesiranja, u skupu podataka je preostalo 61942 oglasa.

3.2 Određivanje kvaliteta lokacije

Oglasi koji su sadržali geografske koordinate nekretnine su upotrebljeni za formiranje novog skupa podataka, od 18374 oglasa, za koje je određen kvalitet lokacije. Za to su, koristeći Overpass API [8] za OpenStreetMap [9], dobavljeni svi javni objekti, poput škola, restorana ili autobuskih stanica, u radijusu od 150m oko nekretnina. Nad tim detektovanim objektima je zatim primenjen *K-means* algoritam za klasterovanje, u cilju formiranja grupa nekretnina koje su slične po kvalitetu lokacije. Klasterovanje je vršeno sa različitim brojem klastera, nakon čega su dodeljeni klasteri spojeni sa cenama nekretnina, i poređene su srednje i prosečne vrednosti cena nekretnina u svakom klasteru. Ispostavilo se da se uvek formira jedan klaster sa nekretninama u čijim okolinama je detektovan mali broj objekata i taj klaster uvek ima najmanje srednje i prosečne vrednosti cene. Preostali klasteri se formiraju na osnovu tipa detektovanih objekata u okolinama nekretnina. Za konačan model odabran model sa 2 klastera, zato što model ima najveću razliku u cenama između klastera. Modeli sa većim brojem klastera imaju mali broj elemenata u nekim klasterima, što bi moglo negativno da utiče na performanse prediktivnih modela.

3.3 Detekcija objekata na slikama

Za detekciju tehničke opremljenosti nekretnina na slikama, obučena je neuronska mreža. Za potrebe formiranja obučavajućeg skupa, ručno je anotirano 7998 slika, koristeći labelImg alat za anotiranje [10], od kojih je 25% izdvojeno kao test skup za evaluiranje mreže. Na slikama su anotirani objekti za koje je smatrano da utiču na cenu nekretnine, poput kućnih aparata ili nameštaja. Za detekciju objekata, preuzeta je unapred kreirana neuronska mreža, koja je već trenirana na COCO [11] skupu podataka, koji sadrži slike svakodnevnih objekata, slične onima koje je bilo potrebno detektovati. Arhitektura preuzete mreže je Faster R-CNN ResNet101 [12]. Mreža je dodatno obučena na ručno anotiranim slikama, i promenjena da kao izlaz vrati detektovane objekte koji su smatrani bitnim za ovaj rad. Obučena

mreža je primenjena na 9862 oglasa koji su imali slike i geografske koordinate, čije slike nisu deo obučavajućeg skupa mreže i nisu anotirane ručno. Detektovani objekti na ovim slikama su dodati kao atributi ovih nekretnina, koje su zatim formirale zaseban skup podataka radi utvrđivanja uticaja detektovanih objekata na predikciju cene.

3.4 Regresija

U prethodnim sekcijama se može videti da su ukupno formirana 3 skupa podataka. Prvi skup podataka sadrži samo osnovne tehničke specifikacije nekretnina. Drugi skup podataka ima dodate ocene okoline nekretnina. Treći skup podata ima ocene okolina i objekte detektovane na slikama nekretnina. U ostatku rada će ovi skupovi podataka biti navedeni kao prvi, drugi i treći. Svaki od ovih skupova je podeljen na trening, validacioni i test skup, tako što je 80% elemenata odvojeno za trening skupove, a 20% za test skupove. Od trening skupova je još 20% elemenata odvojeno za validacione skupove. Za formiranje regresionih modela, odabrano je nekoliko različitih algoritama:

- linearna regresija [13], kao osnova za poređenje,
- linearna regresija sa lasso regularizacijom [14],
- linearna regresija sa ridge regularizacijom [15],
- linearna regresija sa elastic net regularizacijom [16],
- regresiono stablo [17],
- ansambl regresionih stabala u AdaBoost konfiguraciji [18],
- ansambl regresionih stabala u GBT (*Gradient Boosted Trees*) modelu [19].

Algoritmi su optimizovani tako što su iterativno primenjeni nad valdiacionim skupom, sa različitim parametrima. Parametri koji su dali najbolje performanse na validacionom skupu su uzeti kao optimalni. Mera performansi je R^2 vrednost.

U slučaju linearne regresije, primena regularizacija nije dovela do značajnog poboljšanja performansi, pa su ovi modeli odbačeni iz daljeg razmatranja.

4. REZULTATI

U ovoj sekciji je dat pregled ostvarenih rezultata neuronske mreže za detekciju objekata na slikama, kao i regresionih modela za predikciju cene nekretnine.

4.1 Evaluacija neuronske mreže

Neuronska mreža je evaluirana na prethodno izdvojenom test skupu slika, i ostvarila je MAP (*Mean Average Precision*) od 0.594. Najbolje performanse je imala za objekte koji su se često pojavljivali na slikama, poput radijatora centralnog grejanja, sa AP (*Average Precision*) od 0.834, ili klima uređaja sa AP od 0.842.

Pokazalo se da mreža najviše greši na retko zastupljenim objektima, poput interfona ili termoakumulacionih peći, kao i da pravi greške na objektima koji previše liče na neke druge objekte, poput umivaonika koji liče na bide. Ovo bi se moglo rešiti anotiranjem većeg broja slika za obučavajući skup podataka.

4.2 Evaluacija regresionih modela

Evaluacija regresionih modela je izvršena na test skupovima podataka, nad kojima nije vršena nikakva optimizacija modela. Drugi skup podataka je evaluiran i pre i posle dodavanja podataka o kvalitetu lokacije, a treći skup je evaluiran i pre i posle dodavanja podataka o kvalitetu lokacije i objekata detektovanim na slikama, sa ciljem ispitivanja uticaja ovih podataka na performanse prediktivnih modela. Mera za evaluaciju performansi je ostvarena R^2 vrednost. Rezultati evaluacije su dati u tabeli 1.

Tabela 1. *Ostvarene R^2 vrednosti na test skupovima*

Skup podataka	Linearna regresija	Regresiono stablo	AdaBoost	GBT
Prvi	0.69	0.71	0.741	0.759
Drugi, bez klastera	0.708	0.679	0.691	0.754
Drugi sa klasterima	0.716	0.762	0.742	0.814
Treći bez klastera i slika	0.73	0.625	0.691	0.751
Treći sa klasterima	0.734	0.735	0.775	0.851
Treći sa klasterima i slikama	0.739	0.754	0.814	0.856

Na osnovu rezultata se može zaključiti da dodavanje podataka o kvalitetu lokacije znatno poboljšava performanse modela, dok dodavanje objekata detektovanim na slikama ima manji uticaj na performanse prediktivnih modela.

Ovo može da znači da ovi podaci ne utiču u tolikoj meri na cenu, ili da su mogli biti integrisani u skup podataka na bolji način. Model sa najboljim performansama je bio GBT. Najbolju R^2 vrednost je imao za treći skup podataka, nakon dodavanja svih dostupnih atributa. U tabeli 2 je dat pregled grešaka GBT modela, sa srednjim vrednostima cena i površina primera.

Tabela 2. *Greške GBT modela na trećem skupu*

Greška modela (u evrima)	Broj primera	Srednja vrednost površine	Srednja vrednost cene
do 10000	1345	50	56650
10000-50000	539	60	95000
50000-100000	64	118	205505
100000-200000	23	145	370000

Može se videti da greška modela raste sa cenom i površinom nekretnina. To može biti posledica malog broja primera sa velikim cenama i površinama u obučavajućem skupu podataka, kao i činjenicom da na njihovu cenu utiču faktori koje nije moguće predstaviti na stranici za oglašavanje, poput većeg broja pomoćnih objekata i terasa, obezbeđenja zgrade i slično.

5. ZAKLJUČAK

U ovom radu je predstavljen model za predikciju adekvatne cene nekretnine na osnovu njenih tehničkih specifikacija, i lokacije. Podaci o nekretninama su dobavljeni iz oglasa, preuzetih sa veb stranice za oglašavanje nekretnina. Kvalitet lokacije nekretnine je određen na osnovu značajnih objekata u njenoj okolini. Skup atributa nekretnine je proširen značajnim objektima unutar nekretnine, detektovanim na slikama iz oglasa.

Za detekciju objekata na slikama obučena je neuronska mreža. Mreža je ostvarila MAP od 0.594. Analiza pogrešno detektovanih objekata je pokazala da mreža najviše greši kod tipova objekata koji su retko zastupljeni i koji previše liče na druge objekte.

Za predikciju cene nekretnine obučeno je nekoliko regresionih modela, i njihove performanse su poređene. Najbolje rezultate je dao GBT(*Gradient Boosted Trees*) model, sa ostvarenom R^2 vrednošću od 0.856, nakon uključivanja podataka o lokaciji i objekata detektovanim na slikama u skup atributa. Analiza grešaka modela je pokazala da model najviše greši kod primera sa velikom površinom, cenom, novije gradnje, kao i primera koji imaju nepotpune ili neispravno unesene tehničke specifikacije.

Performanse mreže za detekciju objekata na slikama bi se mogle poboljšati dobavljanjem većeg broja obeleženih slika za obučavanje. Detekcija značajnih objekata u većem radijusu oko okoline, zajedno sa podatkom o broju detektovanih objekata istog tipa bi mogao dati bolju reprezentaciju kvaliteta lokacije.

Neispravno popunjene tabele sa tehničkim specifikacijama nekretnine predstavljaju veliki izvor grešaka za prediktivni model. Pokazalo se da su često tačni podaci navedeni u formi slobodnog komentara u oglasu.

Ekstrakcija karakteristika nekretnina iz tih komentara bi omogućila dopunu nenavedenih podataka u tabelama oglasa, kao i ispravljanje pogrešno navedenih podataka.

LITERATURA

[1] Eman H Ahmed and Mohamed Moustafa. House Price Estimation from Visual and Textual Features. November 2016.

[2] Azme Bin Khamis and Nur Khalidah Khalilah Binti Kamarudin. Comparative study on estimate house price using statistical and neural network model. *International Journal of Scientific & Technology Research*, 3(12):126–131, 2014.

[3] John Ottensmann, Seth Payton, and Joyce Man. Urban Location and Housing Prices within a Hedonic Model. *Journal of Regional Analysis and Policy*, 38, January 2008.

[4] Fanhua Kong, Haiwei Yin, and Nobukazu Nakagoshi. Using GIS and landscape metrics in the hedonic price modeling of the amenity value of urban green space: A case study in Jinan City, China. *Landscape and Urban Planning*, 79(3):240–252, March 2007.

[5] Nekretnine.rs. dostupno na https://www.nekretnine.rs.

[6] Scrapy | A Fast and Powerful Scraping and Web Crawling Framework.dostupno na https://scrapy.org.

[7] Stephen Conroy, Andrew Narwold, and Jonathan Sandy. The value of a floor: valuing floor level in high-rise condominiums in san diego. *International Journal of Housing Markets and Analysis*, 6(2):197–208, 2013.

[8] overpass api. dostupno na https://wiki.openstreetmap.org/wiki/Overpass_AP.

[9] Openstreetmap. dostupno na https://www.openstreetmap.org.

[10] darrenl. Tzutalin. labelimg. git code (2015)., September 2018. dostupno na https://github.com/tzutalin/labelImg.

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Giuseppe Bonaccorso. *Machine learning algorithms*. Packt, 2017.

[14] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[15] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[16] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[17] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[18] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[19] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Kratka biografija:

Mladen Vidović rođen je u Loznici 14.03.1994. godine. Osnovne akademske studije na Fakultetu tehničkih nauka, smer Softversko inženjerstvo i informacione tehnologije završio je 2017. godine. Iste godine je upisao master akademske studije na Fakultetu tehničkih nauka, smer Softversko inženjerstvo i informacione tehnologije. Položio je sve ispite propisane planom i programom.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 624.05
DOI: https://doi.org/10.24867/01CG01Planinac

PLANIRANJE PROIZVODNJE ELEMENATA KONSTRUKCIJE MONTAŽNE HALE ANALIZOM RESURSA

PLANNING OF THE PRODUCTION OF STRUCTURAL ELEMENTS OF THE PREFABRICATED HALL BY RESOURCE ANALYSIS

Jelena Planinac Jokić, Jasmina Dražić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAĐEVINARSTVO

Kratak sadržaj – *U radu su analizirani problemi vezani za proizvodnju elemenata konstrukcije montažne hale. Planiranje proizvodnje rađeno je u tri varijante s ciljem da se analizom osnovnih resursa i dinamike proizvodnje predloži najpovoljnije rešenje.*

Ključne reči: *montažna hala, elementi, proizvodnja (prefabrikacija), planiranje, resursi*

Abstract – *This thesis analyses the problems related to the production of elements of the construction of the prefabricated hall. Production planning is done in three variants with the aim of proposing the most favorable solution by analyzing the basic resources and the dynamics of production.*

Keywords: *prefabricated hall, elements, production (prefabrication), planning, resources*

1. UVOD

Montažni način građenja podrazumeva skup tehničkih, tehnoloških i organizacionih mera i metoda, koje se primenjuju sa ciljem da se proizvodnja ubrza i olakša i da se izvrši racionalizacija utroška resursa za tu proizvodnju [1].

U ovom radu je detaljno planirana proizvodnja elemenata konstrukcije montažne hale: temeljnih čašica, temeljnih greda, stubova, nosača kranske staze, glavnih nosača, rožnjača i olučnih greda. Planirana je proizvodnja elemenata na osnovu analize osnovnih resursi, količina materijala (betona, oplate i armature) i radne snage. Planiranje proizvodnje rađeno je u tri varijante:

- Varijanta I – bez vremenskih i prostornih ograničenja
- Varijanta II – sa vremenskim ograničenjem i
- Varijanta III – sa prostornim ograničenjem (ograničen broj kalupa).

Pri razmatranju varijanti planiranja proizvodnje elemenata, uveden je princip paralelizacije u radu, pa je proizvodnja organizovana sa jednom i dve radne brigade. Cilj rada je bio da se za sve elemente konstrukcije montažne hale, na osnovu analize osnovnih resursa i dinamike proizvodnje, predloži najpovoljnija (optimalna) varijanta.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Jasmina Dražić, red.prof.

2. SPECIFIKACIJA ELEMENATA KONSTRUKCIJE HALE

Razmatrani objekat je skladište sa upravnom zgradom u Rumenki, čiji je investitor PEZOS EXPORT-IMPORT doo, iz Petrovaradina. Podeljen je u dve celine:

- skladište - pravougaone osnove dimenzija osovinski 20,82m x 105,02m, unutrašnje čiste visine 8,65m (od gotovog poda do donje ivice glavnog nosača) i
- upravna zgrada koja je funkcionalno povezana sa samim skladištem.

U radu su analize rađene za elemente konstrukcije tipa-jednobrodna hala. Konstrukcija objekta je montažna armiranobetonska iz programa "NOVOTEHNA" Novi Sad.

Tabela 1. *Geometrijske karakteristike temeljnih čašica*

TEMELJNE ČAŠICE			
		Temelji samci se izvode na licu mesta u betonu, MB-30 i armaturom RA 400/500 i GA 240/360, na dubini fundiranja Df=1,85m. Postoje dva tipa temeljnih čašica: TČ1 i TČ2. Čašice su kvadratnog poprečnog preseka, visine 110cm sa debljinom zida od 20-25cm.	
šifra	**geometrijske karakteristike** b/d/h [cm]	**kom.**	**zapremina** [m^3]
TČ1	110/110/110	24	0.874
TČ2	110/100/110	20	0.824

Tabela 2. *Geometrijske karakteristike temeljnih greda*

TEMELJNE GREDE			
		Temeljne grede se prozvode u armiranom betonu, MB-40, armaturom RA400/500.	
šifra	**geometrijske karakteristike** b/d/h [cm]	**kom.**	**zapremina** [m^3]
TG1	20/105/470	31	0.930
TG1a	20/105/470	1	0.875
TG2	20/105/335	4	0.650
TG2a	20/105/350	2	0.680
TG3	20/105/980	2	1.630
TG4	20/105/950	1	1.940
TG4a	20/105/950	1	1.570
TG4b	20/105/950	1	1.620

Glavna konstrukcija je prefabrikovana armirano-betonska, koju čine montažni elementi: temeljne čašice, temeljne grede, stubovi, nosači kranske staze, glavni krovni nosači, rožnjače i olučne grede [2].

Specifikacija elemenata konstrukcije prikazana je u tabelama 1, 2, 3, 4, 5, 6 i 7.

Tabela 3. *Geometrijske karakteristike stubova*

STUBOVI			
		Stubovi se izvode od betona kvaliteta MB-40 i MB-30, dimenzionisani su prema statičkom proračunu i armirani rebrastom armaturom RA400/500 i GA240/360, sa adekvatnim rešenjima detalja.	
šifra	**geometrijske karakteristike** b/d/h [cm]	**kom.**	**zapremina** [m^3]
S1	50/50/1050	12	2.760
S1a	50/50/1050	2	2.760
S1b	50/50/1117	1	2.810
S1b-I	50/50/1117	1	2.810
S1c	50/50/1117	1	2.810
S1d	50/50/1117	1	2.810
S1e	50/50/1050	2	2.760
S1f	50/50/1050	2	2.760
S2	50/50/1223	1	2.970
S2-I	50/50/1223	1	2.970
S3	40/50/1115	19	2.190
S3a	40/50/1115	1	2.190

Tabela 4. *Geometrijske karakteristike nosača kranske staze*

NOSAČI KRANSKE STAZE			
		Nosači kranske staze proizvode se od betona MB-40, sa ugrađenom armaturom RA400/500 i GA 240/360.	
šifra	**geometrijske karakteristike** b/d/h [cm]	**kom.**	**zapremina** [m^3]
NKS1	40/110/1028	16	4.420
NKS1 a	40/110/1054	2	4.480
NKS2	40/110/1174	2	5.000

Tabela 5. *Geometrijske karakteristike glavnih nosača*

GLAVNI NOSAČI			
		Glavni nosači se izrađuju od betona kvaliteta MB-50 i armature tipa RA400/500. Krovnu konstrukciju objekta hale čine: glavni krovni „dvopojasni“ nosač tipa GN1, raspona L=20,50m i nosač GN2, tipa „T“110.	
šifra	**geometrijske karakteristike** b/d/h [cm]	**kom.**	**zapremina** [m^3]
GN1	40/254/2050	9	5.460
GN2	35/110/1034	4	2.130

Tabela 6. *Geometrijske karakteristike rožnjača*

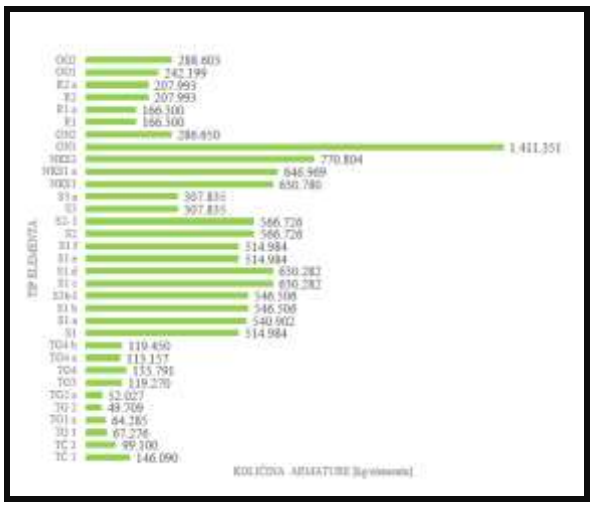
ROŽNJAČE			
		Rožnjače se proizvode od betona MB-40 i armirane su sa rebrastom armaturom RA400/500, a za uzengije se koristi armatura GA240/360.	
šifra	**geometrijske karakteristike** b/d/h [cm]	**kom.**	**zapremina** [m^3]
R1	20/68/1028	36	1.13
R1a	20/66.5/1028	9	1.13
R2	20/68/1148	4	1.27
R2a	20/66.5/1148	1	1.27

Tabela 7. *Geometrijske karakteristike olučnih greda*

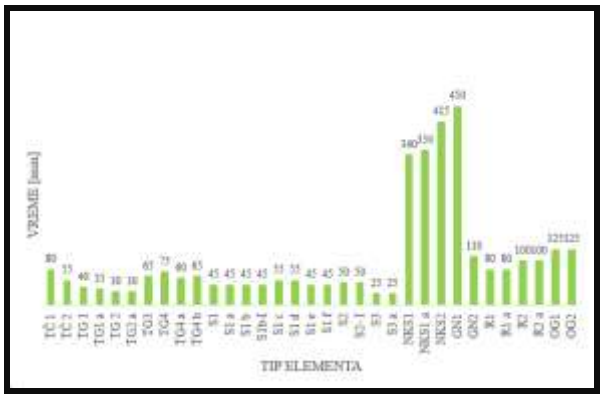
OLUČNE GREDE			
		Olučne grede su elementi "T" preseka visine h=60cm, širine flanše b=50cm i širine rebra br=15cm, Proizvodi se dva tipa, OG1 i OG2.	
šifra	**geometrijske karakteristike** b/d/h [cm]	**kom.**	**zapremina** [m^3]
OG1	50/60/1028	18	1.660
OG2	50/60/1148	2	1.850

3. OSNOVNI MATERIJAL ZA PROIZVODNJU ELEMENATA I VREME UGRAĐIVANJA

Resursi na osnovu kojih su rađene analize u ovom radu obuhvatile su osnovni materijal za proizvodnju elemenata, armaturu, oplatu (kalupe) i beton. Na osnovu internih normi preduzeća NOVOTEHNA (mereni su utrošci osnovnih materijala, kao i vreme potrebno za njihovo ugrađivanje) [2], a prema specifikaciji elemenata konstrukcije, izračunate su ukupne količine materijala za sve elemente konstrukcije i vreme potrebno za njihovo ugrađivanje. Izlazni rezultati su prikazani grafikonima na slikama 1, 2, 3, 4, 5 i 6.

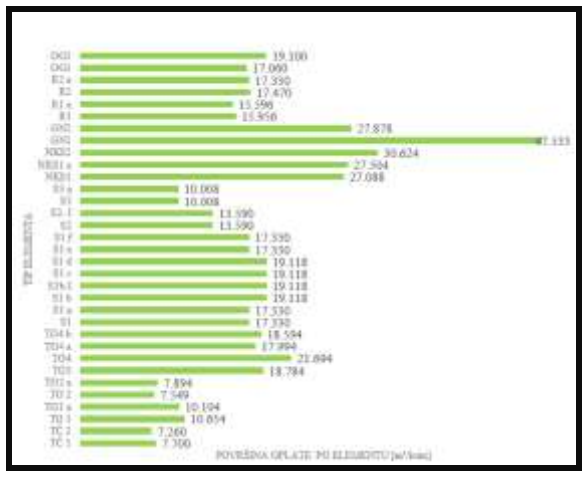


Slika 1. *Ukupan utrošak armature po tipu elementa*

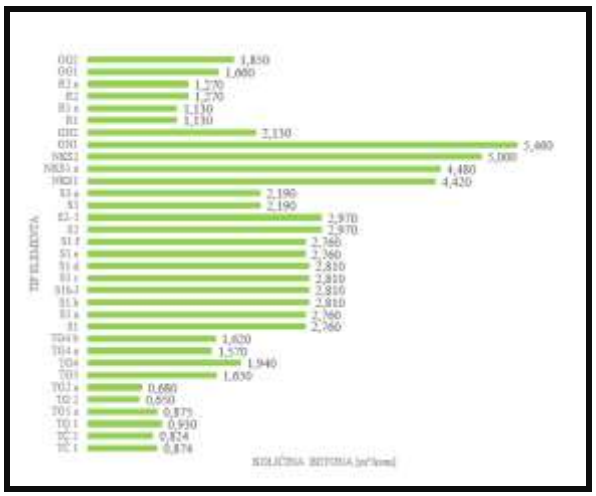


Slika 2. *Prosečno vreme za ugrađivanje armature po tipovima elementa*

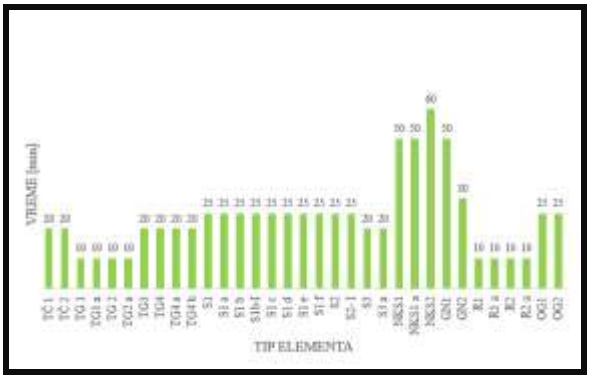
Slika 3. *Ukupan utrošak oplate po tipu elementa*



Slika 4. *Prosečno vreme postavljanja oplate po tipovima elementa*



Slika 5. *Ukupan utrošak betona po elementu*



Slika 6. *Prosečno vreme za ugrađivanje betona po tipovima elementa*

4. PLANIRANJE PROIZVODNJE

Proizvodnja prefabrikovanih elemenata organizovana je u proizvodnim pogonima, sa mogućim ograničenjima u prostornim kapacitetima, mehanizaciji i opremi. Sa druge strane, kontinuitet procesa izgradnje montažne hale (prefabrikacija-transport-montaža), i poštovanje ugovorenih rokova (zahtevi investitora), može ograničiti vreme proizvodnje. Vremensko i prostorno ograničenje pri planiranju proizvodnje zahteva paralelizaciju radova, a sigurno direktno utiču na dinamiku i osnovne resurse proizvodnje. Da bi se predložilo najpovoljnijije rešenje, planirana je proizvodnja elemenata konstrukcije montažne betonske hale u tri varijante.

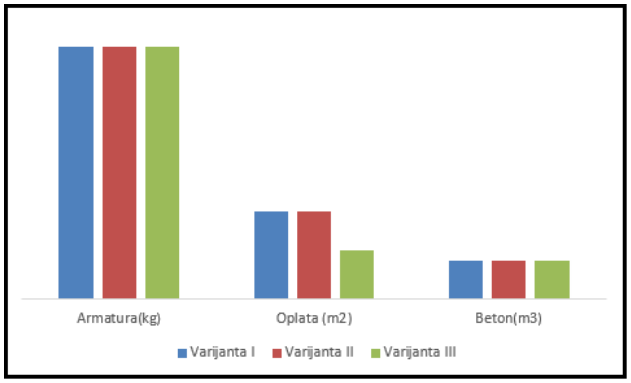
- Varijanta I - bez vremenskih i prostornih ograničenja,
- Varijanta II - sa vremenskim ograničenjem i
- Varijanta III - sa ograničenim brojem kalupa.

Poštujući redosled i trajanje operacija pri proizvodnji elemenata (priprema kalupa, armiranje, betoniranje, odležavanje i zaparivanje), formirane su radne brigade (tip brigade, broj radnika), planirano je trajanje proizvodnje pojedinačno po elementu, za sve elemente jedne vrste i sračunato je ukupno vreme potrebno za proizvodnju svih elemenata konstrukcije [3]. U prvoj varijanti proizvodnja je organizovana sa jednom kompleksnom radnom brigadom, dok je kontinuitet procesa proizvodnje i uvedena ograničenja u drugoj i trećoj varijanti, rezultirao paralelizacijom radova (rad organizovan sa dve kompleksne radne brigade).

Rezultati uporedne analize resursa, po varijantama, dati su u tabelama 8 i 9 i grafikonom na slikama 7 i 8.

Tabela 8. *Ukupna količina osnovnog materijala za sve tri varijante*

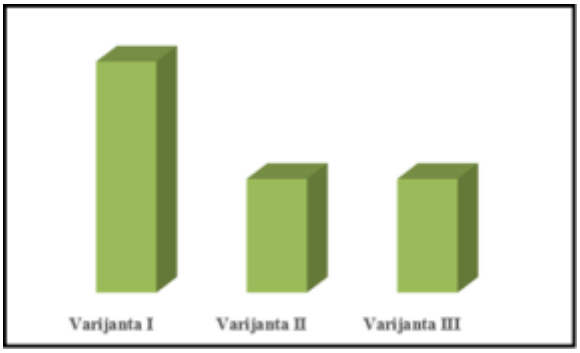
	armatura [kg]	**oplata** [m²]	**beton** [m³]
Var. I	67.747,51	1.573,28	457,951
Var. II	67.747,51	1.573,28	457,951
Var. III	67.747,51	868,29	457,951



Slika 7. *Uporedna analiza materijala za sve tri varijante*

Tabela 9. *Vreme proizvodnje i radne brigade po varijantama*

	vreme	**brigada**	**radnici**
Var. I	67	1	1A+1T+1B+1PR
Var. II	33	2	1A+1T+1B+1PR
Var. III	33	2	1A+1T+1B+1PR



Slika 8. *Uporedna analiza vremena proizvodnje za sve tri varijante*

5. ZAKLJUČAK

U radu je analiziran proces proizvodnje elemenata konstrukcije montažne hale poslovnog kompleksa u Rumenci. Za sve tri varijante izračunati su osnovni resursi u količinama materijala (oplata, armatura i beton) i radnoj snazi (broj i sastav radnih brigada). Proizvodnja u prvoj varijanti organizovana je sa jednom kompleksnom radnom brigadom, dok je u varijantama proizvodnje II i III primenjen princip paralelizacije, pa je proizvodnja organizovana sa dve kompleksne brigade.

Detaljni proračuni su pokazali da je za proizvodnju elemenata konstrukcije potrebno u:

- Varijanti I – 67 radna dana,
- Varijanti II – 33 radna dana, a u
- Varijanti III – 33 radna dana.

Na osnovu komparativne analize osnovnih resursa i vremena trajanja proizvodnje za sve tri analizirane varijante, zaključeno je da je varijanta III najpovoljnija, pa je u ovom slučaju predložena kao optimalno rešenje proizvodnje.

Detaljan pristup planiranju proizvodnje, prikazan u radu, omogućava kontrolu i praćenje svih faza prefabrikacije (kvalitet procesa i kvalitet elemenata), a izbor najpovoljnije varijante doprinosi efektima izgradnje na nivou celog objekta, vezanim za troškove i vreme izgradnje.

6. LITERATURA

[1] Trivunić M., Dražić J. – Montaža betonskih konstrukcija hala, AGM knjiga, Beograd, 2009.

[2] Interne građevinske norme preduzeća Novotehna, Novi Sad.

[3] Planinac J. - Analiza resursa za proizvodnju elemenata konstrukcije montažne hale u Rumenci, Diplomski (bachelor) rad, FTN Novi Sad, 2017.

Kratka biografija:

Jelena Planinac Jokić rođena je u Zenici 1988. god. Master rad na Fakultetu tehničkih nauka iz oblasti Građevinarstva – Tehnologija i organizacija građenja odbranila je 2018.god.

Jasmina Dražić rođena je u Novom Miloševu 1958.god. Doktorirala je na Fakultetu tehničkih nauka 2005.god., a od 2015.god. je u zvanju redovnog profesora. Oblast Zgradarstvo – građevinske konstrukcije i tehnologije.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 624.012.4.059
DOI: https://doi.org/10.24867/01CG02Cvetkovic

PROCENA STANJA I SANACIJA AB KONSTRUKCIJE ZA PRIMARNO PREČIŠĆAVANJE OTPADNIH VODA PREMA EVROPSKIM NORMAMA

ASSESSMENT AND REPAIR OF RC STRUCTURE FOR PRIMARY WASTEWATER TREATMENT ACCORDING TO EUROPEAN STANDARDS

Jelena Cvetković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAĐEVINARSTVO

Kratak sadržaj – *U radu je prikazana procena stanja armiranobetonskih objekata za primarno prečišćavanje otpadnih voda u Valjevu. Radi utvrđivanja stepena oštećenja i vrste sanacionih radova izvršen je makroskopski pregled dostupnih elemenata. Na osnovu analize uočenih oštećenja dat je opis sanacionih mera u cilju obezbeđenja stabilnosti i poboljšanja trajnosti i upotrebljivosti konstrukcije, prema Evropskim standardima EN 1504.*

Ključne reči: *procena stanja, oštećenja, sanacija, armirani beton, EN 1504, razdelnice*

Abstract – *In this paper the assessment of a RC structure for primary wastewater treatment in Valjevo is shown. In order to determine the level and cause of damages as well as the type of repairing measures a visual inspection of structural elements was done. Based on this examination detailed description of repairing measures is given with the main goal of providing stability and improving structural durability and serviceability according to European standards EN 1504.*

Keywords: *assessment, damages, repairing, reinforced concrete, EN 1504, structural joints*

1. UVOD

Rad se sastoji iz dva dela, teorijsko-istraživačkog i stručnog dela. U prvom delu rada, koji je teorijsko-istraživačkog karaktera, analizirane su vrste razdelnica u betonskim konstrukcijama i to radne, prividne i dilatacione razdelnice. Objašnjen je značaj izvođenja razdelnica, kao i primena svake od vrsta. Takođe, prikazani su načini konstruktivnog oblikovanja i postupci izvođenja razdelnica. Date su osnovne karakteristike i primeri materijala za ispunu i elemenata za obezbeđivanje vodonepropustljivosti razdelnica.
Stručni deo rada obuhvata procenu stanja i sanaciju armiranobetonske konstrukcije postrojenja za prečišćavanje otpadnih voda koje se nalazi u Valjevu. Prikazana je detaljna analiza uočenih oštećenja, na osnovu koje su dati predlozi sanacionih radova, date detaljne karakteristike sanacionih materijala, a sve to prema Evropskim standardima za sanaciju i zaštitu betonskih konstrukcija – EN 1504 [5-9]. Takođe su date i preporuke u smislu što efikasnijeg redosleda izvođenja radova.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Mirjana Malešev, red. prof.

2. RAZDELNICE

Razdelnice predstavljaju značajan deo konstrukcije, jer sa aspekta trajnosti konstrukcije omogućavaju pravilnu osnovu tehnologije građenja, a sa druge strane, sa aspekta stabilnosti, omogućavaju različito ponašanje delova konstrukcije koji se nalaze u različitim uslovima sredine ili imaju drugačije karakteristike u smislu geometrije.

2.1. Vrste razdelnica

U zavisnosti od namene i načina izvođenja postoji više tipova razdelnica, pa prema tome može se izvršiti sledeća podela:

- Radne razdelnice,
- Prividne (lažne) razdelnice,
- Dilatacione razdelnice [1].

3. RADNE RAZDELNICE

Radne razdelnice dele velike objekte na manje radne jedinice čime se postiže jednostavnija izrada oplate i armature i smanjenje količina ugrađenog betona. Za radne razdelnice karakteristična je kontinuacija armaturnih šipki kroz razdelnicu.

Izrađuju se prvenstveno iz dva razloga i to:

- Razlozi tehnološke prirode,
- Smanjenje štetnih posledica skupljanja betona [1].

3.1. Konstruktivno oblikovanje i izvođenje radnih razdelnica

Konstruktivno oblikovanje radnih razdelnica [1] zavisi od tipa konstrukcije, njenih dimenzija i prisutne armature, položaja u konstrukciji, kao i pravca pružanja radnih razdelnica. Imajući ovo u vidu, razlikuju se nekoliko tipova oblikovanja radnih razdelnica:

- Ravne razdelnice,
- Razdelnice sa ispustima (pero i žljeb),
- Maskirane razdelnice,
- Razdelnice sa zaptivnim trakama.

4. PRIVIDNE RAZDELNICE [1]

Prividne razdelnice formiraju se upravno na radne razdelnice kod pločastih elemenata velikih površina. Funkcija ovih razdelnica jeste da se smanje štetni efekti temperaturnih promena, skupljanja betona ili neravnomernog sleganja konstrukcije.

5. DILATACIONE RAZDELNICE

Dilataciona razdelnica je opšti naziv za otvor između dva konstruktivna elementa koji su izvedeni tako da omogućavaju pomeranja konstruktivnih elemenata koja prouzrokuju unutrašnje i spoljašnje sile.
U zavisnosti od načina rada, razlikuju se tri osnovna tipa i to:

- Dilatacione razdelnice za vertikalna pomeranja,
- Dilatacione razdelnice za horizontalna pomeranja,
- Dilatacione razdelnice za dinamičke uticaje [10].

5.1. Konstruktivno oblikovanje i izvođenje radnih razdelnica

Dilatacione razdelnice izvode se tako da omoguće potpuno odvajanje dva konstruktivna elementa. Armatura je u potpunosti prekinuta, a konstruktivni elementi odvojeni. Spojnica se ispunjava posebnim elastičnim materijalom koji obezbeđuje njen rad i vodonepropusnost.

6. PROCENA STANJA OBJEKTA

6.1. Opis objekta

Predmetna konstrukcija predstavlja deo postrojenja za prečišćavanje otpadnih voda koji služi za primarno prečišćavanje. Sastoji se iz dve celine: objekat AI i objekat AII. Objekat AI je podeljen u konstruktivnom smislu na rezervoarsku konstrukciju otvorenog tipa sa crpnim pumpama i njihovim nosačima i zgradu sa elektropostrojenjem i automatskim rešetkama za prečišćavanje otpadnih voda. Temeljenje rezervoara je izvršeno na koti 167.20m. Zgrada objekta AI je izgrađena u kombinovanom sistemu sa zidovima za ukrućenje, u nivoima koji predstavljaju prohodni deo objekta. Neprohodni deo objekta je tunelska konstrukcija kojom se sprovodi otpadna voda u objekat AII. Sa južne strane, paralelno sa objektom, nalazi se ulazna kineta – rampa sa potpornim zidovima koja se spušta do ulaza u podrumske prostorije. Ona je dilatacijom od 3cm odvojena od zgrade AI. Objekat AII predstavlja posebnu celinu, dilatiranu od objekta AI. Počinje nastavkom tunela kojim se sprovodi otpadna voda u rezervoarski deo konstrukcije AII – peskolov/mastolov, koji je dug 40m, a završava delom upravnim na peskolov/mastolov,koji primarno prečišćenu vodu treba da razvede na dve strane na dalje prečišćavanje. Objekat AII je armirano-betonska konstrukcija, koja predstavlja protočni rezervoar. Zbog svoje dužine podeljen je poprečno na pola dilatacijom širine 3cm. Sastoji se od spoljašnjih zidova i pregradnih elemenata. Objekat je fundiran na ploči, na koti 166.80m.

6.2. Vizuelni pregled konstrukcije

U okviru procene stanja konstrukcije [2] obavljen je vizuelni pregled svih dostupnih elemenata konstrukcije. Na otvorenom delu objekta AI uočeni su: mrežaste prsline i otpadanje zaštitnog sloja betona usled dejstva mraza, korozija armature zbog prisustva vlage i zbog nedovoljne debljine zaštitnog sloja betona, pukotine kroz ceo presek usled dugotrajnog skupljanja betona, prsline usled sedanja betona i na mestima neadekvatnog prekida betoniranja sa procurivanjem, abrazija površina zidova uz crpne pumpe, mehanička oštećenja, oštećene dilatacije i biološka korozija (Slika 1).

Slika 1. *Oštećenja zidova crpnih pumpi*

Na delu ulazne kinete uočeno je nepostojanje zaštitnog sloja i njegova nedovoljna debljina, kao i korozija armature.

Karakteristična oštećenja na objektu AII su korozija armature usled prisustva vlage i malog zaštitnog sloja, prsline na mestima prekida betoniranja, procurivanje i ispiranje $Ca(OH)_2$, ljuskanje i otpadanje betona, biološka korozija, oštećene dilatacije. Pored toga, na spoljašnjim zidovima su vidljive pukotine usled hidrauličkog skupljanja (Slika 2), mrlje od rđe, betonska gnezda, a na dovodnoj tunelskoj konstrukciji: prsline usled sedanja betona sa procurivanjem, oštećena termoizolacija i industrijski pod. Na pregradnim zidnim nosačima uočene su pukotine kroz ceo presek usled gubitka oslonaca.

Slika 2. *Prsline usled hidrauličkog skupljanja*

6.3. Nedestruktivna ispitivanja

Nedestruktivnim metodama [2] ispitano je postojanje karbonizacije betona kolorimetrijskom metodom na objektu AII i izmerena je površinska tvrdoća betona upotrebom sklerometra na 8 mesta na spoljašnjim zidovima otvorenog dela objekta AI i objekta AII. Utvrđeno je postojanje karbonatizovanog betona, ali dubina nije mogla biti precizno određena. Merenjem površinske tvrdoće betona sklerometrom dobijene su vrednosti čvrstoća većih od projektom predviđenih 30 MPa (Tabela 1). Uzrok za ovo povećanje čvrstoće pri pritisku je najverovatnije karbonizacija betona.

Tabela 1. *Deo rezultata ispitivanja sklerometrom*

merno mesto 1			
40	48	33	39
37	31	40	33
40	37	38	50
31	41	38	46
odskok, sr = 38.875			
čvrstoća = 38 MPa			

6.4. Zaključak

Na osnovu analize podataka dobijenih vizuelnim pregledom dostupnih elemenata konstrukcije, te rezultata nedestruktivnih ispitivanja, zaključeno je da nosivost i stabilnost konstrukcije u celini nije ugrožena, dok se to ne može reći za upotrebljivost i trajnost. Što se tiče pojedinačnih elemenata, ističe se da je ugrožena stabilnost pregradnih zidnih nosača između peskolova i mastolova, usled promene statičkog sistema gubitkom oslonca.

7. SANACIJA

Predlog sanacije bazira se na Evropskom standardu EN 1504 [4-8]. U ovom standardu definisani su sistemi i primena konstruktivne i nekonstruktivne sanacije, karakteristike materijala za sanaciju i zaštitu betona i armature, ali i informacije o kontroli kvaliteta radova i primeni proizvoda na licu mesta. Prema standardu EN 206-1 definisane su klase izloženosti elemenata konstrukcije radi obezbeđenja trajnosti konstrukcije i sanacionih materijala [3].

Predviđeno je izvođenje i konstruktivne i nekonstruktivne sanacije. Predlozi sanacije dati su za svako od oštećenja po elementima ili celinama, pa su tako obrađena sanaciona rešenja za zidove crpnih pumpi, ulaznu kinetu, tunelsku konstrukciju za dovod otpadne vode u rezervoar AII, pregradne elemente rezervoara AII, spoljašnje zidove rezervoara AII i razvodnu tunelsku konstrukciju. U okviru svakog od ovih delova tabelarno su data uočena oštećenja i predlozi sanacionih rešenja, opis pripremnih radova, takođe, tabelarno su prikazani odabrani principi, metode sanacije (Tabela 2) i karakteristike materijala prema EN 1504 [4-8], kao i opis sanacionih radova [3].

Tabela 2. *Primer odabranih principa i metoda sanacije*

Deo ulazne kinete		
Princip	**Metoda**	**Standard**
1. Zaštita od prodora (PI)	1.3. Premazivanje	EN 1504 – 2
2. Kontrola vlažnosti (MC)	2.3. Premazivanje	EN 1504 – 2
3. Restauracija betona (CR)	3.2. Ulivanje betona ili maltera	EN 1504 – 3
7. Očuvanje/obnavljanje pasivnosti (RP)	7.1. Povećanje debljine zaštitnog sloja dodatnim malterom ili betonom	EN 1504 – 3
	7.2. Zamena kontaminiranog ili karbonizovanog betona	EN 1504 – 3
8. Jačanje otpornosti (IR)	8.3. Obloga/premazivanje	EN 1504 – 2
11. Kontrola anodnih oblasti (CA)	11.1. Aktivni premaz armature	EN 1504 – 7

Na svim elementima predviđena je zaštita armature, nanošenje novog zaštitnog sloja betona i zaštita u vidu završnog premaza.

Zaštitni premaz armature treba da ispuni princip CA (11.1) [8]. Takođe, treba da bude na bazi polimera i da štiti armaturu od korozije i omogući dobru atheziju sa reparaturnim materijalom.

Novi beton i novi zaštitni sloj treba da zadovolje principe CR (3.1. i 3.2.), PR (5.3.), RC (6.3.) i RP (7.1., 7.2.), prema EN 1504 – 3 [6]. Standardom EN 1992-1-1 definiše se minimalna debljina zaštitnog sloja c_{nom} [9]. Preporučuje se dodavanje polipropilenskih vlakana ovom materijalu, beton bi trebalo da je polimer-modifikovani, a savetuje se i upotreba sulfatno-otpornog cementa zbog agresivnog dejstva sredine.

Površinski premaz treba da zadovolji sledeće principe: PI (1.3.), MC (2.3.) i IR (8.3.) i treba da ispunjava sledeće uslove:

- Linearno skupljanje - ≤ 0.3%,
- Termički koeficijent - $\alpha T \leq 30 \cdot 10\text{-}6 \cdot K\text{-}1$,
- Propustljivost vodene pare: klasa II -5m < Sd < 50m,
- Propustljivost CO2 - Sd > 50m,
- Adhezija - ≥ 1,0 N/mm2,
- Kapilarno upijanje - $w < 0{,}1\ kg/m^2 \cdot h^{0,5}$ [5].

Nekonstruktivna sanacija prslina metodom zasecanja i zapunjavanja izvršiće se kod zidova crpnih pumpi, spoljašnjih zidova rezervoara AII i dovodne i razvodne tuneske konstrukcije.

Materijal za zapunjavanje prslina treba da ispuni princip PI (1.5.) [7], da ima mogućnost nanošenja na vlažan supstrat, vodootpornost i vrednost zatezne adhezije ≥ 3,0 N/mm^2.

Konstrukcijska sanacija prslina u vidu injektiranja primeniće se na spoljašnjim zidovima i pregradnim zidnim nosačima rezervoara AII, kao i kod dovodne tunelske konstrukcije.

Materijal za injektiranje pukotina/prslina mora ispunjavati kriterijume za princip SS (4.5.). Prema EN 1504 – 5 [7], zahtevane karakteristike materijala su:

- Adhezija - prvo gubitak kohezije u substratu,
- Skupljanje - < 3%,
- Čvrstoća pri zatezanju cepanjem - > 7 N/mm2,
- Razvoj čvrstoće na zatezanje - > 3 N/mm^2 posle 72h (za minimalnu dozvoljenu temperaturu negovanja).

Preporuka je da se za injektiranje koriste epoksidne smole.

Pregradnim zidnim nosačima P1 potrebno je obezbediti izgubljene oslonce. Oslonac će biti izveden u obliku čeličnog stuba koji podupire zidni nosač.

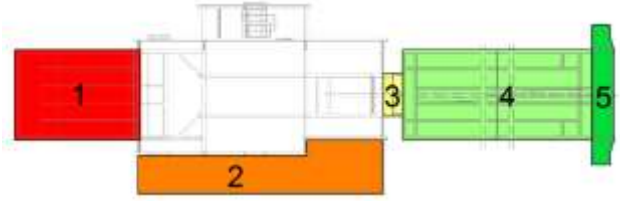
Zamena oštećenih dilatacionih spojnica predviđa se kod spoljašnjih zidova nenatkrivenog objekta AI i objekta AII. Pri tome se vrši zamena betona celom debljinom poprečnog preseka u ovim zonama.

Na gornjoj ploči dovodne tunelske konstrukcije predviđa se izvođenje novog industrijskog poda od modifikovane poliuretanske smole, agregata i cementa, na bazi vode sa samorazlivajućim svojstvima [11]. Na spoljnjim stranama bočnih zidova i na spoljnoj strani donje ploče ovog dela konstrukcije neophodno je postavljanje nove termoizolacije. Nova termoizolacija treba da bude otporna na dejstvo mraza i atmosferskih uticaja, kako ne bi došlo do oštećenja koja su zatečena na starim termoizolacionim pločama. S tim u cilju, preporučuje se upotreba ploča od ćelijastog (celularnog) stakla, jer je to jedini materijal koji bi ispunio navedene zahteve.

8. FAZE IZVOĐENJA RADOVA

Objekte za sanaciju možemo podeliti u 5 nezavisnih celina kao na slici 3 i to:

1. Zidovi crpnih pumpi i gredni element zgrade AI,
2. Ulazna kineta,
3. Tunelska konstrukcija za dovod otpadne vode u rezervoar AII,
4. Rezervoar AII,
5. Razvodna tunelska konstrukcija.



Slika 3. *Celine*

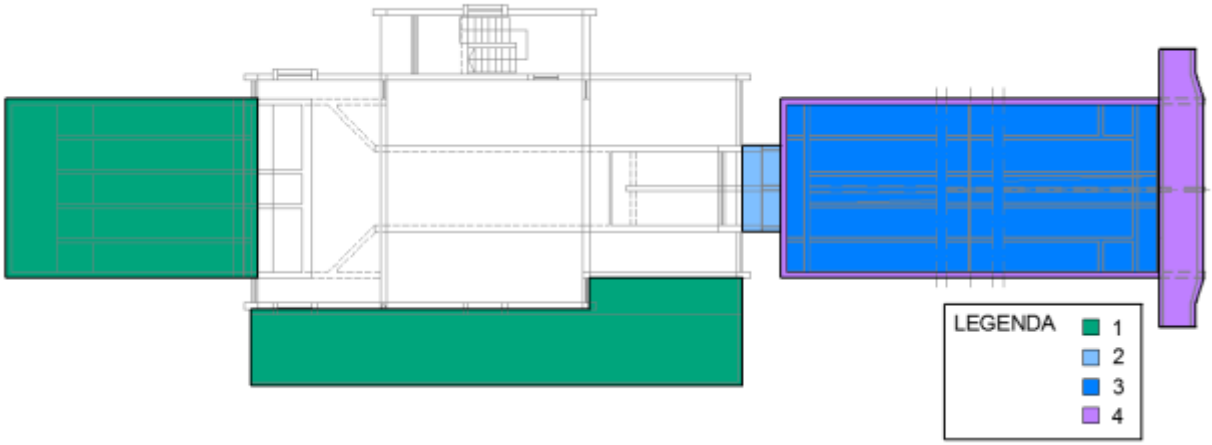
Pre početka radova, neophodno je obustaviti dovod otpadne vode u sistem, kako bi se rezervoari ispraznili i očistili. Ovo podrazumeva i prekid rada crpnih pumpi i automatske rešetke za prečišćavanje. Vrlo je važno da se sanacija obavi u što kraćem vremenskom periodu, kako bi postrojenje što pre nastavilo sa radom.

Potrebno je izvršiti prvo pripremne, a zatim sanacione radove u okviru kojih su dati i završni radovi. Konstrukcijska sanacija prslina treba da se izvrši pre početka pripreme površina i uklanjanja betona.

Što se tiče preporuka i varijacija na temu redosleda izvođenja radova po celinama, istovremeno se mogu izvoditi radovi u okviru zidova crpnih pumpi, grednog elementa zgrade AI i ulazne kinete, jer su pozicije elemenata za saniranje blizu, a obim sanacionih radova na delu ulazne kinete mali.

Paralelno sa ovim radovima, mogu se vršiti radovi na tunelskoj konstrukciji za dovod otpadne vode u rezervoar AII, s tim što se preporučuje da završni radovi na ovoj celini budu izvedeni na kraju, po završetku sanacionih radova na rezervoaru AII.

Što se tiče rezervoara AII, preporuka je da se prvo izvedu radovi na unutrašnjim, pregradnim elementima rezervoara, a zatim da se pristupi saniranju spoljašnjih zidova rezervoara. Priprema i sanacija razvodne tunelske konstrukcije može teći paralelno sa radovima na spoljašnjim zidovima rezervoara AII (Slika 4).



Slika 4. *Redosled izvođenja radova po celinama*

9. LITERATURA

[1] Muravljov M.: *Osnovi teorije i tehnologije betona*, V izdanje, Beograd 2010.

[2] Malešev M., Radonjanin V.: *Trajnost i procena stanja betonskih konstrukcija*, Skripta sa predavanja, Fakultet tehničkih nauka, Novi Sad

[3] Radonjanin V., Malešev M.: *Sanacija betonskih konstrukcija*, Skripta sa predavanja, Fakultet tehničkih nauka, Novi Sad

[4] „EN 1504-1 Products and systems for the protection and repair of concrete structures: Definitions“

[5] „EN 1504-2 Products and systems for the protection and repair of concrete structures: Surface protection systems for concrete“

[6] “EN 1504-3 Products and systems for the protection and repair of concrete structures: Structural and non-structural repair“

[7] „EN 1504-5 Products and systems for the protection and repair of concrete structures: Concrete injection“

[8] „EN 1504-7 Products and systems for the protection and repair of concrete structures: Reinforcement corrosion protection“

[9] „SRPS EN 1992-1-1“, 2015.

[10] http://rc5.gaf.ni.ac.rs/dec/arhcons/doc/homes/kostic/Osnovne%20Studije%20Arhitekture/Arhitektonske%20konstrukcije%20II/Predavanje%202%20-%20Dilatacione%20razdelnice%20%282016-2017%29.pdf (pristupljeno u septembru 2018.)

[11] www.srb.sika.com (pristupljeno u avgustu 2018.)

Kratka biografija:

Jelena Cvetković rođena je u Valjevu 1992. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Građevinarstvo – Konstrukcije odbranila je 2018. god.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 624.042.7
DOI: https://doi.org/10.24867/01CG03Prodanovic

PROJEKAT VIŠESPRATNE ZIDANE ZGRADE PREMA EVROKODU I UPOREDNA ANALIZA DOMAĆIH I EVROPSKIH STANDARDA ZA ZIDANE ZGRADE

DESIGN OF MULTY-STOREY MASONRY BUILDING AND COMPARATIVE ANALYSIS OF NATIONAL AND EUROPEAN STANDARDS

Ljubiša Prodanović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAĐEVINARSTVO

Kratak sadržaj – *U radu je prikazan projekat višespratne zidane zgrade spratnosti (Pr + 3) na području Novog Sada, prema Evrokod standardima. U drugom delu rada prikazana je uporedna analiza proračuna zidanih zgrada prema domaćim propisima i Evrokodu.*

Ključne reči: *Višespratna zidana zgrada, upredna analiza, Evrokod.*

Abstract – *The paper contains the design project of multy-storey masonry building (GF + 3) in Novi Sad according to Eurocode standards. In the second part of papers is presented a comparativ analysis of national and Eurocode regulations for the calculation of masonry buildings.*

Keywords: *Multy-storey masonry building, comparativ analysis, Eurocode.*

1. UVOD

Projektnim zadatkom je predviđena izgradnja višespratne zidane zgrade spratnosti (Pr+3) u Novom Sadu, pravougaonog oblika u osnovi, a prema zadatom arhitektonskom rešenju. Fundiranje je izvršeno na temeljnoj ploči ojačanoj temeljnim gredama. Noseća konstrukcija objekta projektovana je kao zidana konstrukcija, sa AB međuspratnim tavanicama, AB stepenicama za vertikalnu komunikaciju i zidovima za ukrućenje. Podaci o dejstvima [1, 2] uzeti su u skladu sa namenom objekta kao i podaci o tlu u skladu sa lokacijom. Projektom su obuhvaćeni analiza opterećenja, proračun merodavnih uticaja, dimenzionisanje, neophodni konstrukcijski detalji, kao i planovi oplate i armiranja. U istraživačkom delu sprovedena je uporedna analiza proračuna zidanih zgrada prema Evrokodu i domaćim propisima.

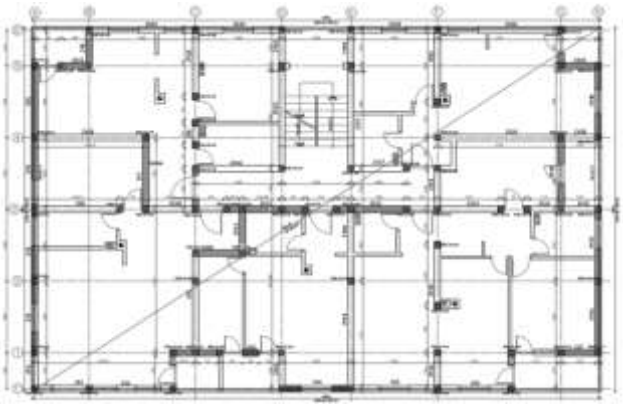
2. OPIS PROJEKTA

2.1 Arhitektonsko rešenje

Objekat je pravugoane osnove dimenzija 23,05 x 14,75 m. Arhitektonskim konceptom predviđeno je da suteren (prizemlje) bude iskorišten kao skladišni prostor, odnosno ostave, pojedinih stanova i 8 garažnih mesta. Tri tipske etaže predviđene su kao stambeni deo objekta, sl. 1.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Đorđe Lađinović, red. prof.

Za vertikalnu komunikaciju predviđeno je stepenište postavljeno tako da bude u neposrednoj blizini ulaza u zgradu ali i u blizini samih ulaza u pojedine stanove. Prilikom odabira položaja stepeništa u zgradi jedan od uslova je bilo i prirodno osvetljenje ovog prostora. Kao krovno rešenje predviđen je ravni krov.



Slika 1. *Tipska stambena etaža*

2.2. Konstruktivni sistem

Projektnim zadatkom predviđen je konstruktivni sistem kao zidana zgrada, što podrazumeva glavne nosive zidane zidove, ortogonalno ravnomerno raspoređene. Zidovi su izvedeni od opeke projektovane marke, ukrućeni su horizontalnim i vertikalnim serklažima, te je međuspratna tavanica izvedena kao kontinualna krstasto armirana AB ploča. Zbog velike krutosti objekta, smanjen je broj nosivih zidanih zidova, te su isti zamenjeni zidovima od lakoagregatnih materijala koji služe isključivo kao pregradni. Nakon dinamičke analize, radi postizanja boljeg ponašanja konstrukcije prilikom seizmičkog delovanja dodat je jedan AB zid za ukrućenje koji se proteže od temelja do krovne ploče, nalazi se u fasadnom zidu na južnoj strani objekta. Osna rastojanja variraju od 1,45 do 5,00 m u oba pravca.

Temeljna ploča je ojačana grednim nosačima na mestima gde automobili ulaze u garaže radi ublažavanja i boljeg prihvatanja dinamičkog dejstva istih.

Stepenice su formirane kao dvokrako stepenište sa međupodestom na polovini spratne visine. Dimenzije kao i položaj stepeništa je uslovnjeno prirodnim osvetljenjem i udobnim korištenjem u skladu sa namjenom objekta.

Armirano betonsko delovi konstrukcije su izvedeni u klasi betona C25/30, a korišćen je čelika za armiranje

S500. Kvalitet zidanih zidova je definisan u skladu sa preporučenim vrednostima EC (f_b = 10 MPa; f_m = 5 MPa).

2.3. Analiza opterećenja

Sopstvena težina konstruktivnih elemenata (grede, stubovi, zidovi, ploče...) automatski su generisani prema zadatim parametrima. Sopstvena težina nekonstruktivnih elemenata koji imaju karakter stalnog opterećenja (podne podloge i obloge, krov, pregradni zidovi, instalacije, opterećenje od tla i td.) imaju karakteristične vrednosti usvojene u skladu sa EN 1991-1-1:2002 [2] a potom dodatno aplicirane na model za proračun.

Korisna opterećenja, u zavisnosti od namene objekta i njegovih delova usvojena su prema EN 1991-1-1:2002 [2], u ovom slučaju razmatrane su sledeće grupe prostorija: stambeni prostor; balkoni i stepeništa; hodnici; garaže i ostave; krov.

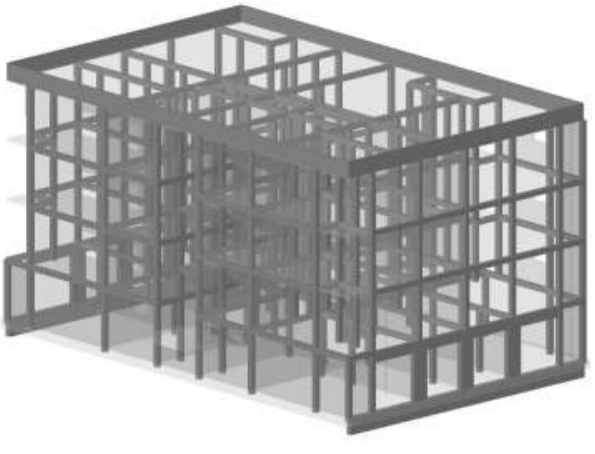
Opterećenje snegom je razmatrano za datu lokaciju, Novi Sad, čija je nadmorska visina oko 81 mnm, usvojeno je u skladu sa EN 1991-1-3:2003 [3], u iznosu od 1,0 kN/m^2 (na strani sigurnosti).

Proračun opterećenja vetrom sproveden je primenom standarda EN 1991-1-4:2005 [4], modelirano je kao površinsko opterećenje na fasadne zidove, i automatski konvertovano na linijsko tamo gdje je to bilo potrebno.

Seizmičko opterećenje je generisano primenom softwera Tower 6.0, a u skladu sa EN 1998-1-2004 [7]. Za izračunavanje seizmičkih sila primenjena je multimodalna spektralna analiza.

2.4. Statički i dinamički proračun

Statički i dinamički proračun je izvršen primenom softverskog paketa Tower 6.0 [11], primenom teorije prvog reda.



Slika 2. *3D model*

Tlo je modelirano kao Vinklerov model tla, što podrazumeva niz elastičnih opruga koje omogućavaju rad konstrukcije koja odgovara približno realnim uslovima.

Prilikom modalne analize usvojene su pretpostavke da je međuspratna tavanica kruta i da su mase koncetrisane u nivoima tavanica, usvojeni su konačni elementi dimenzija 0,25x0,25 m radi tačnijih rezultata.

Kao rezultat dinamičke analize modela dobijeni su periodi oscilovanja koji su dalje iskorišćeni pri seizmičkom proračunu, a za definisanje koeficijenata učešća masa za modalnu analizu korišćene su odredbe EN 1991:2002.

Za određivanje uticaja u nosećoj konstrukciji od dejstva seizmičkih sila primenjena je multimodalna spektralna analiza u saglasnosti sa odredbama EN 1998-1:2004, a proračun je sproveden primenom softverskog paketa Tower 6.0.

Prema seizmološkoj karti za predmetnu lokaciju objekta usvojeno je projektno ubrzanje tla u iznosu $a_g = 0{,}2g$, a projektni elastični spektar je konstruisan za kategoriju tla tipa „C“ i III kategorija objekta.

Faktor ponašanja usvojen je sa vrednošću q = 2,5, isti je u funkciji klase dutilnosti objekta (k_D = 0,5 za DC „L“), pravilnosti konstrukcije objekta po visini (k_R = 1,0 za pravilne konstrukcije), i preovlađujućeg loma konstruktivnih sistema sa zidovima (k_W = 1,0).

Za potrebe dimenzionisanja definisana su dva pravca delovanja seizmičkih sila u X i Y pravcu, ali nisu iskorišteni kao takvi nego kao kvadratni koren zbira kvadrata zadatih dejstava i na taj način je formirano jedinstveno seizmičko dejstvo alternativnog karaktera.

2.5. Proračunske kontrole

Sprovedene su proračunske kontrole u cilju potvrde kvaliteta odabranog koncepta konstrukcije. Izvršene su sledeće kontrole:

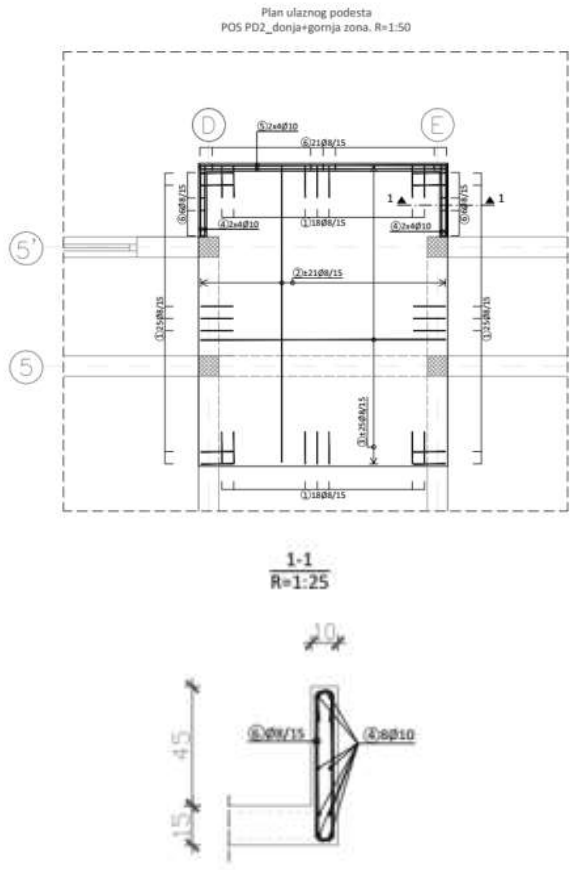
- Provera graničnog stanja nosivosti za nosivost tla u skladu sa EN 1997-1:2004, napon se kontroliše za dve kombinacije opterećenja.
- Provera graničnog stanja upotrebljivosti za sleganje tla u skladu sa EN 1997-1:2004.
- Kontrola normalnog napona u stubovima u skladu sa EN 1992-1-1:2004 za kvalitet betona C25/30.
- Kontrola relativnog spratnog pomeranja prema EN 1998-1:2004, pomeranja su očitana kao pomeranja krovne ploče, kao najvišeg dela objekta.
- Kontrola zidanih zidova i to: a) kontrola vitkosti zida; b) kontrola na vertikalno opterećenje; c) kontrola zida na bočno opterećenje; d) kontrola zida na smičuće opterećenje.

2.5. Dimenzionisanje elemenata

Primenom softwerskog paketa izvršeno je dimenzionisanje prema kompletnoj šemi opterećenja, merodavne kombinacije opterećenja su automatski odabrane. Dimenzionisanje i armiranje elemenata je izvedeno prema EN 1992-1-1:2004. Zaštitni slojevi su definisani prema klasama izloženosti.

Dimenzionisani su svi AB elementi: ploče, stubovi, grede i serklaži kako vertikalni tako i horizontalni u cilju provere da li minimalna zahtevana armatura zadovoljava proračunske zahteve. Zidani zidovi su prethodno u arhitektonskom rešenju pretpostavljeni te kao takvi proračunski kontrolisani, ustanovljeno je da svi nosivi zidovi (pretpostavljene debljine d_z = 25 cm) zadovoljavaju uslove po pitanju dimenzija.

Poštujući sve odredbe i preporuke Evrokoda izrađeni su planovi oplate i armature za sve razmatrane elemente. Kao primer, prikazani su planovi armiranja podesta nadstrešnice, sl. 3.



Slika 3. *Plan armiranja podesta – Nadstrešnice*

3. UPOREDNA ANALIZA

U okviru istraživačkog dela Master rada zadatak je da se izvrši poređenje proračuna zidanih konstrukcija prema Pravilniku o tehničkim normativima za zidane zidove [9] i proračuna zidanih konstrukcija prema EN 1996-1-1 [6]. Poređenje je sprovedeno teoretski uz opisivanje pojedinačno svakog koraka pri procesu proračuna zidanog zida uz mogućnost uvida u prednosti i mane svakog od načina proračuna.

3.1. Analiza opterećenja

Pored opterećenja stalnog karaktera koja su jednaka u oba slučaja, veoma bitnu su ona opterećenja koja su u funkciji namene objekta kao i opterećenja u zavisnosti od spoljnih uticaja (vetar, sneg i seizmičko dejstvo). Korisna opterećenja su ona koja se značajno razlikuju, te je uočeno da EN detaljnije razmatra moguće slučajeve te daje mogućnost boljeg definisanja opterećenja. Opterećenje od vetra i snega imaju približno iste vrednosti, dok se značajno razlikuje seizmičko opterećenje gde je kod domaćeg pravilnika za određivanje uticaja primenjena metoda statički ekvivalentnih sila, a u slučaju EN multi modalna spektralna analiza.

3.2. Elementi za zidanje

Domaći pravilnik za zidane konstrukcije predviđa elemente za zidanje od veštačkog ili prirodnog kamena kao i EN standardi. Razlika je u tome što EN zahteva detaljniju klasifikaciju elemenata za zidanje prema šupljinama kako vertikalnim tako i horizontalnim, dok je prema domaćem pravilniku zabranjena upotreba elemenata sa horizontalnim šupljinama za zidanje nosivih zidova. Čvrstoće na pritisak, savijanje i smicanje se po istom principu određuju, kao i modul elastičnosti i modul klizanja.

3.3. Analiza elemenata konstrukcije

Pri analizi elemenata konstrukcije misli se na analizu dejstava, ekscentričnosti, uticaje II reda i td. Domaći pravilnik je u ovom delu sačinjen po ugledu na EN, te su razlike minimalne. Osim gore nabrojanih analiza u ovom delu su obrađene:

- Efektivna visina nosećeg zida – on mora da bude procenjena uzimajući u obzir relativne krutosti elemenata konstrukcije koji su povezani sa posmatranim zidom, kao i efikasnost veza.
- Efektivna debljina zida jednostrukog zida, dvostrukog zida, zida sa licem, zida sa horizontalnim spojnicama – trakama i popunjenog dvoslojnog zida.
- Vitkost zida dobijena kao odnos vitkosti efektivne visine zida i efektivne debljine zida, vrednost ne sme da bude veća od 27.

3.4. Proračun nosivosti zida i kombinacije dejstava

Pri formiranju kombinacija dejstava, kako prema domaćim propisima tako i prema EN standardima razmatra se parcijalni keoficijent sigurnosti kao i faktor kombinacije pojedinih slučajeva dejstava, te u zavisnosti od proračuna usvaja se preporučena kombinacija opterećenja.

Prema domaćem pravilniku proračun može da se sprovede prema dozvoljenim naponima i prema graničnom stanju loma. EN predviđa proračun nosivih zidova prema graničnom stanju nosivosti i graničnom stanju upotrebljivosti. Za razliku od domaćeg pravilnika, u EN se dopušta primena uprošćene metode proračuna ukoliko su zadovoljeni određeni uslovi kao što su spratno ograničenje, pravilnost objekta i td.

3.5. Konstrukcijsko oblikovanje

U oba slučaja zahteva se ostvarenje pravilnog zidnog sloga uz poštovanje svih zahteva kada su u pitanju horizontalne i vertikalne spojnice, minimalna površina zida kao i minamalna debljina zida.

Dodatnu pažnju oba pravilnika zahtevaju pri postojanju žlebova bilo kog oblika, jer u određenim situacijama mogu značajno da utiču karakteristike zida.

3.6. Seizmički uslovi i pravila za zidane zgrade

Prema pravilniku o tehničkim normativima za izgradnju objekata visokogradnje u seizmičkim područjima, osnovni sistem zidanih konstrukcija su noseći zidovi u oba ortogonalna pravca objekta povezani u visini međuspratnih konstrukcija horizontalnim serklažima. Međuspratne konstrukcije moraju biti krute u svojoj ravni, nije dozvoljena kombinacija AB skeleta i masivnog sistema.

Najveći razmak zidova se kreće u rasponu od 5,00 do 7,50 m u zavisnosti od debljine zida. Vertikalni serklaži se betoniraju nakon zidanja i njihov razmak ne sme biti veći od 5,00 m. Za zidanje u seizmičkim područjima dozvoljena je upotreba samo produžnog cementnog maltera, propisana je minimalna čvrstoća altera u zavisnosti od seizmičke zone, kreće se od M25-M50.

Evrokod predviđa minimalne vrednosti čvrstoće na pritisak kako za elemente za zidanje tako i za malter:

- Elementi za zidanje, upravno na površinu spojnice, $f_{b,\min} = 5$ N/mm^2.
- Malter, za nearmirane zidove i zidove sa serklažima, $f_{m,\min} = 5$ N/mm^2.

Smičući zidovi se moraju postaviti u najmanje dva ortogonalna pravca, može se iskoristiti bilo koji tip tavanica ako su ispunjeni opšti uslovi za kontinuitet i efektivno delovanje dijafragme. Razmak vertikalnih serklaža ne sme biti veći od 5,00 m, dok je maksimalan razmak zidova 7,00 m.

U evrokodu se dopušta izostavljanje direktnog dokaza sigurnosti svakog konstrukcijskog elementa ukoliko su zadovoljeni uslovi za definisanje objekta kao "jednostavna zidana zgrada".

4. ZAKLJUČAK

Na osnovu svega gore navedenog, moguće je uočiti tri značajne razlike:

- Prva razlika odnosi se na vrednosti korisnih opterećenja zgrada – u evropskim standardima vrednosti opterećenja su malo veće nego u našim propisima, pa se mogu očekivati i razlike u veličini uticaja u nosećoj konstrukciji;
- Druga razlika se odnosi na seizmičku analizu i na proračun seizmičkog dejstva – u nacionalnim propisima se za proračun uticaja od seizmičkog dejstva koristi statički ekvivalentna metoda dok se u evropskim standardima proračun sprodvodi metodom multi modalne analize;
- Treća razlika se odnosi na način proračuna zidanih zidova – u Evrokodu je omogućena primena uprošćenih postupaka i metoda ukoliko su zadovoljeni uslovi koje klasifikuje objekat kao "jednostavna zidana zgrada", te proračun noseće konstrukcije postaje znatno brži i jednostavniji.

4. LITERATURA

[1] Pakvor Aleksandar, Perišić Života, Aćić Mirko, "Evrokod 0: EN 1990:2002. Osnove proračuna konstrukcija".Prevod sa Engleskog jezika: dr Aleksandar Pakvor. Građevinski fakultet Univerziteta u Beogradu: Beograd 2009.

[2] Najdanović Dušan, "Evrokod 1: EN 1991-1-1:2002. Dejstva na konstrukcije; deo 1-1: Zapreminske težine, sopstvena težina, korisna opterećenja za zgrade".Prevod sa Engleskog jezika: dr Aleksandar Pakvor. Građevinski fakultet Univerziteta u Beogradu: Beograd 2009.

[3] Najdanović Dušan, "Evrokod 1: EN 1991-1-3:2003. Dejstva na konstrukcije; deo 1-3: Dejstva snega".Prevod sa Engleskog jezika: dr Aleksandar Pakvor. Građevinski fakultet Univerziteta u Beogradu: Beograd 2009.

[4] Marković Zlatko, "Evrokod 1: EN 1991-1-4:2005. Dejstva na konstrukcije; deo 1-4: Dejstva vetra".Prevod sa Engleskog jezika: dr Aleksandar Pakvor. Građevinski fakultet Univerziteta u Beogradu: Beograd 2009.

[5] Pakvor Aleksandar, Perišić Života, Aćić Mirko, "Evrokod 2: EN 1992-1-1:2004. Proračun betonskih konstrukcija; deo 1-1: Opšta pravila i pravila za zgrade".Prevod sa Engleskog jezika: dr Đivota Perišić. Građevinski fakultet Univerziteta u Beogradu: Beograd 2006.

[6] Stevanović Boško, Glišović Ivan, Muravljov Mihailo, Jevtić Dragica, Zakić Dimitrije, "Evrokod 6: EN 1996-1-1:2005. Proračun zidanih konstrukcija; deo 1-1: Opšta pravila za armirane i nearmirane zidane konstrukcije".Prevod sa Engleskog jezika: dr Stevanović Boško, mr Glišović Ivan. Građevinski fakultet Univerziteta u Beogradu: Beograd 2009.

[7] Đorđe Lađinović, "Evrokod 8: EN 1998-1:2004. Proračun seizmički otpornih konstrukcija; deo 1:Opšta pravila, seizmička dejstva i pravila za zgrade".Prevod sa Engleskog jezika: dr Lađinović Đorđe, dr Folić Radomir, dr Brčić Stanko, dr Brujić Zoran, dr Tatjana Kočetov Mišulić, Andrija Rašeta. Građevinski fakultet Univerziteta u Beogradu: Beograd 2009.

[8] Muravljov Mihailo, Stevanović Boško, "Zidane i drvene konstrukcije zgrada. Građevinski fakultet Univerziteta u Beogradu: Beograd 2003

[9] Pravilnik o tehničkim normativima za zidane zidove. Beograd 1991.

[10] Pravilnik o tehničkim normativima za objekata visokogradnje u seizmičkim područjima.

[11] "Tower 6.0" – Upustvo za rad sa programom.

[11] "Allplan 2015" – Upustvo za rad sa programom.

Kratka biografija:

Ljubiša Prodanović rođen je u Doboju 1990. godine. Osnovne akademske studije završio je 2015. godine na Fakultetu tehničkih nauka u Novom Sadu. Master rad iz oblasti "Seizmička analiza konstrukcija" pod mentorstvom prof. dr Đorđa Lađinovića odranio je 2018. godine.

Kontakt: ljubisa_pro@hotmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 624.9
DOI: https://doi.org/10.24867/01CG04Kurcubic

IZGRADNJA MAGISTRALNOG KANALA OROM-ČIK-KRIVAJA PODSISTEMA TISA-PALIĆ, REPUBLIKA SRBIJA

CONSTRUCTION OF THE MAIN-MAGISTRAL CHANNEL OROM-ČIK-KRIVAJA SUBSYSTEM TISA-PALIĆ, REPUBLIC OF SERBIA

Branka Kurćubić, Vladimir Mučenski, *Fakultet tehničkih nauka, Novi Sad*

Oblast – TEHNOLOGIJA I ORGANIZACIJA GRAĐENJA

Kratak sadržaj – *predmet ovog rada jeste planiranje i organizovanje izvođenja kanalskog profila na II deonici magistralnog kanala Orom-Čik-Krivaja od km 9+450 do km 24+450. Ovim se podrazumeva planiranje procesa tehnologije iskopa i nasipa profila, izbor odgovarajuće protivfiltracione obloge, kao i izbor odgovarajuće mehanizacije.*
Ključne reči: *Tehnologija i organizacija građenja, Kanal Orom-Čik-Krivaja*

Abstract – *The subject of work is planning and organizing perform of channel profile on II section of magistral channel Orom-Čik-Krivaja from the km 9+450 to km 24+450. This implies planning the process of technologies of the excavation and the profile of the embankment, the selection of the appropriate anti-filtration coating, as well as the selection of the appropriate mechanization.*
Keywords: *Construction technology, Orom-Čik-Krivaja channel*

1. UVOD

Severna Bačka predstavlja teritoriju omeđenu mađarskom granicom, rekom Tisom, kanalom DTD Bezdan-Vrbas-Bečej, ukupne površine oko 450.000 ha. Čitavo područje se proteže između kota 90 mnm i 130 mnm, a generalne prirodne karakteristike predstavljaju specifičan geološki sastav tla, duboki nivoi podzemne vode i visoka plodnost zemljišta.
Radi prevazilaženja problema deficita vlage, još 40-ih godina prošlog veka započete su aktivnosti radi zadržavanja sopstvenih voda na području, kao i dovođenja vode iz Dunava i Tise na području severne Bačke.
Namena predmetnog magistralnog kanala je pretežno snabdevanje vodom za navodnjavanje, zbog čega je izražen njegov sezonski karakter eksploatacije, pretežno u vegetacionom periodu, sa maksimumom tokom dva letnja meseca: jula i avgusta. Ta okolnost ima odlučujući uticaj na cenu m^3 vode, pa svaka vansezonska potrošnja može povećati njegovu produktivnost i ekonomičnost. S druge strane sezonski rad omogućava održavanje kanala u visokom stepenu funkcionalnosti.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladimir Mučenski.

Takođe, omogućava i eventualne investicione zahvate u cilju povećanja njegovih kapaciteta.
Danas se u Vojvodini za potrebe poljoprivrede može koristiti oko 2 miliona ha, odnosno 92 % ukupne teritorije.

2. USLOVI IZGRADNJE I EKSPLOATACIJE KANALA

Sa morfološkog aspekta teren duž trase kanala je zatalasan i prilično neujednačen sa visinskim razlikama u intervalu između 105 mnm i 110 mnm. Kota projektovanog dna kanala se kreće u intervalu od 103,88 mnm do 103,04 mnm, a niveleta kanala od 105,9 do 104,30 mnm, evidentno je da se projektovani kanal na najvećem delu trase nalazi u useku, a samo delimično u nasipu.
Osnovna namena projektovanog kanala jeste da se njime transportuje voda za potrebe vodosnabdevanja, posebnu pažnju treba obratiti na promene okvašenog dela profila kao i na zone terena ispod projektovanog dna i iznad projektovane nivelete.
Kanal se u fazi eksploatacije mora održavati tj. čistiti, izmuljavati od rastvorenog taloga tako da će u funkciji vremena kanal dobiti pravougaoni oblik.

3. TRASA KANALA

Magistralni kanal deonica II počinje na račvi (km 9+450), gde se trasa kanala lomi prema severu, a magistralni kanal (deonica I) se rasterećuje (2,4 m3/s) preko spojnog kanalau Čik, odnosno akumulaciju Svetićevo. Deonicom II se distribuira 2,9 m3/s, sve do izgrađenog propusta na ukrštanju kanala (km 12+500) sa međunarodnim putem E-75 (Beograd-Subotica). Na toj stacionaži se proticaj smanjuje na 2,62 m3/s i takav se održava do kraja, do izliva u Krivaju (km 24+400). ukrštanje trase kanala i vodotoka Čik predviđeno je prelaskom magistralnog kanala preko deonice Čika po lokaciji nasute zemljane brane Azotara (km 15+275–15+475).
Dogradnjom nasipa za izradu korita kanala, kao i nadvišenjem postojeće krune brane na kotu 105,50 mnm, sa njene nizvodne strane, formiraće se kanal u nasipu, čija je osovina paralelona sa postojećom osovinom brane na razmaku 8,50 m. Na desnom boku doline, osovina kanala skreće na sever i seče osovinu brane, da bi tako nastavila do stacionaže km 16+450. ukrštanje sa železničkom prugom Beograd-Subotica biće ostvareno propustom na km 17+151.

4. TEHNOLOGIJA IZVOĐENJA RADOVA NA KANALU

Izvođenje radova na predmetnom kanalu predviđa se savremenim tehničkim sredstvima i materijalima, na najekonomičniji način. Reljef terena i njegov geološki sastav, kao i hidraulički zahtevi, uslovili su tehničko rešenje sledećih karakteristika:

Na predmetnoj deonici kanala razlikuju se tehnologije iskopa kanala i to u zavisnosti da li se radi o kanalu u useku ili o kanalu u nasipu. Za oba slučaja prva faza predstavlja skidanje sloja humusa i to 0,5 m (kanal u useku) i 0,3 m (kanal u nasipu). Humus se skida buldožerima i transportuje u privremene deponije pored kanala, da bi se kasnije izvršilo njegovo razastiranje na jednu ili obe strane ili materijal iskoristio za nasip. Jednostrani transport humusa predviđa se samo na deonicama sa deponijama i to na suprotnu stranu od njih. Na deonicama kanala u nasipu površine se humuziraju, profil nasipa izvan koherentnog tela se popunjava humusom. Preostale količine se razastiru, a nedostajuće količine za izradu proširenja nasipa, podužnim transportom iz deonica u useku nasipaju koherentnim materijalom. Proširenje nasipa nabija se u slojevima do 50 cm prelaskom točkova kamiona i gusenicama buldožera preko cele površine svakog sloja.

Izrada kanalskog profila u useku projektovana je u dve faze:

1. Široki iskop u gornjim delovima profila buldožerom i transport na jednu ili obe strane kanala na privremenu deponiju;
2. Iskop donjih delova profila kanala hidraulik bagerima i odlaganje na privremenu deponiju na jednu ili obe strane kanala.

Razastrta zemlja u sloju od 30 cm preko sloja humusa umanjiće plodnost zemljišta, te je predviđena rekultivacija tih površina. Pod tim se podrazumeva rigolovanje do dubine 70-80 cm kako bi se humus izbacio na površinu, te đubrenje stajnjakom u količini 10 vagona po hektaru tokom 6-10 godina.

Izrada kanalskog profila u nasipu-nakon skidanja humusnog sloja ispod zbijenog jezgra nasipa vrši se nabijanje posteljice vibroježevima, a zatim iskop kanalskog profila i nasipanje u slojevima do 50 cm sa planiranjem buldožerima i nabijanjem vibroježevima do zbijenosti 95% po Proktoru. Nedostajući materijal iz kanalskog profila za izradu nasipa transportuje se podužno kamionima iz deonica sa viškom iskopa. Predviđen je bagerski iskop profila sa jednovremenim utovarom u kipere. Lokacija pozajmišta iskopa sa trase nasipa određene su po ekonomskim, agronomskim I drugim kriterijumima. Profili nasipa izvan zbijenog koherentnog jezgra ispunjavaju se preostalim materijalom iz iskopa kanalskog profila ili podužno transportovanim materijalom sa deonica u useku. Zbijanje ovih delova profila predviđa se točkovima kamiona ili gusenicama buldožera.

Kvalitet tog materijala nije bitan pošto služi samo kao ispuna prilikom planiranja dna i kosine nasipa, koja je širine oko 0,5 m.

Za izradu protivfiltracione obloge kanala predviđeno je i valjanje kosina i dna glatkim valjkom uz prethodno kvašenje isplaniranih površina vodom iz cisterni, kako bi se ostvarila odgovarajuća posteljica za naleganje folije. Preciznost planiranja i valjanja posteljice obloge direktno utiče na njenu sigurnost i trajnost, kao i na protočnu sposobnost kanalskog profila.

Iskop ankernih rovova za foliju na bankinama, dimenzija 30x20 cm predviđa se ručno uz odbacivanje iskopanog materijala van kanalskog profila. Precizno čišćenje grudvi iz zemlje iz prethodno uvaljanog kanalskog korita predviđa se ručno sa evakuacijom u deponije.

Polaganje folije po profilu kanala treba da obavi stručna i specijalizovana organizacija, u skladu sa važećim standardima i iskustvima. Dužina folije u poprečnom i podužnom profilu treba da je veća 2-3% od stvarnih dimenzija korita, zbog naknadnog skupljanja folije. Nakon polaganja i spajanja folije na licu mesta, vrši se njeno ankerisanje u iskopanim rovovima po bankinama, tako što se ručno zatrpaju zemljom uz nabijanje.

Neobloženi, parabolični kanalski profil izvešće se bagerima sa planiranjem kosina i dna sa tačnošću ±10 cm u odnosu na projektovane linije. U nivou ureza horizontalnog nivoa predviđa se iskop posteljice za humusni tepih (0,1x0,1 m) bagerom, kojim se takođe nasipa humus sa ručnim razastiranjem i planiranjem. Nakon toga se vrši ručna sadnja trske, rogoza ili ševara.

Izlaz za životinje se formira od GeoWeb mreže postavljene preko vodonepropusne obloge kanala (HDPE folija-geomembrana), na bankini i po kosini kanala pokrivajući zonu oscilacije nivoa vode i ispod vode, tako da se životinja može pri izlasku osloniti na ispunjenju ćelije. Ćelije se iznad nivoa vode pune humusom i zatravljuju, a ispod nivoa vode se pune tucanikom granulacije 30-45 mm. Tucanik i humus su dobra podloga za izlazak životinja iz kanala.

5. MEHANIZACIJA

Buldožeri spadaju u mašine koje se koriste za iskop i/ili utovar. Otkopavanje buldožerima obavlja se zasecanjem materijala, uz njegov transport guranjem i razastiranjem, s grubim planiranjem. Najčešće se primenjuju za otkop i premeštanje materijala. Naročito su pogodni za skidanje humusa i pripremu terena za iskope ili izgradnju nasipa, a takođe za otkopavanje i izgradnju kraćih useka i materijalnih rovova odnosno zaseka, gde se prenos materijala obavlja na kraća rastojanja i najčešće u poprečnom smislu. Da bi se postigao optimalan kapacitet u određenim uslovima, poseban značaj ima izbor odgovarajućeg buldožera i primena racionalne tehnološke šeme rada. Radni učinak buldožera se povećava ako se transport materijala obavlja po padu terena. Pri kretanju po razstrtom materijalu, buldožeri obavljaju delimično i njegovo zbijanje, iako su efekti neretko slabi zbog malog specifičnog pritiska gusenica na tlo.

Hidraulični bager – osnovna namena hidrauličnog bagera je iskop masa, za koje operacije se koristi čeona i dubinska kašika različitih veličina i oblika, kombinovanim sa različitim dužinama strele, u zavisnosti od vrste iskopa i materijala. Nza ostale radove (planiranje, sabijanje, razbijanje stena i smrznute zemlje, kopanje bunara, podizanje tereta, itd.) koriste se odgovarajući radni organi koji se neprekidno usavršavaju uz evidentna poboljšanja na osnovnim mašinama, što bagerima još više proširuje i onako široku namenu.

Valjci sa glatkim točkovima – veoma je pogodan za dopunsko zbijanje po površini sloja kod nasipa izgrađenih od kamenih i drugih nevezanih materijala. Njima se postiže dobro zatvaranje sloja koji se zbija, odnosno ravnanje njegove površine.

6. RADOVI NA OBLAGANJU KANALA

Geotekstil je netkani tekstil koji je tehnologijom izrade dobijanja runa iz visokokvalitetnih sintetičkih vlakana poliestera i polipropilena i učvršćen mehaničkim putem, iglanjem, tj. stvaranjem petlji ili mršenjem vlakana u čvrstu tvorevinu sličnu filcu. Osnovne funkcije geotekstila su:

- Razdvajanje ili separacija (sprečava međusobno mešanje agregara, omogućava povećanje kompaktnosti, povećava nosivost i otpor na brazdanje, povećava otpor na smrzavanje u bazi agregata);
- Armiranje (izrazito svojstvo armiranja slabo nosivog tla i poboljšanje kapaciteta nosivosti tla);
- Filtriranje (zbog svoje strukture ima svojstvo propusnosti vopde kroz sam materijali zadržavanje čestica tla);
- Dreniranje (odvođenje vode iz tla, a pogodni za ovu funkciju su težih gramaža i deblji koji imaju značajnu poroznost);
- Izolacija (ova funkcija je potrebna u mnogim rešenjima u građevinarstvu).

Nepropusna HDPE folija se postavlja preko prethodno postavljenog geotekstila. Na krajevima se vrši sidrenje folije sa geotekstilom. Sidrenje se vrši klinovima dužine 20 cm, na rastojanju od 1 m. Podloga na koju se postavlja folija mora biti suva, ravna, dobro sabijena i bez oštrog kamenja ili vegetacije. Rolne HDPE folije razmoravaju s epo dužini i međusobno preklapaju na preklop od cca 12 cm. Krajevi folije s epostavljaju u ankerni rov, koji se po završetku postavljanja zatrpava. Pre punjenja kanala vodom potrebno je proveriti varove.
GEOWEB mreža-geomembrana – To je materijal koji se proizvodi ekstrudiranjem, od polietilena visoke gustoće ili polietilena vrlo niske gustoće. Može imati glatku ili hrapavu površinu. Primenjuju se za osiguranje vodonepropusnosti akumulacija, rezervoara, brana, kanala, izradu temeljnog i prekrivnog nepropusnog sloja odlagališta otpada, itd.

7. ORGANIZACIJA RADOVA

Na početku analize mogućih varijanti mehanizacije koja će se koristiti za iskop i nasip profila kanala, ceo kanal je podeljen na tri deonice na kojima se za obale I dno kanala koriste iste mašine. Analiza je vršena na osnovu rada dva buldožera, dva kamiona kipera za transport iskopanog materijala do obližnje deponije, dva hidraulička bagera i jednog glatkog valjka. Cene iskopa i materijala su ostale iste ali se procena bazira na vremenu potrebnom za iskop kanalskog profila.
Kao najpovoljnija usvojena je varijanta sa paralelizacijom radova jer sve aktivnosti počinju istovremeno a radovi će se završiti za nešto više od pet meseci, pre zime, pa neće doći do prekida radova.

8. ANALIZA VREMENA I CENA IZVOĐENJA RADOVA

8.1 Gantogram

Upotrebom programa Microsoft Project dobijen je gantogram radova. Gantogram je dinamički plan u kome su prikazane sve aktivnosti. On pokazuje datume početka i završetka svake aktivnosti, kao i ukupnu sliku odvijanja radova u toku vremena i omogućava uvid u projektovanu paralelizaciju radova. Gantogram omogućava kontrolu količine izvedenih radova i utrošak resursa. Kritične aktivnosti su prikazane crvenom bojom, dok su aktivnosti koje se ne nalaze na kritičnom putu prikazane plavom bojom.
Analiza različitih kombinacija iskopa kanala je obavljena izradom gantograma za tri moguće varijante. Ovim se podrazumeva upoređivanje potrebnog vremena za završetak predviđenih aktivnosti.
Kod usvajanja mehanizacije vodilo se računa o njihovom praktičnom učinku, ali i o ceni radova, kako kod biranja tipa mašina tako i kod usvajanja potrebnog broja istih mašina za ceo projekat.
Kao najbolja, usvojena je varijanta u kojoj se iskop profila kanala vrši iz tri pozicije sa paralelizacijom radova. Prva pozicija predstavlja iskop leve obale, druga pozicija iskop desne obale i treća pozicija predstavlja iskop dna kanala. Sve tri pozicije počinju istovremeno.
Kao dan početka rada usvojen je 04.03.2019. godine, a kraj radova predviđen je za 03.09.2019. godine. Usvojeno je da se radi šest radnih dana u nedelji, sa dvanaestočasovnim radnim vremenom. Radovi se neće izvoditi u zimskim mesecima (od novembra do marta).

9. ZAKLJUČAK

Predmet rada je izgradnja podsistema Tisa-Palić deonice II magistralnog kanala Orom-Čik-Krivaja, od km 9+450 do km 24+450, koja bi obezbedila potrebnu količinu vode za navodnjavanje poljoprivrednog zemljišta, ribnjake, snabdevanje industrije i gradova, rekreaciju, turizam, kao i osvežavanje vode u prirodnim jezerima Palić i Ludoš.
Kota projektovanog dna kanala se kreće u intervalu od 103,8 mnm do 103,04 mnm, a niveleta kanala od 105,90 mnm do 104,30 mnm.
Izrada kanalskog profila u useku projektovana je u dve faze:

1. Široki iskop u gornjim delovima profila buldožerom i transport na jednu ili obe strane kanala na privremenu deponiju;
2. Iskop donjih delova profila kanala hidraulik bagerima i odlaganje na privremenu deponiju na jednu ili obe strane kanala.

Iskopani materijal se direktno utovara u kupere bagerom, prevozi do mesta ugradnje i istovara kipovanjem.
Transport materijala za izradu kanala u nasipu obavlja se kiperima sa najbliže deonice kanala u useku.
Oblaganje kanala se vrši postavljanjem geotekstila. Preko prethodno postavljenog geotekstila postavlja se vodonepropusna HDPE folija.

Izlaz za životinje se formira od GeoWeb mreže postavljene preko vodonepropusne obloge kanala.
U ovom radu predstavljeno je detaljno planiranje iskopa II deonice magistralnog kanala Orom-Čik-Krivaja, kroz dinamički plan, pomoću gantograma. Proračun je urađen na osnovi zapremine iskopa kanala. Izrada gantograma sa svim potrebnim podacima urađena je primenom softverskog paketa „Microsoft Project 2016“.
Kao najbolja, usvojena je varijanta u kojoj se iskop profila kanala vrši iz tri pozicije sa paralelizacijom radova. Prva pozicija predstavlja iskop leve obale, druga pozicija iskop desne obale I treća pozicija predstavlja iskop dna kanala. Sve tri pozicicije počinju istovremeno.
Kao dan početka rada usvojen je 04.03.2019. godine, a kraj radova predviđen je za 03.09.2019. godine. Usvojeno je da se radi šest radnih dana u nedelji, sa dvanaestočasovnim radnim vremenom. Radovi se neće izvoditi u zimskim mesecima (od novembra do marta).
Izvođačka firma obezbedila je radnu snagu, kao i mehanizaciju. Deo mehanizacije izvođačka firma poseduje, a one mašine koje ne poseduje je dužna da nabavi.
Usvojena je odgovarajuća mehanizacija koja je potrebna da se izvrše radovi, kao i broj mašina koji je potreban za ceo projekat.
Završetak radova omogućiće navodnjavanje oko 15.500 ha poljoprivrednih površina.

10. LITERATURA

[1] Studija opravdanosti i studije uticaja na životnu sredinu magistralnog kanala Orom-Čik-Krivaja (deonica II) podsistema Tisa-Palić.
[2] Mehanizacija u građevinarstvu, Ratko S. Ćulibrk, Momir M. Plavšić, Univerzitet u Novom Sadu, Građevinski fakultet u Subotici, 2007. godine.
[3] Tehnologija i organizacija građenja, Milan Trivunić, Zoran Matijević, FTN izdavaštvo, Novi Sad, 2009. godine.

Kratka biografija:

Branka Kurćubić rođena je u Čačku 1987. godine. Diplomirala je na Fakultetu tehničkih nauka 2016. godine, na smeru hidrotehnika. Master rad na Fakultetu tehničkih nauka u Novom Sadu iz oblasti Tehnologija i organizacija građenja, odbranila je 2018. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 658.7
DOI: https://doi.org/10.24867/01DS01Kosanin

KRITIČNA TRANSPORTNA INFRASTRUKTURA I KRIZNI MENADŽMENT

CRITICAL TRANSPORT INFRASTRUCTURE AND CRISIS MANAGEMENT

Verica Košanin, *Fakultet tehničkih nauka, Novi Sad*

Oblast - SAOBRAĆAJ

Kratak sadržaj – *Kritična transportna infrastruktura predstavlja onu infrastrukturu čije bi ometanje ili uništenje imalo značajan uticaj na održavanje ključnih društvenih funkcija. Krizni menadžment je aktivnost neophodna za savladavanje situacija koje ugrožavaju egzistenciju preduzeća i može se definisati kao posebna forma upravljanja preduzećem. Kako bi se što bolje odgovorilo na krizu koja može zahvatiti preduzeće, bitno je razvijati i unapređivati rezilijentnost preduzeća, odnosno sposobnost da se što bolje odreaguje na neku negativnu situaciju. Cilj ovog rada je da se na osnovu odgovarajuće literature izvrši analiza uticaja kritične transportne infrastrukture na odvijanje osnovnih društvenih i privrednih aktivnosti, sa posebnim akcentom na rezilijentnost transportne infrastrukture i krizni menadžment.*

Abstract – *Critical transport infrastructure represents all the infrastructure whose disturbance or destruction would have a significant impact on the maintenance of key social functions. Crisis management is an activity necessary to overcome situations that jeopardize the existence of an enterprise and can be defined as a specific form of corporate governance. In order to better respond to a crisis that can affect the company, it is important to develop and improve the company's resilience, that is, the ability to respond to a better situation in a better way.The aim of this paper is to analyze the impact of critical transport infrastructure on the basic social and economic activities, with particular emphasis on resiliency of transport infrastructure and crisis management, based on appropriate literature.*

Ključne reči: *Vanredne situacije, kritična infrastruktura, kritična transportna infrastruktura, krizni menadžment, rezilijentnost*

1. UVOD

Kvalitetno obavljanje transportne usluge ne bi bilo moguće bez osnovnih elemenata transporta kao što su transportna sredstva i transportna infrastruktura. Vanredne situacije predstavljaju izmenjeno stanje društvene zajednice, izazvano događajima velikih razmera, kojima se paralište funkcionisanje društvenog sistema zemlje ili kada nastanu prirodne i tehničko - tehnološke katastrofe velikog obima koje ugrožavaju život stanovništva i njihovu imovinu, materijalna i kulturna dobra.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Marinko Maslarić, docent.

Funkcionisanje modernog društva, kako u redovnoj tako i vanrednoj situaciji, nije moguće zamisliti bez efikasne zaštite značajnih infrastrukturnih sistema i objekata, te se kao jedan od primarnih i najznačajnijih bezbednosnih izazova novog doba nameće problem zaštite kritičnih infrastruktura. Stoga je uspešna prevencija i upravljanje vanrednim situacijama u direktnoj vezi sa efikasnim sistemom zaštite kritičnih infrastruktura [1]. Kraj svake faze u nekom razvoju, na prelasku iz jedne u narednu fazu, može proizvesti kriznu situaciju. Kriza je nezaobilazni deo razvoja, pa samim tim ni jedno preduzeće ne može da izbegne takve situacije u svom poslovanju. Krizni menadžment obuhvata organizaciju, pripremu, mere i raspoređivanje resursa za savladavanje krize.

2. KRITIČNA INFRASTRUKTURA

2.1. Pojam kritične infrastrukture

Nacionalna bezbednost i bezbednost u opšte, kao veoma važan segment ističu kritičnu infrastrukturu. Infrastruktura se posmatrala kao logistička funkcija kojom se obezbeđuju povoljni uslovi za kvalitetno obavljanje drugih funkcija logističke podrške. Porastom opasnosti od asimetričnih pretnji, naročito terorizma, u savremenim teorijskim analizama, ali i u praksi, sve je prisutniji izraz kritična infrastruktura. Kritična infrastruktura su sistemi, usluge i imovina, fizički ili virtuelni, toliko važni za dobrobit društva da poremećaj ili uništavanje takvih sistema i imovine ozbiljno utiče na zdravlje, sigurnost ili ekonomsko blagostanje građana i na efikasno funkcionisanje vlade [3]. Svaka država ili organizacija mora da definiše i klasifikuje svoju infrastrukturu i odredi koja infrastruktura je kritična. Prema dokumentima Evropske unije kritične infrastrukture moraju biti definisane i navedene u svim zemljama članicama i u svim zemljama koje hoće da postanu članice Evropske unije. Tek definisanjem kritične infrastrukture i određivanjem njenih osnovnih elemenata može se sagledati celokupno stanje mreža kritične infrastrukture i odrediti mere zaštite kritične infrastrukture i načini na koje treba postupati ukoliko dođe do njenog oštećenja.

2.2. Elementi kritične infrastrukture i njihova međusobna zavisnost

Imajući u vidu prethodno navedene definicije kritične infrastrukture, može se konstatovati da kritične infrastrukture u okviru jedne države čine podsisteme glavnog sistema. Da bi neki sistem mogao neometano da funkcioniše, neophodno je da njegovi podsistemi budu međusobno povezani i da su mežusobno zavisni. Elementi kritične infrastrukture su međusobno povezani i kao takvi

moraju dobro funkcionisati na globalnom nivou. Analizirajući različite pristupe definisanja i klasifikovanja kritične infrastrukture, može se zaključiti da ona obuhvata objekte koji omogućavaju: snabdevanje hranom i vodom, poljoprivredu, rad zdravstvene službe i službe hitne pomoći, energiju (električna, nuklearna, gas i nafta, brane), saobraćaj (vazdušni, drumski, železnički, luke, plovne puteve), informacije i telekomunikacije, bankarstvo i finansije, hemijska postrojenja, odbrambenu industriju, pošte i distribuciju roba i nacionalne spomenike i druge kulturne vrednosti.

2.3. Kritična infrastruktura u Republici Srbiji

Republika Srbija ima iskustva sa vanrednim situacijama, pre svega sa onim izazvanim elemetarnim nepogodama. Na teritoriji Republike Srbije u poslednjih deset godina zabeleženo je više od 150.000 požara. Vezano za vanredne situacije na ovoj teritoriji, posebna pažnja se pridaje poplavama (Obrenovac 2014. godine) i zemljotresima (Kraljevo 2009. godine). Zbog prethodno navedenih događaja, Republika Srbija učinila je značajne napore u stvaranju integrisanog sistema zaštite i spasavanja, kako bi se na adekvatan način odgovorilo u uslovima ugrožavanja kritičnih nacionalnih resursa. Polaznu osnovu danas, svakako predstavljaju i Zakon o vanrednim situacijama i niz drugih strategijskih i podzakonskih dokumenata koji u svojoj daljoj razradi moraju da upotpune odgovore na pitanje šta u Republici Srbiji predstavlja kritičnu infrastrukturu, odnosno ko ima nadležnost nad upravljanjem tom kritičnom infrastrukturom.

2.4. Mere zaštite kritične infrastrukture

Porastom opasnosti od brojnih pretnji u savremenim analizama kao i u praksi, postaje sve bitnije zaštititi kritičnu infrastrukturu. Zaštita kritične infrastrukture važna je ne samo zbog oštećenja potencijalne infrastrukture, već i zbog posledica koje ta oštećenja mogu imati na društvo i privredu. Zaštita kritične infrastrukture u vanrednim situacijama se može posmatrati kao deo jedinstvenog procesa prevencije i odgovora na vanrednu situaciju. U tom kontekstu, organizacija uspostavlja, primenjuje i održava procedure za identifikovanje potencijalnih incidenata koji mogu negativno uticati na neku organizaciju, njene aktivnosti, i okruženje. Ove procedure imaju za cilj zaštitu života, imovine, sprečavanje prerastanja incidenta u vanrednu situaciju ili katastrofu, skraćivanje vremena prekida operacija ili aktivnosti organizacije, oporavak najvažnijih aktivnosti organizacije, povratak na redovne aktivnosti i zaštitu reputacije organizacije.

3. KRIZNI MENADŽMENT

3.1. Uvodne napomene

Krize su događaji koji imaju uzrok. Zajednička determinanta svih tih događaja su negativne posledice po čoveka i njegovu okolinu. Svaki događaj ima uzrok nastanka, principe razvoja i način delovanja. Upravljati kriznim situacijama podrazumeva poznavanje ta tri faktora. Privredne kompanije su odavno uvidele značaj kriznog menadžmenta, pa samim tim svaka veća organizacija koja se bavi proizvodnjom robe ili pružanjem nekih usluga, posebnu pažnju posvećuje upravo krizama. Prepoznajući to kao veoma bitan deo celokupne organizacije društvo se konstantno menja ali to ne bi trebalo da ima velike posledice na poslovanje preduzeća. Ukoliko se dešavaju promene u društvu samim tim će doći i do velikih promena u privredi, što znači da se mora obratiti pažnja na sve promene koje mogu negativno ugroziti poslovanje preduzeća. Krizni menadžment obuhvata organizaciju, pripremu, mere i raspoređivanje resursa za savladavanje krize.

3.2. Terminologija

Kriza je uzrokovana malom verovatnoćom i ona se smatra događajima velikog uticaja, koji mogu da utiču na održivost celokupnog sistema. Takođe, kriza se može definisati i kao posledica neočekivanog, nepredvidivog događaja, koju izaziva neki novi sistem koji deluje na već postojeće sisteme. Posmatrajući krizu na ovaj način ne obraća se pažnja na krizu koja može nastati kao kulminacija inkubacija nekih malih događaja koji najčešće bivaju zanemareni a kao posledica njihove inkubacije dolazi do pojave velike krize.
Iako je teško sprečiti krize, njihov uticaj se može smanjiti, a period oporavka može se znatno redukovati ako se efikasno upravlja. Ovo je moguće uspostavljanjem odgovarajuće i napredne strategije upravljanja krizama pre početka krize.

3.3. Krizni menadžment

Krizni menadžment je serija međusobno povezanih ispitivanja vrste kriza i posledica koje mogu predstavljati veliku opasnost za glavne podsisteme preduzeća i shvata se kao aktivnost koja je neophodna za savladavanje situacija koje ugrožavaju egzistenciju preduzeća. Na osnovu realizovanih istraživanja, krizni menadžment (slika 1) podrazumeva četiri osnovne faze [4]:

- faza ublažavanja poznatija kao prevencija se odnosi na preduzete mere za identifikaciju rizika, izbegavanje njihove pojave i smanjenje mogućih negativnih efekata na ljudski život i ličnu svojinu,
- faza pripreme se odnosi na aktivnosti koje su preduzete pri pokretanju događaja koji omogućava kriznim menadžerima i javnosti da budu u mogućnosti da brzo i efikasno reaguju kada se kriza desi,
- faza odazivanja počinje kada se primjenjuju sve pripremne aktivnosti koje su projektovane i obučavane pre pokretanja događaja i
- faza oporavka obuhvata sve aktivnosti koje se sprovode da bi se vratila u normalnu socijalnu i ekonomsku situaciju.

Slika 1: *Faze kriznog menadžmenta, [4]*

3.4. Rezilijentnost i odnos sa kriznim menadžmentom

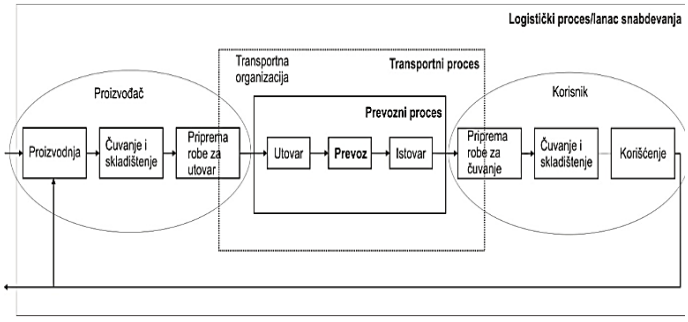
Rezilijentnost je sposobnost da se uvidi mogućnost pozitivnog uticaja na negativnu situaciju, prihvatanje odgovornosti za kriznu situaciju i odabir najboljeg načina reagovanja na nju, procene negativnih posledica koje kriza može da donese i vremena trajanja neke krize. Rezilijentnost (otpornost) je široko rasprostranjen koncept i koristi ga akademija, državni aparat i međunarodni organi zaduženi za sprečavanje katastrofa. Rezilijentnost utiče na sve faze životnog ciklusa rizičnog događaja. Pre pojave krize, cilj kritične infrastrukture je ublažiti pojavu krize i pripremiti se za kritičnu situaciju. U ovoj fazi, u vezi rezilijentnosti, cilj sistema je sprečiti nastajanje incidenata koji bi mogli dovesti do teške krize. Zato kritične infrastrukture imaju za cilj izgradnju otpornih sistema koji mogu izdržati incidente i pripremiti radnike da otkriju signale ranog upozorenja i djeluju što je pre moguće kako bi izbegli dalje štete. Međutim, ne može se uvek izbeći krizna pojava. Kada dođe do krize, cilj kritične infrastrukture je da apsorbuje uticaj i smanji njenu veličinu. Ovo ima veze sa fazom odgovora u fazama kriznog upravljanja. Na kraju, kada situacija bude pod kontrolom, počinje period oporavka. Kritične infrastrukture moraju razviti svoje kapacitete da efikasno deluju od početka u cilju smanjenja posledica. Ova poslednja faza je u potpunosti povezana sa fazom oporavka definisanim unutar ciklusa upravljanja krizama.

4. KRITIČNA TRANSPORTNA INFRASTRUKTURA

Transport ima uticaj na sve druge ljudske aktivnosti u tolikoj meri da se često definiše kao generator ili kao kočnica razvoja. Zavisno od nivoa njegove efikasnosti, pouzdanosti, ekonomičnosti i sigurnosti današnji transportni sistemi su izuzetno kompleksni i zahtevaju sistemski pristup i primenu savremenih tehničko - tehnoloških metoda u njihovom planiranju i upravljanju. Da bi se to postiglo potrebno je pratiti podatke o stanju transportne mreže i definisati specifične indikatore koji će omogućiti merenje efikasnosti transportnih planova.

4.1. Definisanje transportnog sistema

Transport predstavlja jednu od najvažnijih logističkih aktivnosti, pre svega zbog visokih troškova koje sa sobom nosi, a potom i zato što zauzima početno i krajnje mesto u proizvodnom procesu. Slika 2 prikazuje položaj transportnog procesa unutar lanca snabdevanja.



Slika 3: *Mesto transporta u lancu snabdevanja, [2]*

Transportni sistem je veoma složen sistem i obuhvata transportna sredstva, postrojenja i transportne relacije na određenom prostoru. Transportom se može uticati na nivo razvoja nekog tržišta i on neposredno utiče na stepen snabdevenosti tržišta robama i na veću prisutnost ljudi na tržištu, bilo u ulozi proizvođača ili potrošača. Funkcija transporta se najčešće nalazi između funkcije prodaje i kupovine. Osnovna uloga transportnih sistema je da organizuju pružanje usluge sa ciljem zadovoljenja potreba objekata transporta za kretanjem.

4.2. Kritična transportna infrastrukture

Kritična infrastruktura se definiše kao vitalni kapacitet i tehnički sistem organizacije koji obezbeđuju široki spektar društvenih aktivnosti, robe i usluga. Kritična infrastruktura je onaj deo nacionalne infrastrukture (odabrane organizacione institucije, strukture, opreme, usluga i sistema), u kojima uništenje ili onemogućavanje, kao rezultat faktora rizika mogu izazvati opasnost ili poremećaj političkog i ekonomskog delovanja ili izazvati pretnje po život i zdravlje. Nacionalna saobraćajna infrastruktura predstavlja kičmeni stub snabdevanja materijalnim dobrima i uslugama na tržištu rastuće konkurencije i distribuirane ekonomije. Kvalitet i efikasnost ove infrastrukture, preko njene sposobnosti da obezbedi mobilnost i pristup ljudima, uslugama i resursima utiče na kvalitet života i rast ekonomije. Elementi kritične infrastrukture, poput magistralnih pravaca, centara prerade, mesta konekcije sa drugim operaterima u zemlji i inostranstvu kao i sa infrastrukturnim sistemima drugih sektora oduvek su bili vitalne komponente koje doprinose javnoj bezbednosti i nacionalnim ekonomskim aktivnostima. Definisana kritična infrastruktura u transportu je od velike važnosti za državu. Poremećaj nekog elementa kritične saobraćajne infrastrukture negativno se odražava na funkcionisanje celokupnog transportnog sistema. Posledice koje mogu da donesu poremećaji kritične saobraćajne infrastrukture su kompleksni i znatno ih je teže otkloniti.

4.3. Osnovna evropska transportna infrastruktura i kritičnost

Savremeni tokovi robe zahtevaju kombinovanje različitih vidova transporta i razvijenu saobraćajnu infrastrukturu, što je dovelo do znatnog razvoja u ovoj oblasti. Transportna ponuda je takva da transportno tržište danas predstavlja jedno od najrazvijenijih pojedinačnih tržište robe i usluge a transportna potražnja je samo dokaz toga da tržište sve više raste i da se sve više razvija. Kada se govori o značaju evropske kritične transportne infrastrukture, neophodno je spomenuti transevropsku transportnu mrežu. Transevropska transportna mreža doprinosi održivom i multimodalnom razvoju transporta i uklanjanju uskih grla. U ovom smislu, transportne mreže igraju značajnu ulogu u obezbeđivanju održive mobilnosti, kombinujući evropsku konkurenciju sa staranjem o svojim građanima, a u isto vreme obezbeđuju prevoz robe i putnika u Evropi. Najveća slabost sadašnjeg transportnog sistema Evropske unije je nedostatak veza između pomorskog, kopneno-vodnog i železničkog transporta. Vodni transport je i pored blagih pozitivnih pomaka još uvek zapostavljen i pored ekonomske i ekološke povoljnosti (naročito u odnosu na drumski prevoz).

5. UNAPREĐENJE REZILIJENTNOSTI TRANSPORTNE INFRASTRUKTURE

5.1. Konceptualni okvir za izgradnju rezilijentnosti sistema

Usled pojave različitih kriznih događaja, sa negativnim posledicama na blagostanje, ekosistem, ekonomski razvoj i društvo u celini, razvijanje i jačanje koncepta rezilijentnosti je od suštinskog značaja. Upravo koncept rezilijentnosti podrazumeva razumevanje prirode rizika, razvijanje spremnosti da se izbegnu ili ublaže negativne posledice istih, uz korišćenje potencijalnih šansi za razvoj. Rezilijentnost je stalni proces prilagođavanja novonastalim uslovima koji se sastoji u sticanju sve veće i šire kompetencije za reagovanje na krize. U značajnoj je sprezi sa opštim razvojnim procesima, odnosima sa značajnim i specifičnim životnim okolnostima preduzeća. U literaturi je prisutan određen broj različitih konceptualnih okvira za definisanje rezilijentnosti sistema. Tako, na primer postoji okvir koji se sastoji od trinaest indikatora rezilijentnosti grupisanih u tri kategorije: rukovodstvo i kultura, mreža, i spremnost na promene. Po sličnom prinicpu je definisan i program izgradnje rezilijentnosti baziran na osam ključnih atributa rezilijentnosti: svest, agilnost i fleksibilnost, spremnost na promene, znanje, integracije, kultura i vrednosti, liderstvo i komunikacije. Predloženi okviri fokusirani su na organizaciono upravljanje bez pružanja značajnih informacija o drugim dimenzijama rezilijentnosti kao što su: tehničke, ekonomske i socijalne [4].

5.2. Konceptualni okvir za definisanje rezilijentnosti kritične infrastrukture

Okvir rezilijentnosti za kritične infrastrukture sastoji se od dva tipa rezilijentnosti: unutrašnja rezilijentnost i spoljašnja rezilijentnost. U okviru svakog tipa rezilijentnosti, bilo je nekoliko identifikovanih dimenzija rezilijentnosti. Da bi se poboljšale ove dimenzije rezilijentnosti, potrebno je definisati skup rezilijentnih politika i propisa. One su usko povezane sa opštim upravljanjem kritičnom infrastrukturom, kako bi se olakšala njihova implementacija.

Potrebno je dobro razgraničiti vrste i dimenzije rezilijentnosti. Razlikuju se spoljašnja i unutrašnja rezilijentnost i četiri dimenzije rezilijentnosti: tehnička, organizaciona, ekonomska i socijalna rezilijentnost. U skladu sa tim došlo je do podele rezilijentnosti tako da se unutrašnja rezilijentnost deli na tri dimenzije: tehničku, organizacionu i ekonomsku, dok se spoljna rezilijentnost sastoji iz četiri dimenzije: tehničke, organizacione, ekonomske i socijalne.

6. ZAKLJUČNA RAZMATRANJA

Kako bi se sagledao stepen razvijenosti nekog preduzeća, neke države ili regiona najčešće se posmatra transportna infrastruktura. Na razvoj transporta i transportne infrastrukture utiče mnogo faktora. Pitanje kritične transportne infrastrukture je postalo veoma važno zato što poremećaj ili uništenje elemenata ili kompletne kritične infrastrukture može da rezultira ozbiljnim posledicama kao što su oštećenje ljudskog zdravlja, smrtni slučajevi, imovinska šteta, uništavanje životne okoline i ugrožavanje poslovanja različitih privrednih preduzeća.

Uticaj kritične transportne infrastrukture i kriznog menadžmenta na životnu okolinu i poslovanje preduzeća u našim uslovima poslovanja može se sagledati i na primeru vanrednih situacija u Srbiji. Zemljotres koji je 2011. godine pogodio Kraljevo, uništio je veliki broj prilaznih puteva, objekata za stanovanje, društvenih i privrednih objekatai za samo nekoliko sekundi izazvao opšti kolaps na teritoriji čitavog grada. U tom slučaju kritičnu transportnu infrastrukturu su predstavljale sve saobraćajnice tj. celokupna putna mreža, mostovi, aerodrom u Lađevcima, železničke pruge.

Pre toga nije bilo sličnih vanrednih situacija, pa kritična infrastruktura nijednom nije definisana kao takva. Da se postupilo suprotno i da je postojao plan elemenata kritične infrastrukture i razvijena podloga za krizni menadžment došlo bi do mnogo lakšeg suočavanja sa posledicama. Krizni menadžment je obavio svoj posao i u tom slučaju, ali tek nakon što je došlo do štetnog događaja. Da je postojao plan kriznog menadžmenta i delovanja u vanrednim situacijama na ovom području velika je verovatnoća da bi se mogle predvideti neke situacije. Isti slučaj je i kada se govori o poplavama, koje su već nekolko puta bile kobne po stanovništvo Republike Srbije.

Čim je došlo do uništavanja osnovne kritične infrastrukture, to automatski znači i da je došlo do uništavanja privrede. Krize koje se događaju mogu biti i ekonomskog karaktera i često je moguće čuti informacije o privrednom udaru. Struktura kapitala i finansijska politika imaju značajnu ulogu u procesu izbora strategije oporavka preduzeća. Krize poslovanja preduzeća su neželjene pojave ali su nekada neminovne. Većina kriznih situacija se može predvideti tako da je moguće izvršiti pripreme, iako je njen intenzitet nezavisna kategorija.

7. LITERATURA

[1] Mićović, M.: Bezbednosni aspekti funkcionisanja kritične infrastrukture u vanrednim situacijama, Beograd, 2016.

[2] Maslarić, M.: Slajdovi sa predavanja iz predmeta oblikovanje logističkih procesa u lancima snabdevanja , Fakultet tehničkih nauka Novi Sad, 2017.

[3] Škero,M.,Ateljević V.: Zaštita kritične infrastrukture i osnovni elementi usklađivanja sa direktivom Saveta Evrope 2008/114/ES, 2015.

[4] Sarriegi, M.,Hernantes, J.: Resilience Framework for Critical Infrastructures, Universidad de Navarra, Donostia, San Sebastian, 2013.

Kratka biografija:

Verica Košanin je rođena u Kraljevu 1993. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Saobraćaja odbranila je 2016. god.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 336.71:004 004.738.5:339
DOI: https://doi.org/10.24867/01DS02Vujanic

ELEKTRONSKE FINANSIJSKE USLUGE U POŠTI

ELECTRONIC FINANCIAL SERVICES IN POST

Ognjen Vujanić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – Poštanski saobraćaj i telekomunikacije

Kratak sadržaj – *U ovom radu su definisani standardi kvaliteta, tendencije i pravci budućeg razvoja elektronskih finansijskih usluga. Takođe, napravljen je presjek stanja ovih usluga i analizirane su alternative kao prijedlozi za unapređenje istih.*

Ključne reči: *Elektronske finansijske usluge, e-fakturisanje, standardi kvaliteta, mjerenje kvaliteta, direktive*

Abstract – *This paper defines standards of quality, tendencies and directions for the future development of electronic financial services. Also, a cross-section of these services was made and alternatives were analyzed as suggestions for improving them.*

Keywords: *Electronic financial services, e-invoicing, quality standards, quality measurement, directive,*

1. UVOD

Finansijske usluge su od svojih samih početaka predstavljale važan činilac platnog sistema jedne države a i važan činilac u funkcionisanju cjelokupnog društva. U poštanskom saobraćaju finansijske usluge zauzimaju jedno od osnovnih mjesta, takoreći predstavljaju jedan od stubova nosioca same pošte.

Elektronske finansijske usluge su nadgradnja i poboljšanje finansijskih usluga, u elektronskom obliku. Elektronske finansijske usluge su u centru interesovanja ne samo poštanske struke već i mnogih drugih oblasti društva. Ove usluge su već u velikoj mjeri prisutne u funkcionisanju poštaskih operatora širom svijeta.Cilj ovog rada je bilo definisanje elektronskih finansijskih usluga, prikaz standarda kvaliteta, načina mjerenja kvaliteta ovih usluga, zatim kroz prikaz stanja elektronskih finansijskih usluga u najrazvijenijim poštanskim upravama pokazati pravce budućeg razvoja ovih usluga. Takođe, cilj ovog rada je bio da prikaže probleme uključivanja pošte na nova, tehnološkim razvojem generisana, tržišta elektronskih usluga te da ponudi alternativna rješenja kao odgovor.
Metode koje su korišćene u izradi ovog rada podrazumjevaju teorijska istraživanja, korišćenje različite stručne literature te metode deskripcije, analize i komparacije.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Dragana Šarac, vanr. prof.

2. ELEKTRONSKE FINANSIJSKE USLUGE

Elektronske finansijske usluge u poštama predstavljaju odgovor pošte na promjene u društvu pogotovo na promjene i razvoj tehnologijen, posebno razvoj interneta i telefonije. Takođe uvođenje i razvoj ovih usluga predstavlja i odgovor na nove načine plaćanja koji se javljaju i koji u sve većoj mjeri zamjenjuju plaćanje gotovinom. Ove usluge se uglavom baziraju na već poznatim platformama postojećih usluga koje se razvijaju u pravcu njihove elektronske verzije odnosno omogućavanju kompatibilnosti sa novim tehnologijama. Pa je UPU elektronske finansijske usluge klasifikovao na sledeći način [1]:

- Elektronsko upravljanje nalogom,
- Elektronska uputnica,
- Onlajn plaćanje računa,
- Upravljanje računima,
- Elektronsko plaćanje računa za vodu,
- Elektronsko plaćanje računa za struju,
- Elektronsko plaćanje računa za telefon,
- Elektronski transfer novca [1].

2. MJERENJE KVALITETA ELEKTRONSKIH FINANSIJSKIH USLUGA

2.1 Dostupnost prema vrsti usluge

Dostupnost prema vrsti usluge predstavlja procenat jedinica poštanske mreže koje pružaju sve vrste finansijskih usluga. Kao referenca za ovo mjerenje se uzima Preambula sporazuma o platnim uslugama.
Namjena ovog mjerenja jeste da omogući vladama potpis-nicama Preambule sporazuma o poštansko-finansijskim uslugama (člana 10.1) da utvrde zastupljenost jednica koje pružaju sve usluge i da geografski precizira njihov položaj te na taj način posmatra evoluciju istih.
Ovim mjerenjem se omogućava Svjetskom poštanskom savezu da utvrdi stepen implementacije u svakoj od zemalja članica i na taj način da precizno definiše razvoj svetske poštanske mreže.

Ova mjerenja se vrše na godišnjem nivou a rezultati se prikazuju u procentima [2].

Dostupnost prema vrsti usluge (QSM_1) se izračunava na način prikazan u izrazu (1):

$$QSM_1 = \frac{X_1}{Y_1}\ x\ 100 \qquad (1)$$

X_1 – Broj jedinica poštanske mreže koje pružaju uslugu
Y_1 – Ukupan broj jedinica poštanske mreže koje pružaju finansijske usluge

Prema Savjetu za poštansku eksploataciju Svjetskog poštanskog saveza cilj je dostići 80% kao rezultat mjerenja [2].

2.2 Efikasnost pružanja usluga

Efikasnost pružanja usluge (QSM_2) predstavlja procenat uplaćenih novčanih uputnica. Referenca za ovo mjerenje je rezolucija C 74/2008 usvojena na 24. Kongresu Svjetskog poštanskog saveza. Omogućava vladama potpisnicama da utvrde ukupnu efikasnost pružanja usluga. Pruža mogućnost Svjetskom poštanskom savezu da procjeni kvalitet pružanja usluga na globalnom nivou.
Ova mjerenja se vrše na mjesečnom nivou a rezultati se prikazuju u procentima.
Efikasnost pružanja usluge (QSM_2) se izračunava na način prikazan u izrazu (2):

$$QSM_2 = \frac{X_2}{Y_2}\ x\ 100 \qquad (2)$$

X_2 – Broj naloga uplaćenih od države prijema
Y_2 – Broj naloga izdatih od partnerskih država države prijema

Analiza pokazatelja:

- Prema vrsti usluge i prema državi
- Na bilateralnoj osnovi, kao pod-pokazatelj gore navedenog:
  - Nalog izdat od države „A“ i isplaćen od države „B“.
  - Nalog izdat od države „B“ i isplaćen od države „A“.

Efikasnost pružanja usluge (QSM_{2a} i QSM_{2b}) se izračunava na način prikazan u izrazima (3) i (4):

$$QSM_{2a} = \frac{X_{2a}}{Y_{2a}}\ x\ 100 \qquad (3)$$

X_{2a} – Broj naloga izdatih od države "A" i isplaćenih od države "B"
Y_{2a} – Broj naloga izdatih od države "A"

$$QSM_{2b} = \frac{X_{2b}}{Y_{2b}}\ x\ 100 \qquad (4)$$

X_{2b} – Broj naloga izdatih od države "B" i isplaćenih od države "A"
Y_{2b} – Broj naloga izdatih od države "B"

Prema Savjetu za poštansku eksploataciju Svjetskog poštanskog saveza cilj je dostići 90% kao rezultat mjerenja [2].

2.3 Razvoj usluga

Razvoj usluga(QSM_3) obuhvata procenat povećanja broja izdatih novčanih naloga. Omogućava vladama potpisnicama da utvrde razvoj usluga na osnovu dva prethodna pokazatelja (dostupnosti i efikasnosti). Takođe omogućava Svjetskom poštanskom savezu da na nivou država članica i na globalnom nivou posmatra evoluciju ovih vrsta usluga. Ova mjerenja se vrše na mjesečnom nivou i rezultati se prikazuju u procentima. Razvoj usluga (QSM_3) se izračunava na način prikazan u izrazu (5):

$$QSM_3 = \frac{X_3}{Y_3}\ x\ 100 \qquad (5)$$

X_3 – Broj naloga izdatih za mjesec dana (m) u n – toj godini umanjen za broj naloga izdatih za mjesec dana (m) u n – 1 god.

Y_3 – Broj naloga izdatih za mjesec dana (m) u n – 1 godini

Izdati nalozi predstavljaju naloge koji su primljeni od strane izabranog operatora na prijemu (u taj broj ne ulaze nalozi koje su bilo koji način odbijeni). Prema Savjetu za poštansku eksploataciju Svjetskog poštanskog saveza cilj je dostići od 5% do 10% kao rezultat mjerenja [2].

2.4 Vremenska dostupnost platnih naloga

Vremenska dostupnost platnih naloga (QSM_4) predstavlja procenat platnih naloga koji su dostupni u rokovima koje je predvidio Savjet za poštansku eksploataciju. Vrijeme za obradu platnog naloga se računa kad je poruka o prijemu registrovana od strane izabranog operatora koji prima platni nalog. Usklađenost vremena predstavlja usklađivanje vremena izdavanja naloga od izabranog operatora i vremena isplaćivanja od izabranog operatora. Ovo mjerenje omogućava vladama država koje izdaju i primaju platne naloge da provjeravaju vrijeme prenosa te da ga upoređuju sa kriterijumom kojeg je postavio Savjet za poštansku eksploataciju. Za Svjetski poštanski savez ovo predstavlja mogućnost praćenja homogenosti tehničkih parametara između operatora, te mogućnost planiranja korektivnih mjera po potrebi. Ova mjerenja se vrše na mjesečnom nivou i rezultati se prikazuju u procentima. Vremenska dostupnost platnih naloga (QSM_4) se izračunava na način prikazan u izrazu (6):

$$QSM_4 = \frac{X_4}{Y_4}\ x\ 100 \qquad (6)$$

X_4 – Broj platnih naloga registrovanih u predviđenom roku
Y_4 – Broj izdatih platih naloga

Izdati nalozi predstavljaju naloge koji su primljeni od strane izabranog operatora na prijemu (u taj broj ne ulaze nalozi koje su bilo koji način odbijeni) [2].

2.5 Obrada reklamacija na vrijeme

Obrada reklamacija na vrijeme (QSM_5) predstavlja procenat obrađenih reklamacija u roku od 10 dana. Referenca za ovo mjerenje je Sporazum o platnim uslugama, član 19., član RP 1901 i član RP 1902.
Ovo mjerenje omogućava vladama zemalja potpisnica Sporazuma o platnim uslugama da utvrde da li se reklamacije obrađuju u roku koji je ovaj sporazum predvidio (10 dana).
Ovo omogućava Svjetskom poštanskom savezu da prati usklađenost obrade reklamacija na državnom i na globalnom nivou. Ova mjerenja se vrše na mjesečnom

nivou i rezultati se prikazuju u procentima. Obrada reklamacija na vrijeme (QSM_5) se izračunava na način prikazan u izrazu (7):

$$QSM_5 = \frac{X_5}{Y_5}\ x\ 100 \qquad (7)$$

X_5 – Broj reklamacija obrađen u roku postavljenom od strane UPU
Y_5 – Ukupan broj reklamacija [2].

2.6 Zadovoljstvo korisnika

Zadovoljstvo korisnika (QSM_6) predstavlja obim reklamacija kao procenat od ukupnog broja izdatih naloga.
Referenca za ovo mjerenje je Sporazum o platnim uslugama, član 19., član RP 1901 i član RP 1902. Vladama država potpisnica ovo mjerenje pruža praćenje sveobuhvatnog kvaliteta, kvaliteta svake usluge pojedinačno te praćenje ulaznog i izlaznog saobraćaja.
Omogućava Svjetskom poštanskom savezu da prati nekvalitetne stavke na državnom i globalnom nivou na osnovu ciljeva koje je postavio Savjet za poštansku eksploataciju.
Ova mjerenja se vrše na mjesečnom nivou i rezultati se prikazuju u procentima. Za dolazeće naloge (QSM_{6a}) izračunavanje se vrši na način prikazan u izrazu (8):

$$QSM_{6a} = \frac{X_{6a}}{Y_{6b}}\ x\ 100 \qquad (8)$$

X_{6a} – Broj reklamacija
Y_{6b} – Broj izdatih platnih naloga

Za odlazeće naloge (QSM_{6b}) izračunavanje se vrši na način prikazan u izrazu (9):

$$QSM_{6b} = \frac{X_{6a}}{Y_{6b}}\ x\ 100 \qquad (9)$$

X_{6a} – Broj reklamacija
Y_{6b} – Broj izdatih platnih naloga [2].

2.7 Vrijeme izdavanja platnog naloga

Vrijeme izdavanja platnog naloga (QSM_7) predstavlja procenat platnih naloga poslatih na vrijeme na odredište od izdavanja sve do prenosa spisa. Omogućava izabranom operatoru mjerenje udjela transakcija poslatih u fiksnom vremenu svakom od partnera.
Ovo mjerenje omogućava UPU da izmjeri udio transakcija poslatih u fiksnom vremenu. Ova mjerenja se vrše na mjesečnom nivou i rezultati se prikazuju u procentima. Vrijeme izdavanja platnog naloga (QSM_7) se izračunava na način prikazan u izrazu (10):

$$QSM_7 = \frac{X_7}{Y_7}\ x\ 100 \qquad (10)$$

X_7 – Broj platnih naloga poslatih u postavljenim rokovima odredišnog IO
Y_7 – Broj platnih naloga poslatih od IO slanja prema IO prijema
IO – Izabrani operator [2].

2.8 Ukupno vrijeme isplate

Ukupno vrijeme isplate (QSM_8) predstavlja procenat platnih naloga koji su isplaćeni u okviru vremenskih rokova predviđenih u standardu kvaliteta uplata i isplata po platnom nalogu. Omogućava izabranom operatoru a i Svjetskom poštanskom savezu mjerenje stvarnog nivoa isplata u vremenskokm okviru koji je predviđen standardom. Ova mjerenja se vrše na mjesečnom nivou i rezultati se prikazuju u procentima. Ukupno vrijeme isplate (QSM_8) izračunava se na način prikazan u izrazu (11):

$$QSM_8 = \frac{X_8}{Y_8}\ x\ 100 \qquad (11)$$

X_8 – Broj platnih naloga poslatih u postavljenim rokovima odredišnom IO
Y_8 – Broj platnih naloga poslatih od IO slanja prema IO prijema
IO – Izabrani operator [2].

2.9 Vrijeme prenosa

Vrijeme prenosa (TTM_1) predstavlja procenat platnih naloga koji se prenesu u predviđenom vremenskom roku od strane izabranog operatora slanja. Ovaj pokazatelj će korisnicima pružiti cjelokupni kvalitet mreže i sistema. Ova mjerenja se vrše na mjesečnom nivou i rezultati se prikaziju u procentima.

Vrijeme prenosa (TTM_1) se izračunava na način prikazan u izrazu (12):

$$TTM_1 = \frac{T_1}{T_2}\ x\ 100 \qquad (12)$$

T_1 – Broj platnih naloga primljenih u postavljenim rokovima kod odredišnog IO
T_2 – Broj platnih naloga poslatih od IO slanja prema IO prijema
IO – Izabrani operator [2].

3. DIREKTIVA O PLATNIM USLUGAMA U EVROPSKOJ UNIJI

Direktiva o platnim uslugama je stupila na snagu 1. novembra 2009. godine sa ciljem obezbjeđivanja istih pravila u oblasti elektronskog plaćanja u svim državama članicama Evropske unije i u Lihtenštajnu, Norveškoj i Islandu. Na ovaj način je korisnicima omogućeno plaćanje u čitavoj Evropi, lako i sigurno na način kao što su to vršili u svojim matičnim zemljama. Ova direktiva daje detaljne informacije svim vrstama plaćanja i čini plaćanje bržim i jednostavnijim. Ona omogućava novim platim institucijama kao što su institucije koje se bave prenosom novca i ritejlom te operatorima mobilne telefonije da pored banaka pružaju platne usluge. Kao pružaoci platnih usluga zajedno sa bankama se navode još i platne institucije i druga platna tijela.

Direktiva pokriva sve vrste elektronskih i bezgotovinskih plaćanja počev od kreditnih transfera, direktnih zaduženja, plaćanja karticom (uključujući i plaćanja kreditnom karticom) i novčanih doznaka za mobilna i onlajn plaćanja. Plaćanja u kešu i putem čeka nisu pokrivena ovom direktivom. Plaćanja u bilo kojoj evropskoj valuti ne samo u evru su pokrivena direktivom sve dok pružaoci

platih usluga imaju uplatioca i primaoca iz jedne od navedenih zemalja. Direktiva o platnim uslugama čini informacije o plaćanju jasnijima na više načina. Pružalac platih usluga mora korisnicima dati ključne informacije koje su im potrebne prije i nakon plaćanja. Ovo podrazumijeva pružanje informacija, prije korišćenja usluge plaćanja, koje se tiču detaljnih uslova, uključujući informacije o pružaocu platnih usluga, karakteristika usluge plaćanja, vremena obrade, ograničenja trošenja, troškova i prava na povrat.

O svakoj promjeni ovih stavki pružalac platih usluga mora obavjestiti korisnika i to dva mjeseca unaprijed. Nakon svakog obavljenog plaćanja korisnici dobijaju informacije o iznosu, datumu i troškovima tako da mogu da provjere ispravnost uplate. U prodavnicama će korisnici moći jasnije vidjeti koliko plaćaju naknade uključujući i onlajn prodavnice.

Takođe, ritejleri mogu da daju popuste korisnicima ako ovi plaćaju na načine koji su jefitiniji po same ritejlere (npr. ako koriste debitne karitice ili kartice ritejlera). Ritejlerima se sa druge strane dopušta i da korisnicima naplate korišćenje platnih načina koji njima ne odgovaraju izuzev ako je to zabranjeno državnim zakonima. Direktiva nudi i načine da se zaštite prava korisika, te daje mogućnost za refundacije [3]:

- Kod neovlaštenog zaduživanja, gdje korinik po primjećivanju istog treba da obavjesti operatora i u roku od najkasnije 13 mjeseci može da dobije naknadu.
- Kod prekomjernog naplaćivanja naknada, ako je koristik primjetio da mu je naplaćena veća naknada nego što je naznačeno, ima mogućnost da ovo prijavi operatoru a ovaj je dužan da isto ispita u roku od dva mjeseca [13].
- Kod pogrešne obrade, korisnik ima pravo da se žali i traži ispravku u roku od 13 mjeseci [3].

4. ZAKLJUČAK

Napredovanjem informacionih tehnologija došlo je do stvaranja novih tržišta a samim tim i tržišta koja se vezuju i za domen poštanskog saobraćaja. Upravo ubrzan razvoj ovih tehnologija je otvorio vrata novim servisima koji trebaju da odgovore potrebama modernog korisnika. Ova nova tržišta i servisi su se nametnuli pošti i uopšteno poštanskom saobraćaju kao veliki izazov a ujedno i kao nova poslovna mogućnost.
Kao što je u ovom radu i prikazano, neke od najrazvijenijih poštanskih uprava nastoje da u što većoj mjeri u svoje poslovanje uvedu servise koji će im omogućiti uključenje u nova e-tržišta. Države i njihovi poštanski operatori su na različite načine pristupali razvoju i uvođenju elektronskih usluga, mada je skoro svima zajedničko to da su svoje servise nastojali digitalizovati i kao elektronske ponuditi svojim korisnicima.
Ono što se nameće kao zaključak jeste upravo potreba za uključivanjem poštanskih operatora u svjetske tokove e-poslovnih i e-tržišnih usluga. Odnosno, uzimanje učešća u ovim tokovima kroz stvaranje strategije zajedničkog i organizovanog dijelovanja u cilju stvaranja jedinstvene mreže.
Upravo stvaranje zajedničkih strategija te zajedničko nastupanje na tržištu elektronskih finansijskih usluga može donijeti poštanskim operatorima prednost u odnosnu na konkurenciju. Ovaj način zajedničkog organizovanja operatora kroz objedinjavanje servisa i mreže, predstavlja jedan od načina da se poštanska djelatnost izbori sa konkurencijom i zahtjevima tržišta, te da kroz unapređenje postojećih i razvoj novih servisa stvori nove izvore prihoda.
Takođe, ovaj način zajedničkog djelovanja na tržištu elektronskih finansijskih uluga i usluga e-poslovanja predstavlja moguć recept i za poslovanje Pošta Srbije. Kao što je u ovom radu i navedeno primjer regionalne saradnje poštanskih operatora iz Srbije, BiH, Hrvatske i Crne Gore u oblasti elektronskih finansijskih usluga već postoji i on treba da posluži kao osnova za unapređenje ovih usluga a samim tim i unapređenje poslovanja ovih operatora.
Ovakav vid regionalne saradnje predstavlja jedan od načina da se pošte pozicioniraju na modernom tržištu, koje se sve brže mijenja razvojem novih tehnologija.

5. LITERATURA

[1] Šarac D., Finansijsko poslovanje u poštanskom saobraćaju, FTN izdavaštvo, Novi Sad, 2014.
[2] http://presentations.upu.int/?repository_id=poc&folder=%252F2014%252FServices%2520financiers%2520postaux%252FForum%252028.03.2014#1 (pristupljeno u junu 2018.)
[3] http://ec.europa.eu/finance/payments/docs/framework/psd_consumers/psd_en.pdf (pristupljeno u junu 2016.)

Kratka biografija:

Ognjen Vujanić rođen je u Prijedoru u Republici Srpskoj 1992. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Poštanski saobraćaj i telekomunikacije – „Elektronske finansijske usluge u pošti“ odbranio je 2018.godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 656.11.021
DOI: https://doi.org/10.24867/01DS04Stokic

ПЛАН ИЗРАДЕ ГЕНЕРАЛНОГ ПРОЈЕКТА ЗА ПАРКИНГ ГАРАЖУ У ПОЖАРЕВЦУ

PLANNING THE REALIZATION OF THE GENERAL PROJECT FOR PARKING GARAGE IN POZAREVAC

Никола Стокић, Предраг Атанасковић, *Факултет техничких наука, Нови Сад*

Област – САОБРАЋАЈ

Кратак садржај – *Основни циљ израде овог рада је планирање и анализа свих неопходних активности и ресурса везаних за израду техничке документације на нивоу генералног пројекта са претходном студијом оправданости, процена трошкова и процена потребног времена за реализацију пројекта уз примену знања и метода из области управљања пројектима.*

Кључне речи: *Управљање пројектима, паркирање и јавне гараже*

Abstract – *The main goal of this work is planning and analysis of all necessary activities and resources related to the preparation of tehnical documentation at the level of the general project with a preliminary feasibility study, costs estimation and time estimation needed for the realization of the project using knowledge and methods from domain of project management.*

Keywords: *Project management, parking and public garages*

1. УВОД

У свакодневном привредном и друштвеном животу термин пројекат је у веома широкој употреби. Било да се ради о освајању новог тржишта, увођењу новог информационог система или нове организације, развоју новог производа, реконструкцији производног погона, изградњи новог објекта, увек се говори о реализацији одређеног пројекта. Пројекат се најчешће дефинише као сложени и непоновљиви подухват који се предузима у будућности да би се постигли циљеви у предвиђеном времену и са предвиђеним трошковима.

Сложеност савремених пројеката изражава се кроз велики обим и широку структуру ових подухвата, дуго време трајања, огроман буџет, велики број учесника у реализацији пројекта и друге параметре.

Поред свих знања из области управљања пројектима, квалитетним и добрим ресурсима, одговарајућем буџету пројекта и реално одмереном времену који је потребан за реализацију једног пројекта, неопходно је и коришћење софтверских алата за управљање пројектима [1].

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Предраг Атанасковић.

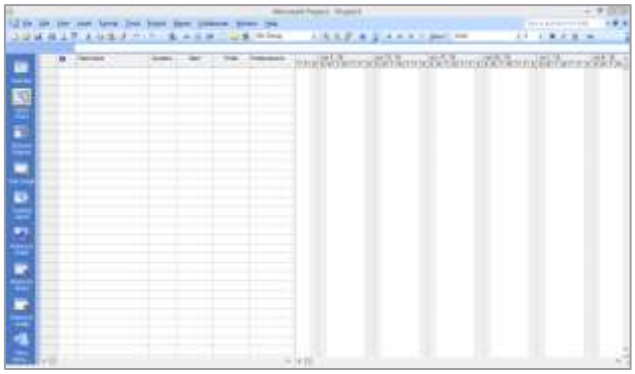
2. УПРАВЉАЊЕ ПРОЈЕКТИМА ПРИМЕНОМ СОФТВЕРСКОГ ПАКЕТА MS PROJECT

Софтверски пакет Microsoft Office Project (MSP) је саставни део окружења Microsoft Office. MS Project Standard је desktop апликација за управљање пројектима заснована на оперативном систему Microsoft Windows и уједно основни софтверски пакет из породице Microsoft Office Project.

MSP има техничке могућности које се пре свега огледају у следећем:

- може да прати већи број пројеката у истом тренутку,
- праћење свих информација које корисник сакупља о предметном пројекту,
- прати, контролише и упозорава корисника о трајању активности и прекорачењима,
- прати, контролише и упозорава корисника о трошковима,
- даје могућност прављења пројектног плана у стандардним добро дефинисаним форматима,
- даје могућност повезивања активности и ресурса на ефикасан начин,
- даје могућност кориснику пакета задржавање контроле над пројектом који се реализује или је у процесу димензионисања,
- даје могућност повезивања са осталим окружењем Microsoft Office пакета [2].

На слици 1. је приказан основни изглед радног простора MS Project-a.



Слика 1. *Основни изглед радног простора у MS Project*

3. АКТИВНОСТИ НА ПРОЈЕКТУ

Задаци - активности су основ за изградњу сваког пројекта. Активности описују рад на неком пројекту помоћу појмова: рок, трајање и потребни ресурси. Активност на пројекту представља задатак који је описан ресурсом који извршава дату активност, временом потребним да се та активност реализује и трошковима. Разликују се активности које се баве израдом техничке документације и пројеката који се баве реализацијом према техничкој документацији.

Основна подела активности:

- сумарни задаци - активности,
- подактивности,
- кључни задаци на пројекту - MILESTONE [2].

Сумарни задаци на пројекту јесу они задаци на пројекту који се реализују у финкцији одређених временских рокова и који су описани ценама, извођачима задатака и временима трајања. Цео пројекат се састоји од већег или мањег броја сумарних активности, а сумарне активности се састоје од већег броја појединачних активности на пројекту које представљају подактивности тих сумарних активности. Кључни догађаји (MILESTONE) су они догађаји на пројекту који су од велике важности за реализацију неког пројекта [3]. У табели 1. је приказана листа задатака - активности које је потребно реализовати у оквиру генералног пројекта са претходном студијом оправданости за изградњу паркинг гараже.

Табела 1. *Приказ листе задатака - активности на пројекту*

1. УВОД И ОСНОВНЕ ПОСТАВКЕ
1.1. Циљеви пројекта
1.2. Поставке пројекта
2. УТВРЂИВАЊЕ И АНАЛИЗА СТАЊА ПАРКИРАЊА
2.1. Реално стање паркирања
2.2. Утврђивање захтева паркирања и потражње у граду
2.3. Утврђивање политике паркирања у граду
2.4. Одређивање приступа истраживању проблема паркирања
3. ТЕХНИЧКЕ АНАЛИЗЕ
3.1. Грађевинско - техничка решења
3.2. Саобраћајна технологија
3.3. Графички приказ просторног решења
3.4. Структура трошкова изградње гараже
3.5. Динамика изградње
3.6. Експлоатација и одржавање
4. АНАЛИЗА ТРОШКОВА
4.1. Трошкови експлоатације
4.2. Трошкови одржавања
5. ПРЕТХОДНА СТУДИЈА ОПРАВДАНОСТИ
5.1. Анализа постојећег стања
5.2. Тржишни аспект - анализа и пројекција
5.3. Приказ техничко - технолошких решења у генералном пројекту
5.4. Анализа набавног тржишта
5.5. Просторни аспект
5.6. Претходна анализа утицаја на животну средину
5.7. Финансијска анализа и оцена
5.8. Друштвено - економска анализа и оцена
5.9. Анализа осетљивости и ризика инвестирања
5.10. Предходна анализа извора финансирања и финансијских обавеза
5.11. Претходна анализа организационих и кадровских могућности
6. ЗАКЉУЧНА РАЗМАТРАЊА
6.1. Закључци и препоруке

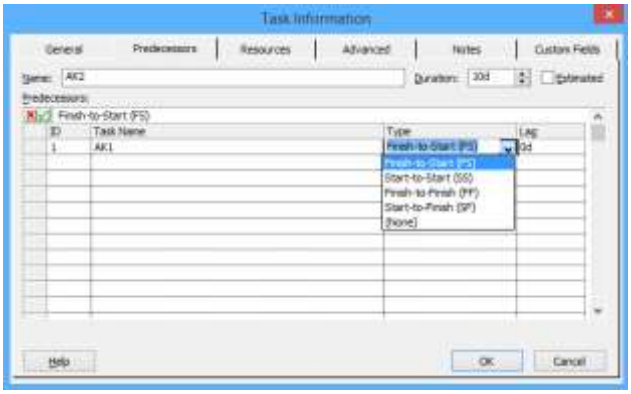
4. ПОВЕЗИВАЊЕ ЗАДАТАКА - АКТИВНОСТИ НА ПРОЈЕКТУ

Сваки пројекат састоји се од низа задатака – активности. Сви планирани задаци морају се извршити у оквиру реализације неке пројектне активности. Активности су осим временске одређености - дужине трајања, одређени и начином повезивања, све у склопу техничко-технолошких планова, врсте пројекта и расположивих ресурса у датом тренутку.

Постоје четири основна типа међусобних релација међу активностима на неком пројекту:

1. FINISH TO START (FS) - заврши да би почео,
2. START TO START (SS) - почни да би почео,
3. FINISH TO FINISH (FF) - заврши да би завршио,
4. START TO FINISH (SF) - почни да би завршио [3].

На слици 2. је приказан начин повезивања активности у Microsoft Project-у.



Слика 2. *Повезивање активности у Microsoft Project-у*

5. КРЕИРАЊЕ БАЗЕ РЕСУРСА НА ПРОЈЕКТУ

Да би се неки пројекат могао реализовати потребно је располагати одговарајућим ресурсима. Сваки пројекат користи ресурсе, или више њих у исто време. Ресурси су углавном ограничени обимом пројекта, буџетом пројекта, расположивим кадровима и они увек улазе у укупне трошкове пројекта.

У основне ресурсе на пројекту спадају:

- *Људство* - то су кадрови, учесници на пројекту у које спадају: инжењери, специјализовани извршиоци, помоћни извршиоци, менаџери, техничка подршка и сл.
- *Опрема* - обухвата техничку опрему: возила, специјалне машине, специјалне алате, пресе, оплате, скеле, рачунарска опрема и сл.
- *Материјали* - подразумевају се сви материјали који се користе у извођењу и реализацији пројекта (материјали се разликују у зависности од врсте пројекта).
- *Техника* - је техничка подршка везана за рачунарске системе, системе за комуникацију, лабораторије и сл [3].

Додељивање ресурса постављеним задацима – активностима на пројекту врши одговорно лице (project manager), који има довољно знања и искуства да постави саму организацију рада, да процени који су ресурси потребни, да процени која стручна спрема је потребна, колико је потребно времена појединачном ресурсу или групи да постављени задатак - активност реши.

6. ВРЕМЕ ПОТРЕБНО ЗА РЕАЛИЗАЦИЈУ ПРОЈЕКТА

Како би се дошло до потребних времена трајања активности на пројекту, примењена је Делфи метода. Фазе примене Делфи методе су:

1. Припрема анкетног листа
2. Избор експерата који ће оцењивати постављена питања
3. Слање анкетног листа експертима
4. Обрада попуњених анкетних листова
5. Анализа одговора
6. Излазни резултати

Анкетирани експерти су оцењивали потребно време трајања за израду сумарних активности на пројекту имајући у виду и подактивности као и обим послова који се морају реализовати према пројектом задатку. У оквиру истраживања анкетирано је укупно 8 пројектаната.

У табели 2. приказани су систематизовани резултати према добијеним анкетним листовима, где су приказане оцене експерата изражене у броју дана везане за оцену потребног времена за реализацију сумарних активности на пројекту.

Табела 2. *Одговори експерата о потребном броју дана за реализацију сумарних активности*

Сумарне активности на пројекту	Експерти								Средња вредност (дана)
	1	2	3	4	5	6	7	8	
	Трајање у данима								
1. Увод и основне поставке	7	7	8	9	7	7	9	7	7.63
2. Утврђивање и анализа стања паркирања	25	22	24	25	25	26	25	27	24.88
3. Техничке анализе	41	41	40	43	42	43	39	40	41.13
4. Анализа трошкова	10	10	10	9	9	9	10	12	9.88
5. Претходна студија оправданости	75	72	70	71	70	70	70	71	71.33
6. Закључна разматрања	5	5	5	5	5	7	5	5	5.25

7. ТРОШКОВИ ПРОЈЕКТА

Трошкови представљају новчана средства која се улажу у сваки део реализације пројекта, односно исти приказују колико израда техничке документације кошта. При изради техничке документације то су средства за плаћање инжењера, материјална средства, средства за набављање техничке опреме, а са аспекта реализације то су средства намењена за набавку материјала, опреме, радне снаге и сл. [3].

Трошкови могу бити:

директни - они на које се рачуна, изражавају се као:

$$T_d = T_{rada} + T_{opreme} + T_{prostora} \text{ обично су } 90\% > T_d > 95\% \quad [1]$$

индиректни - који се очекују и који се могу са сигурношћу предвидети:

$$T_i = T_{konsultanti} + T_{saradnici} \quad [2]$$

непредвиђени - $T_n \leq 3\%$.

$$T_{projekta} = T_d + T_i + T_n \quad [3]$$

Највеће укупне трошкове на пројекту има сумарна активност Претходна студија оправданости 892.840,00 дин. што је и оправдано с обзиром на број подактивности које обухвата ова сумарна активност. Следећа сумарна активност са највећим укупним трошковима је сумарна активност Техничке анализе који износе 514.800,00 дин. Најмање укупне трошкове има сумарна активност Закључна разматрања 76.200,00 дин.

8. РИЗИЦИ У ПЛАНИРАЊУ РЕАЛИЗАЦИЈЕ ПРОЈЕКТА

Ризик постоји у сваком пројекту, а вероватноћа да ће се одређени ризични догађај и остварити, зависи од његове природе. Ризик као и већина елемената планирања, мења се током напретка пројекта и захтева праћење. Како се пројекат приближава одређеном ризичном догађају, неопходно је преиспитати почетне претпоставке и планове за деловање, а по потреби и извршити и одређена прилагођавања [1].

8.1. Утицај појединачних ризика на показатеље пројекта

У току реализације самог пројекта може доћи и до повећања трошкова ресурса. За људске ресурсе је то цена рада (дин/час), а за опрему може бити повећање или смањење цене изнајмљивања.

Први ризични догађај који је анализиран у оквиру пројекта представља повећање цене свих ресурса за 5%. У табели 3. је приказана процена ризика повећања трошкова сумарних активности на пројекту.

Табела 3. *Процена ризика повећања трошкова сумарних активности на пројекту*

Сумарне активности	Трошкови активности без ризика	Трошкови активности са повећањем цене ресурса за 5%
1. Увод и основне поставке	83.840,00 дин.	88.064,00 дин.
2. Утврђивање и анализа стања паркирања	226.200,00 дин.	237.600,00 дин.
3. Техничке анализе	514.800,00 дин.	540.744,00 дин.
4. Анализа трошкова	100.000,00 дин.	105.040,00 дин.
5. Претходна студија оправданости	892.840,00 дин.	937.840,00 дин.
6. Закључна разматрања	76.200,00 дин.	80.040,00 дин.

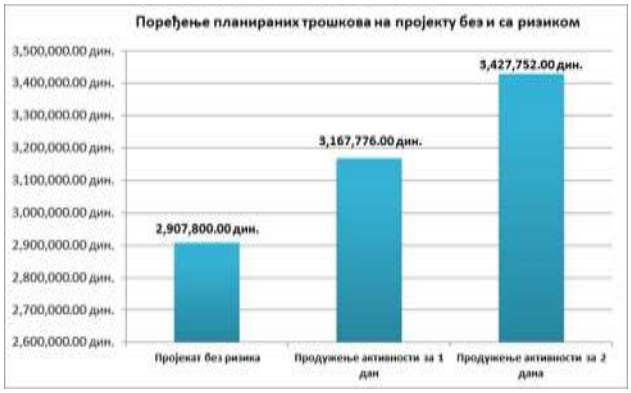
Други ризични догађај који је анализиран у оквиру пројекта представља повећање цене свих ресурса за 10%. У табели 4. је приказана процена ризика повећања трошкова сумарних активности на пројекту.

Табела 4. *Процена ризика повећања трошкова сумарних активности на пројекту за 10%*

Сумарне активности	Трошкови активности без ризика	Трошкови активности са повећањем цене ресурса за 10%
1. Увод и основне поставке	83.840,00 дин.	92.288,00 дин.
2. Утврђивање и анализа стања паркирања	226.200,00 дин.	249.000,00 дин.
3. Техничке анализе	514.800,00 дин.	566.688,00 дин.
4. Анализа трошкова	100.000,00 дин.	110.080,00 дин.
5. Претходна студија оправданости	892.840,00 дин.	982.840,00 дин.
6. Закључна разматрања	76.200,00 дин.	83.880,00 дин.

Трећи ризик који је анализиран у оквиру пројекта јесте продужење времена реализације свих активности за један радни дан. Осим што ће са продужењем времена трајања сваке активности доћи и до повећања трошкова пројекта, сада се повећавају и часови рада (Work) потребни за реализацију пројекта, што је разлог директне повезаности ова два параметара.

Последњи анализирани ризик у оквиру пројекта јесте продужење времена реализације свих активности за два радна дана. И у овом случају као и у претходном долази до повећања трошкова потребних за реализацију пројекта као и потребних часова рада. На слици 3. је приказано поређење планираних трошкова на пројекту без и са ризиком.



Слика 3. *Поређење планираних трошкова на пројекту без и са ризиком*

8.2. Утицаји различитих сценарија на ефекте и показатеље пројекта

Сценарио представља деловање више неповољних ризика истовремено. У оквиру мастер рада биће размотрена два сценарија:

- Сценарио 1: повећање цене ресурса за 5% и продужење времена реализације подактивности на пројекту за 2 дана.
- Сценарио 2: повећање цене ресурса за 10% и продужење времена реализације подактивности на пројекту за 1 дан.

На слици 4. је приказано поређење показатеља пројекта за сва три сценарија.

Слика 4. *Планирани трошкови на пројекту за сва три сценарија*

9. ЗАКЉУЧАК

Задатак мастер рада базиран је на примени знања стеченог на предмету *„Управљање пројектима“* како би се приказале технике планирања времена потребних за реализацију активности на пројекту, начин и поступак повезивања активности и креирање базе неопходних ресурса и додељивање истих активностима на пројекту. На основу добијених резултата могу се извести следећи закључци.

Потребни ресурси који могу реализовати пројекат су: руководилац пројекта, главни пројектант саобраћаја, главни пројектант грађевине, два пројектанта саобраћаја, један пројектант грађевине и један економиста.

У функцији активности који чине саставни део садржаја пројекта, а према областима које су проистекле из пројектног задатка, пројекат траје укупно 160 дана, а укупно време трајања у часовима износи 6.856 часа.

Део пројекта који је завршен у планираном пресеку стања износи 53% (ово је произвољно постављена ситуација). Тиме се могу пратити активни трошкови на пројекту (Actual Cost) који износе 1.504.648,00 динара и преостали трошкови (Remaining Cost) у износу од 1.403.152,00 динара. Планирани укупни трошкови на пројекту (Total Cost) износе 2.907.800,00 динара тако да се може закључити да пројекат према инвестиционој вредности спада у пројекат са средњом инвестиционом вредношћу (укупни трошкови пројекта износе 24.624,09 евра и тиме спада у пројекте чији су укупни трошкови између 10.000 – 100.000 евра што је карактеристично за пројекте средње инвестиционе вредности).

Посматрајући сумарне активности на пројекту, највише средстава је потребно за реализацију сумарне активности „Претходна студија оправданости“ у износу од 892.840,00 динара што је и оправдано с обзиром на број подактивности које ова сумарна активност садржи као и ангажованост самих ресурса. Најмање средстава потребно је за реализацију сумарне активности „Закључна разматрања“ у износу од 76.200,00 динара.

У овом мастер раду комплетно је одрађен план рада, потребни ресурси и трошкови пројекта везани за израду Генералног пројекта паркинг гараже у Пожаревцу. Све активности у оквиру садржаја генералног пројекта су реалне активности које су неопходне за израду генералног пројекта паркинг гараже у Пожаревцу.

10. ЛИТЕРАТУРА

[1] Атанасковић П., „Управљање пројектима - практикум за студенте“, Универзитет у Новом Саду, Нови Сад, 2012.

[2] Атанасковић П., „Управљање пројектима - изводи са предавања“, Универзитет у Новом Саду, Нови Сад, 2017.

[3] Јовановић П., „Управљање пројектом“, Висока школа за пројектни менаџмент, Београд, 2010.

Кратка биографија:

Никола Стокић рођен је у Пожаревцу 1993. године. Дипломски рад је одбранио 2016. године на Факултету техничких наука у Новом Саду из области Саобраћај и транспорт из предмета Паркирање и јавне гараже.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 656.11.08
DOI: https://doi.org/10.24867/01DS05Krsmanovic

ПРИМЕНА ДИНАМИЧКЕ СЕГМЕНТАЦИЈЕ СА АСПЕКТА БЕЗБЕДНОСТИ САОБРАЋАЈА

THE APPLICATION OF DYNAMIC SEGMENTATION FROM ROAD SAFETY ASPECT

Сања Крсмановић, *Факултет техничких наука, Нови Сад*

Област – САОБРАЋАЈ

Кратак садржај – *Анализа безбедности саобраћаја на путној мрежи представља важан задатак управљача пута. У раду су представљени могућности примене диниамичке сегментације са аспекта безбедности саобраћаја.*

Abstract – *Traffic safety analysis on the road network is an important task for the road authority. The paper presents the possibilities of applying dynamic segmentation from the aspect of road safety.*

Кључне речи: *Безбедност саобраћаја, пут, динамичка сегментација.*

1. УВОД

Путна инфраструктура у хијерархији значајности утицаја на безбедност саобраћаја заузима веома високо место. Стога је обавеза државе (као управљача пута) да сопственим механизмима (организација, људски и технички ресурси, методологија рада, финансије и др.) прати и управља безбедношћу саобраћаја над путном мрежом у оквиру својих надлежности.

Географски информациони систем (ГИС) може на одговарајући начин пружити моћне и ефикасне алате за побољшање квалитета процеса доношења одлука у безбедности саобраћаја. Једна од значајнијих функција ГИС је техника динамичке сегментације), која се сматра ефикасним средством за прецизније и флексибилније управљање путевима. Примена динамичке сегментације обезбеђује начин на који се пут може поделити и сегментирати према обележјима, другим речима, овај метод обезбеђује флексибилност да се линеарне функције поделе у нови скуп сегмената где год се промени одређено обележје. У овом приступу обележја се линеарно реферирају и динамички повезују са ентитетима који формирају мрежу.

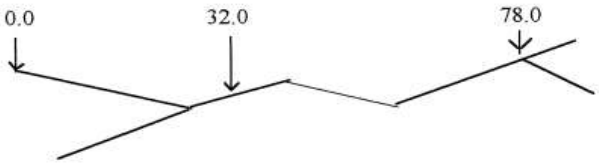
Предмет рада је динамичка сегментација на путној мрежи. Основни циљ рада је сагледати основне елементе и могућности примене динамичке сегментације пута за потребе анализа у безбедности саобраћаја.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Драган Јовановић, ред. проф.

2. ДИНАМИЧКА СЕГМЕНТАЦИЈА

Динамичка сегментација је могућност унутар „Arc/info“ топологије лук-чвор, која пружа окружење за моделовање и анализирање линеарних облика као што су путеви, реке, железничке пруге и линије корисности. Географски информациони системи просторно представљају облике у X и Y координатама. Са X Y представљањем, динамичка сегментација има могућност да складишти, прикаже, испита и анализира особине дуж линеарних облика у смислу познате локације. Динамичка сегментација моделује линеарне облике користећи трасе и догађаје. Мере дуж трасе су употребљене да дају локацију догађаја.

Траса је „лук“ који има мерења дуж трасе како би се омогућило позиционирање догађаја на траси у односу на неку тачку која служи као почетак мерења. Траса може почети на почетку лука, али то не мора увек бити случај. Траса може имати једну или више дефинисаних особина на својој дужини. Мера је вредност која утврђује локацију дуж трасе, где се појављује догађај. Ови догађаји могу бити тачкасти догађаји, као незгоде, или линеарни догађаји, као аутобуске тарифе, или трајни догађаји, као стање коловозне конструкције које се често мења.



Слика 1. *Мере дуж трасе*

Атрибути као што су: класификације путева, типови или стања коловозне конструкције, незгоде итд., су догађаји који су лоцирани дуж трасе која је дефинисана мерама. Белешке догађаја се састоје од барем кључне ставке догађаја и локације мере. Кључна ставка догађаја је нумеричка или словна вредност која успоставља везу између белешке догађаја и трасе којој тај догађај припада.

Постоје три типа догађаја који су приказани у динамичкој сегментацији и то су: тачкасти, линеарни и трајни догађаји. Тачкасти догађаји се разликују од линеарних и трајних догађаја зато што се они појављују на одређеним тачкама дуж трасе што је супротно у односу на линеарне и трајне делове трасе. Примери тачкастих догађаја су локације незгода,

локације вертикалне сигнализације итд. Линеарни догађаји нису трајни зато што имају недостатке у подацима. Употреба две мерне вредности које описују почетак и крај догађаја је оно што чини разлику у структури линеарног догађаја, у односу на структуру трајних и тачкастих догађаја.

Линеарни догађаји су смештени на траси употребом мера локација од (почетни положај) и до (крајњи положај). Трајни догађаји су попут линеарних догађаја у смислу да се односе на локације дуж трасе, али су различити по томе што трајни догађаји представљају податке који обухватају читаву трасу. Подаци за трајне догађаје немају недостатака и стога само бележе тачку на којој се карактеристике мењају. Податак трајног догађаја даје локацију дуж трасе само по мерама до или положај и нема вредности мере од, као код линеарних догађаја.

Један од начина да се сегменти класификују јесте да их поделите на основу фиксне или променљиве дужине сегмената. У статичкој методи сегментације дужина сваког сегмента је иста. Поделом пута на сегменте фиксне дужине добијају се сегменти на том путу, а сегменти са великом густином незгода идентификовани су као сегменти са високом стопом незгода. Неки од недостатака статичких метода су следећи:
1) расподела незгода у неколико сегмената и лоша идентификација сегмената са великом фреквенцијом незгода; Граница неких сегмената може бити унутар сегмената великих густина незгода и то доводи до тога да ће много незгода са истим обележијима да се нађе у два различита сегмента, и самим тим и резултати сегментације нису савршено поуздани.
2) слабо усклађивање између дужине сегмената у методама статичке сегментације и реалне дужине опасних сегмената пута.
Ако је дужина сегмента са већом или мањом стопом саобраћајних незгода већа или мања од дужине сваког фиксног сегмента, одговарајућа дужина сваког сегмента се не може идентификовати помоћу тренутних статичких метода. Због тога ће постојати веће грешке при идентификацији сегмената са великом стопом незгода методом сегментације фиксне дужине.

3. ЗНАЧАЈ ПРИМЕНЕ ДИНАМИЧКЕ СЕГМЕНТАЦИЈЕ

Често се расправља о ограничењима традиционалног ГИС-а при моделовању облика који се могу сврстати у било коју од следећих категорија употребом лук-чвор топологије:
- облици са сегментираним подацима;
- веза једног-ка-многима;
- положај на линеарним облицима.

Ово је велики проблем зато што већина података нема хомогене карактеристике.
Атрибути, као стање коловозне конструкције, могу променити дугачак део пута. Ове разлике могу бити моделоване у традиционалном ГИС-у, али нема потребе за побољшањем података онолико често колико се услови мењају. Пут који је недавно обновљен, нпр. траса приказана на слици 2., имаће хомогене атрибуте између чворова. Само два чвора су потребна за идентификацију хомогеног подручја.

Слика 2. *Хомогеност трасе*

Како време протиче, различити делови коловозне конструкције се погоршавају у различитим нивоима и они се обнављају када дође до потребе за тим. Ови променљиви услови треба да буду изражени дуж лука. Ово води ка додавању и брисању чворова који показују промене, што је приказано на слици 3.

Слика 3. *Приказивање промена на путу*

Ако дође до потребе да се додају други подаци у вези пута, као нпр. локација саобраћајних знакова, саобраћајних незгода итд., број чворова, који су захтевани да би се тачно одредили ти догађаји, постаје толико велики да покривеност постаје тешка за одржавање.
С друге стране, динамичка сегментација сакупља податке за одвојене регионе на табелама догађаја. Табеле догађаја имају записе, назване догађаји, који су забележени дуж трасе употребом мера о којима је расправљано раније. Различити записи догађаја, на пример, локације незгода, локација саобраћајних знакова, ознаке коловозне конструкције итд., могу бити одржани и побољшани одвојено, без утицаја на основну покривености трасе.
Традиционални ГИС користи лук-чвор модел који даје локације у односу на XY координате по Декартовом координатном систему. Ова метода није погодна за линеарне облике који нормално имају пријављена растојања у односу на исту локацију. Приказивање локације незгоде или саобраћајног знака XY координатама не би било практично за људе који морају да идентификују облике на терену.
Динамичка сегментација приказује локације на линеарним облицима употребом линеарних мерних вредности, кодираних у табелама одељака. Одељци имају вредности за почетну меру и завршну меру. Почетна позиција и завршна позиција одељка, изражене су као процентуални положај дуж лука. Имајући почетну и крајњу меру у табели одељка могуће је да се прикаже било која локација дуж трасе.
Традиционални ГИС користи лук-чвор топологију да моделује облике, док динамичка сегментација креира трасе на којима се могу поставити догађаји.
ГИС не може адекватно представљати различите типове података на једној локацији. Ово је случај зато што чворови морају бити креирани сваки пут када дође до промене, што доводи до тога да је база података обимна или захтевна за одржавање. Динамичка сегментација прави трасе, одређује мерења, почињући од референтне тачке одабране од стране корисника. Догађаји су складиштени у табелама догађаја, које могу бити независно ажуриране када се појаве промене. Једини захтев је да се користе исте мерне јединице и референтни положај и у табели особина догађаја (RAT) и у табели

догађаја. Локацију на којој се налази саобраћајни знак, је доста лакше пронаћи, него кад је локација одређена XY координатама.

4. СЕГМЕНТАЦИЈА НА ДРЖАВНОМ ПУТУ ІБ-12

Управљање безбедношћу саобраћаја на путној мрежи представља велики изазов. Последњих година пажњу стручне јавности изазивају савремене процедуре за унапређење безбедности путева, као што су Управљање опасним местима, мапирање ризика на путевима и сл. Примена ових процедура условљена је познавањем основних обележја саобраћајних незгода и њихових последица, пута и саобраћаја. Обележја пута и саобраћаја представљају основу за објективно утврђивање ризика на путевима. Међутим, дистрибуција ризика на путној мрежи се разликује и да би се коректно утврдила величина ризика неопходно је, између осталог, да се хомогенизују обележја пута и саобраћаја.

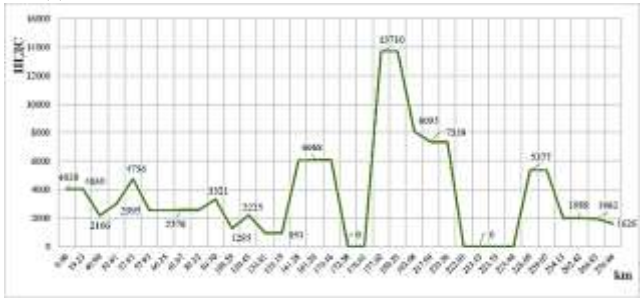
4.1 Предмет и циљ истраживања

Предмет истраживања је примена динамичке сегментације на путном правцу. Основни циљ је извршити сегментацију путног правца на хомогена обележја. Обележја за сегментацију су: просечни годишњи дневни саобраћај (ПГДС) и ограничење брзине. Простор истраживања је државни пут ІБ-12.

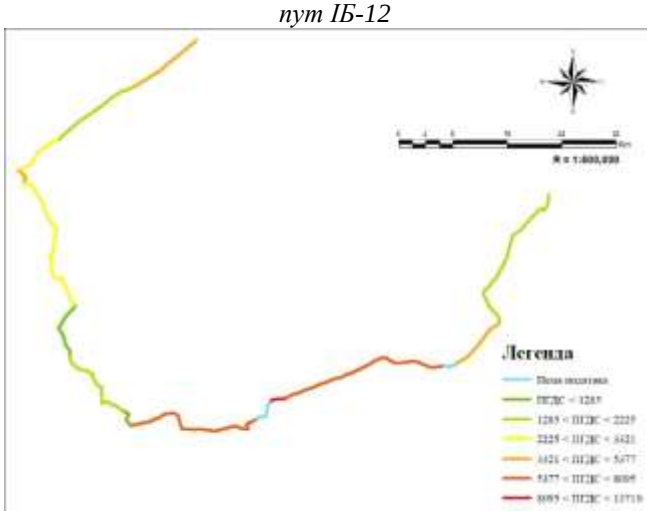
4.2 Резултати

4.2.1 ПГДС

Анализом ПГДС за 2016. годину издвојена су 33 сегмента на путном правцу. Средња вредност ПГДС је 3.625 воз/дан, а максимална вредност је 13.710 воз/дан.



Слика 4. *Сегментација према величини ПГДС-а, пут ІБ-12*



Слика 5. *Сегментација према величини ПГДС-а, пут ІБ-12*

4.2.2 Брзина

Путни правац се простире кроз велики број урбаних средина, али и великом дужином ван урбаних средина, па су доминантна ограничења брзине од 50 km/h и 80 km/h. Међутим, поједина одступања су у функцији елемената пута и окружења, па се на овом путном правцу јављају и ограничења од 70 km/h, 60 km/h, 40 km/h и 30 km/h (График 1.)

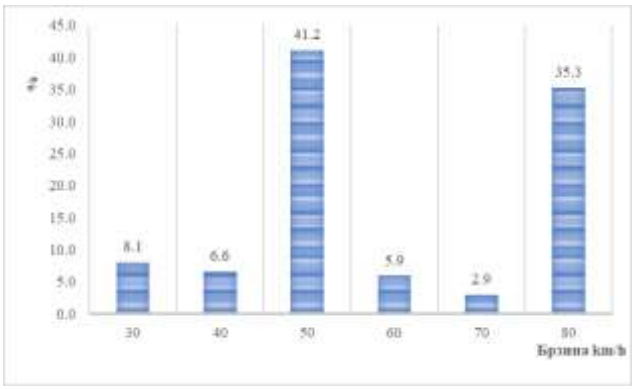
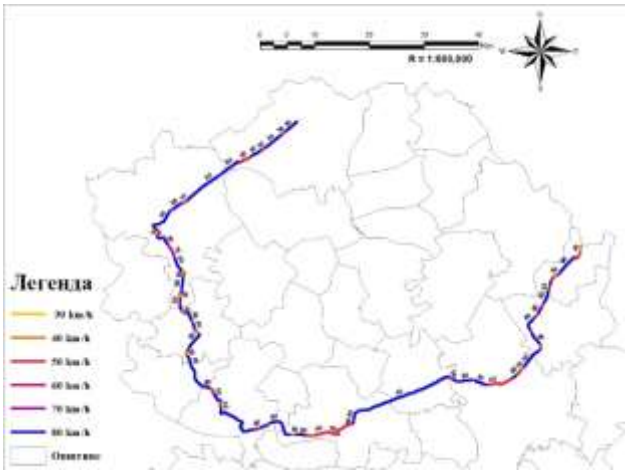


График 1. *Структура деоница према ограничењу брзине, пут ІБ-12*

Укупан број сегмената на основу величине ограничења брзине је 136.

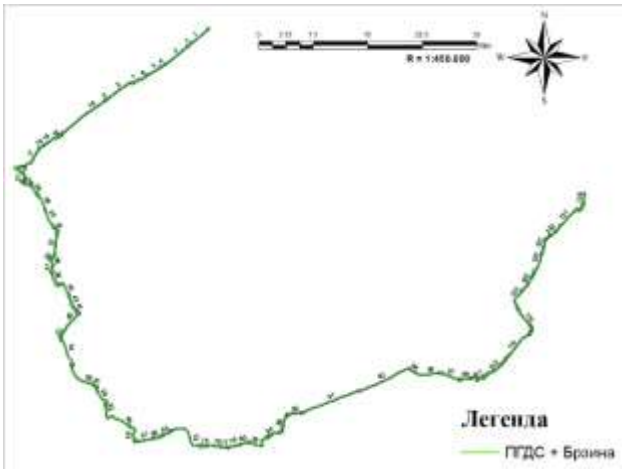
Слика 6. *Сегментација према величини ограничења брзине, пут ІБ-12*



Слика 7. *Сегментација према величини ограничења брзине, пут ІБ-12*

4.2.3 Укупно

Комбинацијом величине ПГДС-а и величине ограничења брзине издвојено је укупно 136 сегмената.



Слика 7. *Сегментација према величини ПГДС-а и ограничења брзине, пут ИБ-12*

5. ЗАКЉУЧАК

Сегментација је користан метод у смислу лоцирања незгода, које се појављују дуж пута, кроз посебни метод путне сегментације и идентификовање сегмената са високим и ниским нивоом незгода.

Стручњаци за безбедност саобраћаја могу утврдити део путева на коме се догађају незгоде. У многим земљама, претодно истраживање у области метода сегментације путева показало је да се идентификација путева у погледу саобраћајних незгода може примењивати дељењем пута на неколико сегмената једнаке дужине и узимајући у обзир број незгода у сваком сегменту.

Као главни део саобраћајне инфраструктуре, путеви играју кључну улогу у друштвеном, економском и културном развоју урбаних и руралних подручја. Једна од главних последица небезбедности на путевима су незгоде са возилима који доводе до огромних људских и финансијских трошкова. С тога, најефикаснији начин да се побољша безбедност путне инфраструктуре је да се спречи појава незгода. Незгоде се могу проучавати у две фазе: превенција и деловање. Главни циљ превенције је да се избегне незгода, идентификују сегменти високог ризика и избегну потенцијални ризици на путу. Насупрот томе, главни циљ деловања је смањење броја и тежине последица тако што се идентификују сегменти високог ризика за настанак незгоде „High Crash Road Segments“(HCRS) и спроводе се адекватне мере. Типична процедура за идентификацију “(HCRS), почиње класификацијом путне мреже у више група сличних карактеристика, названих популација. Затим, референтна популација се дели на сегменте. На крају, мрежа путева приказана је помоћу одговарајућег метода скрининга мреже (NSM) користећи једно или више мерила перформанси за идентификацију „High Crash Road Segments“(HCRS).

6. ЛИТЕРАТУРА

[1] Boroujerdian, A. M., Saffarzadeh, M., Yousefi, H., & Ghassemian, H. (2014). A model to identify high crash road segments with the dynamic segmentation method. Accident Analysis & Prevention, 73, 274-287.

[2] Cadkin, J. (2002). Understanding dynamic segmentation. ArcUser October-December, 40-43.

[3] Castro, M., Sánchez, J. A., Vaquero, C. M., Iglesias, L., & Rodríguez-Solano, R. (2008). Automated GIS-based system for speed estimation and highway safety evaluation. Journal of Computing in Civil Engineering, 22(5), 325-331.

[4] Chang, K. T. (2006). Introduction to geographic information systems (pp. 117-122). Boston: McGraw-Hill Higher Education.

[5] Elyasi, M. R., Saffarzade, M., & Boroujerdian, A. M. (2016). A novel dynamic segmentation model for identification and prioritization of black spots based on the pattern of potential for safety improvement. Transportation Research Part A: Policy and Practice, 91, 346-357.

[6] Kim, K., & Levine, N. (1996). Using GIS to improve highway safety. Computers, environment and urban systems, 20(4-5), 289-302.

[7] Mahmud, A. R., Sohadi, R. U. R., & Mansor, S. (1998). A GIS support system for road safety analysis and management. Pertanika Journal of Science and Technology, 6(1), 81-93.

[8] Steenberghen, T., Dufays, T., Thomas, I., & Flahaut, B. (2004). Intra-urban location and clustering of road accidents using GIS: a Belgian example. International Journal of Geographical Information Science, 18(2), 169-181.

Кратка биографија:

Сања Крсмановић рођена је у Бајиној Башти 1996. год. Мастер рад на Факултету техничких наука из области Саобраћај – Друмски саобраћај одбранила је 2018. год.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 656.11.021
DOI: https://doi.org/10.24867/01DS06Jovanovic

VREDNOVANJE PREDLOGA REŠENJA ZA POBOLJŠANJE USLOVA ODVIJANJA SAOBRAĆAJA NA RASKRSNICAMA NA Bulevaru Evrope

EVALUATION OF SOLUTION PROPOSALS FOR TRAFFIC IMPROVEMENT FOR Boulevard Europe INTERSECIONS

Katarina Jovanović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – SAOBRAĆAJ I TRANSPORT

Kratak sadržaj – *Ovaj rad prikazuje način analiziranja saobraćaja na raskrsnicama i predlog mera za poboljšanje uslova odvijanja saobraćaja sa funkcionalnog i ekonomskog aspekta.*

Ključne reči: *Saobraćaj, nivo usluge, vrednovanje*

Abstract – *This paper presents a way of analyzing traffic at intersections, as well as proposals for improving the conditions of traffic flow from a functional and economic aspect.*

Keywords: *Traffic, LOS, evaluation*

1. UVOD

Raskrsnice, kao mesta gde dolazi do presecanja saobraćajnih tokova, sa aspekta kapaciteta i nivoa usluge, predstavljaju potencijalno kritična mesta na putnoj i uličnoj mreži. Osnovni načini regulisanja saobraćaja na raskrsnicama jesu:

- Saobraćajni znakovi prioriteta;
- Kružni tok odvijanja saobraćaja;
- Svetlosna signalizacija.

Raskrsnice sa kružnim tokom funkcionišu sigurnije ukoliko geometrijski element uslovljavaju smanjenje prilazne, odnosno, brzine u kružnom toku. Nasuprot tome, ovakva geometrija uslovljava smanjenje kapaciteta raskrsnice. Kružne raskrsnice su raskrsnice sa kombinacijom isprekidanog i neisprekidanog saobraćajnog toka.

Okvirni kapacitet za raskrsnice sa jednom saobraćajnom trakom u zavisnosti odprocenta levih skretanja i faktora neravnomernosti glavnog i sporednog pravca kreće se između 20.000 i 26.000 vozila. Za raskrsnice sa dve saobraćajne trake ova vrednost se kreće između 40.000 i 52.000 vozila.

U okviru ovog rada, izvršena je analiza uslova odvijanja saobraćaja na raskrsnici Bulevara Evrope i ulice Rumenački put i raskrsnici Bulevara Evrope, Bulevara vojvode Stepe i ulice Kornelija Stankovića u Novom Sadu, gde su dati predlozi rešenja poboljšanja uslova odvijanja saobraćaja, kao i funkcionalno, ekonomsko vrednovanje predloženih rešenja, nakon čega se dobija optimalno rešenje, koje se predlaže za realizaciju.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Nenad Ruškić.

2. KARAKTERISTIKE ANALIZIRANIH RASKRSNICA

Analizirane raskrsnice nalaze se na području grada Novog Sada koje su locirane u prigradskoj zoni. Raskrsnica K1 predstavlja mesto je ukrštanja Bulevara Evrope i ulice Rumenački put, koja je klasična četvorokraka kružna raskrsnica, sa dve trake na prilazu i dve trake u kruženju. Raskrsnica K2 predstavlja mesto ukrštanja Bulevara Evrope, Bulevara vojvode Stepe i ullice Kornelija Stankovića, to je kružna raskrsnica sa baj-pas trakom, sa dve trake na prilazu i dve trake u kruženju.

Raskrsnica K1 je regulisana elementima horizontalne i vertikalne signalizacije, vozila u kružnom toku imaju prednost pred vozilima iz ulivnih pravaca. Pešački prelazi su postavljeni na svim prilazima raskrsnice. Na raskrsnici su zastupljene staze za kretanje pešaka i biciklista. Pored individualnih gradski putovanja zastupljene su linije javnog gradskog prevoza putnika. Raskrsnica se sastoji od dve trake na prilazu i dve trake u kruženju. Izgled raskrsnice dat je na slici 1.

Slika 1. Analizirana raskrsnica K1

U blizini raskrsnice K2 nalazi se Osnovna škola, ugostiteljski objekti, stambeni objekti, šoping centri, banke. Pešački prelazi su postavljeni na svim prilazima raskrsnice.

Na raskrsnici su zastupljene staze za kretanje pešaka i biciklista. Pored individualnih gradski putovanja zastupljene su linije javnog gradskog prevoza putnika. Raskrsnica se sastoji od dve trake na prilazu i dve trake u kruženju, i ima baj – pas traku na prilazu 3. Izgled raskrsnice dat je na slici 2.

Slika 2. Analizirana raskrsnica K2

3. TEORETSKE OSNOVE

Saobraćajni tok može biti prost i složen. Prost saobraćajni tok sastoji se od jednog niza vozila koja se kreću u jednom pravcu i u jednom smeru. Složen saobraćajni tok se sastoji od dva ili više prostih saobraćajnih tokova s obzirom na međusobne odnose nizova.
S obzirom na strukturu saobraćajni tok može biti:

- Homogen;
- Nehomogen (mešovit) tok;
- Uslovno homogen.

Nivo usluge je kvalitativni pokazatelj odvijanja saobraćaja na raskrsnici. On se zasniva na utvrđivanju vremenskih zastoja po vozilu za svaku grupu traka i za svaki prilaz, kao i za celu raskrsnicu. Za utvrđivanje nivoa uslige raskrsnica koristi se 6-to stepena skala sa nivoima usluge od A do F (A najbolji, F najlošiji).

3.1. Osnovne karakteristike o kružnim raskrsnicama

Projektovanje geometrije savremenih raskrsnica sa kružnim tokom predstavlja traženje kompromisa između kapaciteta i sigurnosti. Raskrsnice sa kružnim tokom funkcionišu sigurnije ukoliko geometrijski element uslovljavaju smanjenje prilazne, odnosno, brzine u kružnom toku. Nasuprot tome, ovakva geometrija uslovljava smanjenje kapaciteta raskrsnice. Takođe mnogi geometrijski elementi uslovljeni su manevarskim sposobnostima najvećeg vozila očekivanog na raskrsnici.

Osnovni oblik i karakteristike zavise i od položaja raskrsnice u mreži (gradska, vangradska), kao i od očekivanog prisustva pešaka odnosno biciklističkog saobraćaja. Takođe, različit pristup projektovanju se primenjuje kod raskrsnica sa jednom ili više traka u kružnom toku.

4. ANALIZA USLOVA ODVIJANJA SAOBRAĆAJA

Brojanje saobraćaja je izvedeno 13.06.2017. god. u periodu od 06:00 do 21:00 časova.

Evidentirane su sledeće kategorije vozila: bicikl, motocikl, putnički automobil, autobus, lako teretno vozilo (vozilo do 3,5 tone i teretni kombi), srednje teretno vozilo (vozilo preko 3,5 tone), teško teretno vozilo (tri osovine), autovoz (tegljač i kamioni sa prikolicom).

Za raskrsnicu K1 od ukupnog protoka po prilazima, uočava se da prilaz 2 ima najmanji protok, zatim prilaz 3 i prilaz 1, dok je na prilazu 4 protok najveći. Ukupno saobraćajno opterećenje, za 15 sati brojanja iznosi 38.413 vozila, što je u proseku 6.403 vozila na sat.

Najopterećeniji period funkcionisanja raskrsnice je jutarnji vršni period, odnosno period od 07:00-08:00, gde je zabeležen protok od 3357 vozila na svim prilazima, dok je najmanja registrovana vrednost saobraćajnog opterećenja registrovana u periodu od 20:00-21:00 časova, gde je ukupan protok vozila sa svih prilaza iznosio 1371 vozila.

Što se tiče strukture toka na analiziranoj raskrsnici, najveći procenat čine putnička vozila sa 84,91%, a zatim laka teretna vozila sa 5,41%. Autobusi su zastupljeni 1,86%, to su uglavnom vozila javnog gradskog prevoza putnika, srednja teretna vozila u strukturi toka učestvuju sa 3,26%, dok teških teretnih vozila ima 3,10%. Autovozova ima u najmanjoj meri, svega 1,46%.

Za raskrsnicu K2 od ukupnog protoka po prilazima, uočava se da prilaz 1 ima najmanji protok, zatim prilaz 3 i prilaz 2, dok je na prilazu 4 protok najveći.Ukupno saobraćajno opterećenje, za 11 sati brojanja iznosi 31268 vozila, što je u proseku 5212 vozila na sat.

Najopterećeniji period funkcionisanja raskrsnice je popodnevni vršni period, odnosno period od 16:00-17:00, gde je zabeležen protok od 3478 vozila na svim prilazima, dok je najmanja registrovana vrednost saobraćajnog opterećenja registrovana u periodu od 20:00-21:00 časova, gde je ukupan protok vozila sa svih prilaza iznosio 2029 vozila.

Što se tiče strukture toka na analiziranoj raskrsnici, najveći procenat čine putnička vozila sa 87,02%, a zatim laka teretna vozila sa 4,79%. Autobusi su zastupljeni 1,99%, to su uglavnom vozila javnog gradskog prevoza putnika, srednja teretna vozila u strukturi toka učestvuju sa 2,42%, dok teških teretnih vozila ima 2,47%. Autovozova ima u najmanjoj meri, svega 1,31%.

5. PROGNOZA SAOBRAĆAJNOG OPTEREĆENJA

Razlikujemo postojeće i planirano stanje. Postojeće saobraćajno opterećenje (može se prebrojati) i koristi se za preduzimanje neposrednih akcija u regulisanju saobraćaja. Prognozirano saobraćajno opterećenje daje buduće količine saobraćaja za koje treba obezbediti odgovarajuće kapacitete gradske putne mreže i sistema javnog gradskog prevoza.

Na osnovu brojanja sa raskrsnice ulica Rumenačka – Kornelija Stankovića, tj.raskrsnice koja je povezana sa analiziranim raskrsnicama, vršena je prognoza budućeg saobraćaja za narednih 10 godina. Na osnovu brojanja iz 2009 godine i brojanja iz 2017 godine na raskrsnici ulica Rumenačka – Kornelija Stankovića zabeležen je pad saobraćaja, ako bi se intezitet saobraćaja izjednačio sa 2009 godinom, došli smo do zaključka da će saobraćaj na analiziranim raskrsnicama porasti za 32%.

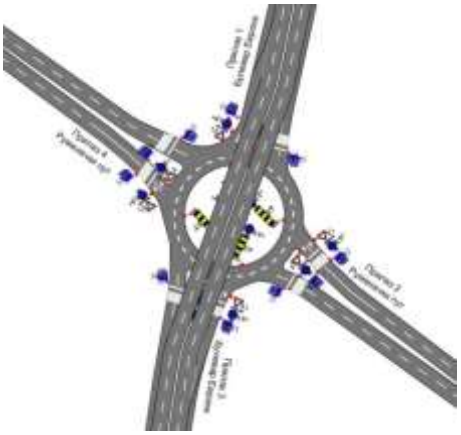
6. PREDLOG MERA ZA POBOLJŠANJE USLOVA ODVIJANJA SAOBRAĆAJA NA ANALIZIRANIM RASKRSNICAMA

Radi poboljšanja uslova odvijanja saobraćaja na analiziranim raskrsnicama, u cilju povećanja bezbednosti nemotorizovanih učesnika u saobraćaju, istovremeno obezbeđujući smanjene brzine kretanja vozila kroz raskrsnicu, u sklopu mera za poboljšanje, predlažu se sledeće varijante:

Varijanta 1 – Postojeće stanje sa dogradnjom nadvožnjaka

Varijanta 2 –Četvorokraka signalisana raskrsnica

Na osnovu predloženih varijanti, metodom proračuna, odrediće se koja je varijanta najbolja za analizirane raskrsnice, koja obezbeđuje najbolje uslove odvijanja saobraćaja, čime se smanjuju ili eliminišu redovi čekanja, smanjuju vremenski gubici i povećava nivo usluge. Izgled Varijante 1 za raskrsnicu K1 dat ne na slici 3.

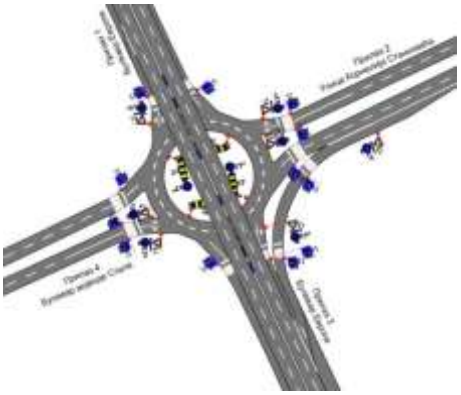


Slika 3. Varijanta 1 za raskrsnicu K1

Izgled Varijante 2 za raskrsnicu K1 dat je na slici 4.

Slika 4. Varijanta 2 za raskrsnicu K1

Izgled Varijante 1 za raskrsnicu K2 dat je na slici 5.



Slika 5. Varijanta 1 za raskrsnicu K2

Izgled Varijante 2 za raskrsnicu K2 dat je na slici 6.

Slika 6. Varijanta 2 za raskrsnicu K2

7. VREDNOVANJE PREDLOGA MERA ZA POBOLJŠANJE USLOVA ODVIJANJA SAOBRACAJA

7.1. Funkcionalno vrednovanje predloženih mera

Funkcionalno vrednovanje vršeno je uz pomoć softverskog paketa SIDRA INTERSECTION, koji je namenjen za proračun kapaciteta i nivoa usluge na raskrsnicama.

Za potrebe izrade ovog rada, u program su unete karakteristike analiziranih raskrsnica, za svaku varijantu ponaosob. U tabeli 1 prikazan je nivo usluge za obe varijante za raskrsnicu K1.

Tabela 1. *Prikaz nivoa usluge za raskrsnicu K1*

Prilazi/Godine	Most 2027		Semafori 2027	
	Vremenski gubici (s/voz)	Nivo usluge	Vremenski gubici (s/voz)	Nivo usluge
Prilaz 1	12,2	B	55,6	E
Prilaz 2	150,9	F	24,3	C
Prilaz 3	325,4	F	35,2	D
Prilaz 4	152,7	F	42,5	D
Raskrsnica	Nivo usluge	F	Nivo usluge	D

U tabeli 2 prikazan je nivo usluge za obe varijante za raskrsnicu K2.

Tabela 2. *Prikaz nivoa usluge za raskrsnicu K2*

Prilazi/Godine	Most 2027		Semafori 2027	
	Vremenski gubici (s/voz)	Nivo usluge	Vremenski gubici (s/voz)	Nivo usluge
Prilaz 1	30,7	D	73,9	E
Prilaz 2	182,2	F	47,9	D
Prilaz 3	15,5	C	51,8	D
Prilaz 4	202,4	F	65,2	E
Raskrsnica	Nivo usluge	F	Nivo usluge	E

7.2. Ekonomsko vrednovanje predloženih mera

Ekonomska analiza se vrši sa stanovišta celog društva i njen cilj je da se uradi procena uticaja projekta na dobrobit stanovništva i regiona. Ekonomska analiza se vrši na osnovu informacija koje se dobijaju iz finansijske analize. Nakon uspostavljanja ekonomskih novčanih tokova, može se proceniti da li će projekat doneti društvu neku društvenu vrednost.

U tabeli 3 dat je prikaz troškova sa aspekta građevinskih radova, za obe predložene varijante za raskrsnicu K1.

Tabela 3. *Troškov gradnje za raskrsnicu K1*

Varijante	Cena ($)
Izgradnja nadvožnjaka	3.928.571,43 $
Izgradnja četvorokrake signalisane raskrsnice	300.382,51 $

U tabeli 4 dat je prikaz troškova sa aspekta građevinskih radova, za obe predložene varijante za raskrsnicu K2.

Tabela 4. *Troškov gradnje za raskrsnicu K2*

Varijante	Cena ($)
Izgradnja nadvožnjaka	3.571.428,57 $
Izgradnja četvorokrake signalisane raskrsnice	321.491,04 $

7.2. Ušteda u troškovima eksploatacije

Troškovi goriva nastaju zaustavljanjem vozila na raskrsnici. Saobraćajne gužve mogu imati različite uticaje na društvo: troškovi održavanja vozila, eksploatacije vozila, povećanje cene vremena putovanja , povećanje potrošnje goriva, troškovi nepruženih transportnih usluga, itd. U tabeli 5 dat je prikaz uštede u troškovima goriva za raskrsnicu K1.

Tabela 5. *Ušteda u troškovima goriva za raskrsnicu K1*

УШТЕДА НА ГОДИШЊЕМ НИВОУ	2.813.087,59 $	3.804.343,20 $
УШТЕДА ЗА ПЛАНСКИ ПЕРИОД	22.504.700,75 $	30.434.745,64 $
↓		
УШТЕДА У ПЛАНСКОМ ПЕРИОДУ КАДА ОДУЗМЕМО ИНВЕСТИЦИЈУ	18.576.129,32 $	30.134.363,13 $

U tabeli 6 dat je prikaz uštede u troškovima goriva za raskrsnicu K2.

Tabela 6. *Ušteda u troškovima goriva za raskrsnicu K2*

УШТЕДА НА ГОДИШЊЕМ НИВОУ	1.515.728,94 $	1.962.005,64 $
УШТЕДА ЗА ПЛАНСКИ ПЕРИОД	12.125.807,49 $	15.696.045,17 $
↓		
УШТЕДА У ПЛАНСКОМ ПЕРИОДУ КАДА ОДУЗМЕМО ИНВЕСТИЦИЈУ	8.554.378,92 $	15.374.554,13 $

8. ZAKLJUČAK

Nakon definisanja osnovnih karakteristika raskrsnice, sledi prikaz rezultata dobijenih brojanjem.

Na osnovu brojanja sa raskrsnice ulica Rumenačka – Kornelija Stankovića, tj.raskrsnici koja je povezana sa analiziranim raskrsnicama, vršena je prognoza budućeg saobraćaja za narednih 10 godina.

Za narednih 10 godina, očekivani broj vozila u vršnom satu za raskrsnicu K1 iznosi 4.431 vozila, dok za raskrsnicu K2 iznosi 4.591 vozila.

Na osnovu prognoze saobraćaja izvršena je analiza nivoa usluge.

U okviru Poglavlja 6 date su varijante predloženih rešenja, i to:

- Varijanta 1 – Postojeće stanje sa dogradnjom nadvožnjaka
- Varijanta 2 – Četvorokraka signalisana raskrsnica

Nakon definisanih predloga mera, izvršeno je njihovo fukcionalno i ekonomsko vrednovanje.

Sa aspekta funkcionog vrednovanja, za raskrsnicu K1 bolji nivo usluge omogućava Varijanta 2, gde su vremenski gubici 39,3 s/voz što predstavlja nivo usluge D. Varijanta 2 za raskrssnicu K2 takođe omogućava bolji nivo usluge (E), gde vremenski gubici iznose 60,1 s/voz.

Sa ekonomskog gledišta, za raskrsnicu K1 povoljnija je Varijanta 2, jer su troškovi izgradnje četvorokrake raskrsnice jeftiniji od troškova izgradnje nadvožnjaka. Za raskrsnicu K2 takođe je povoljnija Varijanta 2 sa ekonomskog aspekta.

Na osnovu uštede u troškovima ekspoloatacije za raskrsnicu K1, Varijanta 2 daje veću uštedu u troškovima goriva. Takođe, Varijanta 2 doprinosi većoj uštedi u troškovima goriva i za raskrsnicu K2.

Varijanta 2, odnosno izgradnja četvorokrake signalisane raskrsnice, izdvaja se kao povoljnije tehničko rešenje za obe analizirane raskrsnice, koje bi uticalo na poboljšanje uslova odvijanja saobraćaja na raskrsnici, jer ostvaruje manje vremenske gubitke, daje boljinovo usluge, izdvaja se kao jeftinije rešenje sa aspekta troškova gradnje i ima veću uštedu u troškovima eksploatacije u odnosu na Varijantu 1.

4. LITERATURA

[1] (2010). Highway Capacity Manual. The National Academies

[2] (2000). Highway Capacity Manual. The National Academies

[3] http://www.putevi-srbije.rs. (n.d.)

[4] ĐorĐević, T., & Vuk, B. (2002). *Kapacitet putnih i uličnih ukrštanja prioritetne raskrsnice (novi koncept)*. Novi Sad: Fakultet tehničkih nauka

[5] Kuzović, Lj. (1994). *Vrednovanje u upravljanju razvojem i ekploatacijom putne mreže.* Beograd: Saobraćajni fakultet.

[6] Kuzović, Lj. (2000). *Kapacitet i nivo usluga drumskih saobraćajnica*. Beograd: Saobraćajni fakultet.

[7] Kuzović, Lj.,Bogdanović, V. (2010). *Teorija saobraćajnog toka.* Novi Sad: Fakultet tehničkih nauka.

[8] Pravilnik o saobraćajnoj signalizaciji. ("Sl. glasnik RS", br. 134/2014)

Kratka biografija:

Katarina Jovanović rođena je u Užicu 1994. god. Master rad na Fakultetu tehničkih nauka iz oblasti Saobraćaj i transport, na smeru Projektovanje i organizacija odbranila je 2018.god.
kontakt: jovanovic_katarina@hotmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 656.1:005.6
DOI: https://doi.org/10.24867/01DS07Mladenovic

МЕРЕ ЗЕ ПОВЕЋАЊЕ КВАЛИТЕТА УСЛУГЕ ТРАНСПОРТА ПРЕДУЗЕЋА „ЕУРО ЛИДЕР“ ДОО- ЖБЕВАЦ

MEASURES TO INCREASE THE QUALITY OF TRANSPORT SERVICES OF THE COMPANY "EURO LIDER" D.O.O.-ZBEVAC

Стефан Младеновић, Павле Гладовић, *Факултет техничких наука, Нови Сад*

Област – САОБРАЋАЈ

Кратак садржај – *Основни циљ израде овог рада јесте упознавање начина пословања транспорта, и да се кроз анализу возног парка уоче недостаци у пословању овог предузећа и предложе мере за превазилажење тих проблема. Потребно је анализирати организациону структуру, структуру возног парка, обрадити финансијске резултате, како би се уочили недостаци и самим тим предложиле мере које ће довести до успешног пословања предузећа.*

Кључне речи: *Систем квалитета, друмски транспорт.*

Abstract – *The main goal of this work is to get acquainted with the mode of transport business, and through the analysis of the fleet, it identifies the shortcomings in the business of this company and proposes measures to overcome these problems. It is necessary to analyze the organizational structure, the structure of the fleet, process the financial results in order to detect the shortcomings and therefore propose measures that will lead to the successful operation of the company.*

Keywords: *Quality system, road transport*.

1. УВОД

Под транспортом се подразумева скуп или комплекс активности на премештању (превозу) путника и робе уз помоћ транспортних средстава (возила) од „извора“ до „циља“ путовања. У том процесу, путници и роба представљају предмет рада, а транспортна средства средства рада .

Транспорту припада важна улога у процесу производње, јер се као обавезан елемент у реализацији производње појављује превоз сировина, материјала, полуфабриканата и готових производа. Без квалитетног транспорта нема ни квалитетне производње ни потрошње, па ни квалитетног животног стандарда. То значи да је транспорт привредна делатност која служи као логистичка подршка свим другим људским активностима [1].

Особеност транспорта се огледа у суштинској разлици његових специфичних карактеристика у односу на друге области материјалне производње, јер нема: производња, продаја, потрошње, већ се прво продаје, затим најчешће истовремено производи и троши и не може се складиштити [4].

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Павле Гладовић.

2. ПОЈАМ И ЗНАЧАЈ ДРУМСКОГ ТРАНСПОРТА

Историјски списи доказују постојање каравана са натовареним животињама још 3500 година пре нове ере. Људи су од давнина били у сталном покрету било због тражења склоништа, у потрази за храном а у каснијем периоду, у доба робне размене и развојем трговине због превоза људи и робе којом се трговало.

У савременим економским теоријама транспорт се најчешће сматра самосталном облашћу материјалне производње и значајном привредном делатношћу. Његова самосталност се огледа у томе што се за организацију и извршење процеса транспорта ангажује посебна радна снага и транспортна средства.

Транспорт представља скуп или комплекс активности помоћу којих се врши премештање (превоз) путника и робе уз помоћ транспортних средстава (возила) од „извора до циља“ путовања и који има своју теорију, технику и технологију управљања.

Посебност транспорта се огледа у његовим специфичним карактеристикама које су различите у односу на друге области материјалне производње јер код транспорта се прво продаје, затим најчешће истовремено производи и троши и његов производ (транспортна услуга) се не може ускладиштити. Појављује се као обавезан елемент у реализацији производње где је неопходно транспортовање сировина, полупроизвода и готових производа.

Транспортни систем се састоји из скупа транспортних средстава, саобраћајница по којима се крећу транспортна средства, техничких уређаја и опреме, објеката који обезбеђују нормалан рад транспортних средстава (објекти за пријем, смештај и отпрему путника, за одржавање транспортних средстава и сл.) и информационих система.

Транспортна средства представљају возила која су намењена за извршење транспортних задатака. Саобраћајнице представљају специјално припремљене и опремљене путеве по којима се крећу транспортна средства.

Технички уређаји, опрема и објекти се састоје из сервисно-ремонтних радионица, ауто-база, гаража, складишта, утоварно-истоварних места, робних и путничких станица (терминала) итд.

3. ИСТОРИЈАТ ПРЕДУЗЕЋА

Предузеће „ЕУРО ЛИДЕР“ је основано 2005. године. Основна делатност фирме је транспорт различитих врста робе, како у земљи тако и у иностранству. Предузеће чине менаџери, диспечери и возачи са дугогодишњим искуством који послове обављају професионално и одговорно уз праћење и међусобну комуникацију.

Предузеће има 500 запослених. Поседују камионе који врше транспорт на свим дестинацијама Европе а о квалитету сведоче многи задовољни корисници. Најчешће дестинације су Немачка, Француска, Холандија, Белгија, Енглеска, Шпанија, Италија, Пољска, Чешка, Румунија, Мађарска итд.

Возни парк предузећа чине 30 шлепера марке „ВОЛВО“ и 30 полуприколице марке „Schmitz“, носивости 24т-33 еуро палете, са еколошким моторима еуро 5 и еуро 6 генерације [2]. Лого предузећа приказан је на слици 1.

Слика 1.- *Лого предузећа*

Удаљеност предузећа од значајнијих регионалних центара приказан а је у следећој табели.

Табела 1.- *удаљеност предузећа од значајних регионалних центара*

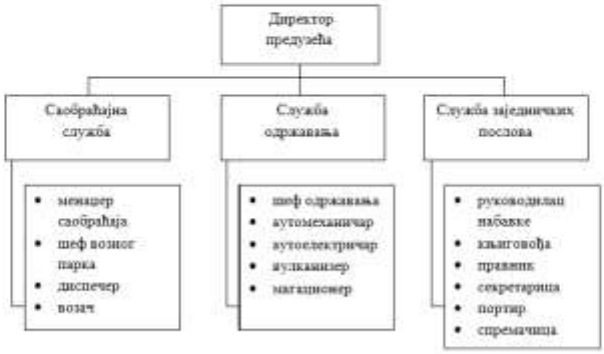
Удаљеност од	Удаљеност (у км)
Београда	360км
Регионалног центра	11км Врање
Магистралног пута	1км Београд-Скопље
Луке	300км Солун
Железнице	5 км Ристовац

4. ПОСЛОВНА ДЕЛАТНОСТ ПРЕДУЗЕЋА

Предузеће „Еуро Лидер“ у својој структури поседује следеће секторе: сектор економско-финансијских послова, аутосервис, обезбеђење, сектор транспорта. Укупан број запослених у предузећу износи 50. На слици 2. можемо видети организациону шему предузећа.

Структура запослених радника у предузећу се може посматрати кроз:

- Струкуту запослених радника по радним јединицама,
- Структуру запослених по степену стручне спреме.



Слика 2.- *Организациона шема предузећа*

У табели 2. дата је структура запослених радника у предузећу по радним местима. Са графика 1. видимо да је највећи проценат запослених радника у саобраћајној служби (68%), затим 18% од укупног броја запослених радника у предузећу је запошљено у служби одржавања а 14% у служби заједничких послова. Када се погледа однос броја запослених радника и броја возила, добија се податак да на једном возилу долази 1,67 радника.

Табела 2- *структура запослених радника по радним местима*

Саобраћајна служба	Менаџер саобраћаја	1
	Шеф возног парка	1
	Диспечер	2
	Возач	30
Служба одржавања	Шеф одржавања	1
	Аутомеханичар	2
	Аутоелектричар	2
	Вулканизер	1
	Магационер	1
Служба заједничких послова	Директор предузећа	1
	Руководилац набавке	1
	Књиговођа	1
	Правник	1
	Секретарица	1
	Портир	2
	Спремачица	2
Укупно	∑	50

График 1- *Процентуални приказ запослених по службама*

5. ОРГАНИЗАЦИЈА РАДА ПРЕДУЗЕЋА

Унутрашњи вид транспорта је онај транспорт где се при његовом обављању не прелази државна граница, то је транспорт који се обавља на територији општине, града,округа...

Да би неко предузеће обављало унутрашњи транспорт мора да поседује одређене дозволе за рад, а самим тим и да има потребна докуметна. Транспортна документа која регулишу транспорт робе представљају доказ за извозника о извршеној продаји робе. Друга важна улога ових докумената односи се на пренос ризика у оним пословима код којих је уговором или одговарајућом инкотермс клаузулом то предвиђено.
Најзначајнија докуметна која се односе на транспорт су:

1) камионски товарни лист (CMR),
2) железнички товарни лист (CIM),
3) шпедитерска потврда,
4) диспозиција за отпрему робе.

Међународни транспорт је транспорт где се при обављању прелази најмање једна државна граница. Да би предузеће обављало међународни транспорт потребно је да има посебна документа за возило и за возача, као и документа која прате реализацију услуге.

Документи за возило:

- саобраћајна дозвола (за вучно возило и за прикључно возило),
- полиса обавезног осигурања
- међународна полиса осигурања ("зелени картон")
- потврде о техничко-експлоатационим условима које морају задовољавати теретна возила којима се обавља међународни друмски превоз

Документи за возача:

- возачка дозвола одговарајуће категорије,
- међународна возачка дозвола
- лична карта
- пасош и визе
- документ о запослености
- обавезно здравствено осигурање и картица међународног здравственог путног осигурања
- тахографски улошци

Документи који прате реализацију услуге:

- путни налог
- товарни лист (ЦМР товарни лист)
- појединачна билатерална или ЦЕМТ дозвола за међународни транспорт
- царински документи (ТИР карнет, АТА карнет, јединствена царинска исправа)

6. СИСТЕМ КВАЛИТЕТА

Реч квалитет потиче од латинске речи “qvlitas”, што значи каквоћу, својства, особине. Иначе ова реч се чује скоро свакодневно у комуникацији између људи, када се говори о квалитету привредних процеса, квалитету активности, квалитету услуга, квалитету живота итд.

TQM- Тотални менаџмент квалитета

Дефиниција гласи : менаџмент приступ у компанији усредсређен на квалитет, заснован на учешћу свих њених чланова, усмерен на дугорочан успех кроз задовољење купца, а у корист свих чланова компаније и друштва [3].

Појам менџмента квалитетом је широк и сагласно стандардима серије ISO 9000 укључује планирање квалитета, управљање квалитетом, обезбеђење и унапређење квалитета, као што је приказано на слици 3.

Слика 3. *Основне компоненте менаџмента квалитетом*

7. ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Основни циљ информационих система је сакупљање, чување, снимање, обрада и преношење информација. Како предузећа, тако и појединци користе ИС за управљање својим операцијама, представљање понуда различитих услуга, унапређењу личних способности и капацитета и сл.

Опрема која се користи у бази је:

- PC рачунари који су повезани са главним сервером тј. везом са главним централним рачунаром,
- Четири персонална рачунара и један лаптоп који имају инсталиране Windows XP оперативне системе који имају апликацију да подржавају рад система за контролу и управљање возила.
- Прикључак Интернет DSL/WAN (getwey) за GPRS комуникацију с опремом у возилима и везу са серверским рачунарем;
- LAN мрежа капацитета 1.5 gb/sek опрема за непрекидно напајање и мрежни преклопник;
- Савремена комуникација преко уграђеног модема.

8. ТРОШКОВИ ЕКСПЛАТАЦИЈЕ ВОЗИЛА У ДРУМСКОМ ТРАНСПОРТУ

Трошкови у друмском транспорту се могу поделити на : сталне и променљиве.

Таб. 3 *Стални трошкови по возилу у току године у еур*

Амортизација возила	19500
Одржавање возила	6500
БЛД по запосленом раднику	24048
Регистрација и осигурање возила	1600
Остали трошкови	5000
Укупно	**56648**

Табела 4- *Променљиви трошкови по возилу у току једне године у еур*

Трошкови гума	12710
Трошкови горива	75950
Трошкови моторног уља	1520
Трошкови уља у мењачу	550
Трошкови за погонски мост	550
Трошкови уља за управљачки механизам	120
Материјал за текуће оправке (2.43% набавне вредности возила)	2900
Текуће оправке	2000
Укупни променљиви трошкови	**96300**

9. АНАЛИЗА ИЗМЕРИТЕЉА РАДА И ЕКСПЛАТАЦИЈЕ ВОЗИЛА

Анализа пређеног пута теретног возила марке „Волво“ у току једног обрта рађена је на релацији Бујановац-Битољ-Нант-Марсељ-Софија. Измеритељи су утврђени удве варијанте: постојеће и предложене а резултати су приказани у табели 5.

Табела 5. *Поређење резултата постојеће и предложене варијанте*

Параметар	Стварна варијанта	Предложена варијанта	Разлика	%
Ai	1	1	/	0
Q	44	68	24t	35↑
ADi	12	12	/	0
ADs	12	12	/	0
ADr	12	12	/	0
AHr	77	77,23	0,23	0,3↑
AHw	73,50	72,235	1,265	1,7↓
ρ	0.27	0.268	0,002	0,7↓
δ	0.95	0.94	0.01	1,05↓
Vs	82,18 km/h	81,68 km/h	0,5	0,6↓
AK	6040 km	5900 km	140	2,3↓
Ve	78,44 km/h	76,4 km/h	2.04 km/h	2.6↓
AKt	4600	4810	210 km	4.4↑
AKn	210	0	210 km	100 ↓
β	0.76	0.82	0.06	7.3↑
ω	0.036	0	0.036	100↓
AZλ	2	3	1	33.3↑
Ksd	503.3 km	491.67 km	11.63	2.3↓
U	99600 tkm	104640 tkm	5040 tkm	4.8 ↑
Kst1	2263.64 km	1538.82 km	724.82	32↓
Kstλ	2300 km	1603.3 km	696.7 km	30.2↓
γ	0.917	0.944	0.027	2.9↑
ε	0.902	0.906	0.002	0.2↑
W'Q	0,571 t/hr	0,88 t/hr	0.309 t/hr	35↑
W'U	1293, 5tkm/hr	1354.91 tkm/hr	61.41t/hr	4.5↑
W Q	0.153 t/hi	0.236 t/hi	0.83 t/hi	35↑
W U	345,83 tkm/hi	363.33 tkm/hi	17.5 tkm/hi	4.8↑

10. ПРЕДЛОГ МЕРА ЗА ПОВЕЋАЊЕ ЕФИКАСНОСТИ РАДА И ПОСЛОВАЊА ВОЗНОГ ПАРКА ПРЕДУЗЕЋА „ЕУРО ЛИДЕР“

1. Запошљављње већег броја стручних лица који ће водити рачуна о измеритељима рада и експлатације возног парка.
2. Треба тежити да вожње у што већој мери буду под теретом.
3. Радити на модернизацији и проширавању возног парка.
4. Увести маркетинг службу која ће радити на испитивању тржишта у погледу транспортних захтева.
5. Увођење сектора за безбедност саобраћаја.
6. Примена ISO стандарда.
7. Код превоза на веће релације примена удвојене посаде.
8. Прављење планова пословања на дужи период.

11. ЗАКЉУЧАК

У оквиру рада извршена је анализа постојећег и жељеног стања предузећа „Еуро Лидер“, односно сагледана је организациона и квалификована структура предузећа, структура возног парка, анализа трошкова и показатељи рада возног парка.

Структура возног парка је на завидном нивоу. Као што је већ речено, неопходно је организовати посебну маркетинг службу која ће радити на промоцији и рекламирању предузећа али и која ће обезбеђивати терете у повратним вожњама како би се смањили празни и нулти километри и повећао приход возила по пређеном километру. Неопходно је имплементирати процесе управљања квалитетом ради што квалитетније организације пословања између самих запослених чиме би се подигао квалитет према потенцијалним корисницима.

12. ЛИТЕРАТУРА

[1] Гладовић П. „Технологија друмског саобраћаја“, Факултет техничких наука, Нови Сад, 2013. Година

[2] Документација аутотранспортног предузећа „Еуро Лидер”

[3] Гладовић П. „Систем квалитета у друмском транспорту “, Факултет техничких наука, Нови Сад, 2012. Година

[4] Гладовић П. „Организација друмског саобраћаја “, Факултет техничких наука, Нови Сад, 2014. Година

Кратка биографија:

Стефан Младеновић рођен је у Врању 1993. године. Дипломски рад је одбранио 2016. године на Факултету техничких наука у Новом Саду из области Саобраћај и транспорт из предмета технологија и организација друмског саобраћаја.

stefanmladenovic7@gmail.com

U realizaciji Zbornika radova Fakulteta tehničkih nauka u toku 2018. godine učestvovali su sledeći recenzenti:

Aco Antić
Aleksandar Erdeljan
Aleksandar Ristić
Bato Kamberović
Biljana Njegovan
Bogdan Kuzmanović
Bojan Batinić
Bojan Lalić
Bojan Tepavčević
Bojana Beronja
Branislav Atlagić
Branislav Nerandžić
Branislav Veselinov
Branislava Kostić
Branislava Novaković
Branka Nakomčić
Branko Milosavljević
Branko Škorić
Damir Đaković
Danijela Lalić
Darko Čapko
Darko Marčetić
Darko Reba
Dejan Ubavin
Dejana Nedučin
Dragan Ivanović
Dragan Ivetić
Dragan Jovanović
Dragan Kukolj
Dragan Mrkšić
Dragan Pejić
Dragan Šešlija
Dragana Bajić
Dragana Konstantinović
Dragana Šarac
Dragana Štrbac
Dragoljub Šević
Dubravka Bojanić
Dušan Dobromirov
Dušan Gvozdenac
Dušan Kovačević
Dušan Uzelac
Duško Bekut
Đorđe Ćosić
Đorđe Lađinović
Đorđe Obradović
Đorđe Vukelić
Đula Fabian
Đura Oros
Đurđica Stojanović
Filip Kulić
Goran Sladić
Goran Švenda
Gordana Milosavljević
Gordana Ostojić
Igor Budak
Igor Dejanović
Igor Karlović
Ivan Beker
Ivana Katić
Ivana Kovačić
Ivana Miškeljin
Jasmina Dražić
Jelena Atanacković Jeličić
Jelena Borocki
Jelena Kiurski
Jelena Radonić
Jovan Petrović
Jovanka Pantović
Ksenija Hiel
Laslo Nađ
Lazar Kovačević
Leposava Grubić Nešić
Livija Cvetičanin
Ljiljana Vukajlov
Ljiljana Cvetković
Ljubica Duđak
Maja Turk Sekulić
Marko Todorov
Marko Vekić
Maša Bukurov
Matija Stipić
Milan Rapajić
Milan Simeunović
Milan Trifković
Milan Trivunić
Milan Vidaković
Milena Krklješ
Milica Kostreš
Milica Miličić
Mijodrag Milošević
Milovan Lazarević
Miodrag Hadžistević
Miodrag Zuković
Mirjana Damnjanović
Mirjana Malešev
Mirjana Radeka
Mirko Borisov
Miro Govedarica
Miroslav Hajduković
Miroslav Popović
Mitar Jocanović
Mladen Kovačević
Mladen Radišić
Momčilo Kujačić
Nemanja Stanisavljević
Nemanja Sremčev
Nikola Đurić
Nikola Jorgovanović
Nikola Radaković
Ninoslav Zuber
Ognjen Lužanin
Pavel Kovač
Peđa Atanasković
Petar Malešev
Predrag Šiđanin
Radivoje Dinulović
Radovan Štulić
Slavica Mitrović
Slavko Đurić
Slobodan Dudić
Slobodan Krnjetin
Slobodan Morača
Sonja Ristić
Srđan Kolaković
Srđan Popov
Srđan Vukmirović
Staniša Dautović
Stevan Gostojić
Stevan Milisavljević
Stevan Stankovski
Strahil Gušavac
Svetlana Nikoličić
Tanja Kočetov
Tatjana Lončar - Turukalo
Toša Ninkov
Uroš Nedeljković
Valentina Basarić
Velimir Čongradec
Veran Vasić
Veselin Perović
Vladimir Katić
Vladimir Strezoski
Vlado Delić
Vlastimir Radonjanin
Vuk Bogdanović
Zdravko Tešić
Zoran Anišić
Zoran Brujić
Zoran Jeličić
Zoran Mitrović
Zoran Papić
Željen Trpovski
Željko Jakšić