УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

ЗБОРНИК РАДОВА
ФАКУЛТЕТА ТЕХНИЧКИХ НАУКА

Едиција: Техничке науке - зборници
Година: XXXVI
Број: 3/2021

Нови Сад

Едиција: „Техничке науке – Зборници“
Година: XXXVI Свеска: 3

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

Савет за библиотечку и издавачку делатност ФТН,
проф. др Милан Мартинов, председник.

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 трећа 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.

У овом броју штампани су радови студената мастер студија, сада већ мастера, који су радове бранили у периоду од 16.10.2020. до 31.10.2020. год., а који се промовишу 22.03.2021. год. То су оригинални прилози студената са главним резултатима њихових мастер радова.

Известан број 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 3. објављени су радови из области:

- машинства,
- електротехнике и рачунарства.

У свесци са редним бројем 4. објављени су радови из области:

- грађевинарства,
- саобраћаја,
- графичког инжењерства и дизајна и
- архитектуре.

У свесци са редним бројем 5. објављени су радови из области:

- инжењерског менаџмента,
- инжењерства заштите на раду и заштите животне средине,
- мехатронике,
- геодезије и геоматике,
- управљања ризиком од катастрофалних догађаја и пожара,
- инжењерства информационих система и
- анимације у инжењерству.

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

STRANA

STRANA

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.7
DOI: https://doi.org/10.24867/12AM01Kis

TEHNOLOŠKA PRIPREMA PROIZVODNJE ALATA ZA INJEKCIONO PRESOVANJE DRŽAČA CEVI ZA PODNO GREJANJE

TECHNOLOGICAL PREPARATION FOR THE PRODUCTION OF MOLDS FOR INJECTION OF THE UNDERFLOOR HEATING PIPE HOLDER

Stevan Kiš, Dejan Lukić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – MAŠINSTVO

Kratak sadržaj – *Osnovni predmet i cilj ovog rada se odnosi na prikaz uloge tehnološke pripreme proizvodnje u procesu izrade alata za injekciono presovanje plastike. Opisane su i realizovane aktivnosti konstruisanja alata za konkretan proizvod - držač cevi za podno grejanje, projektovanja tehnološkog procesa izrade vitalnih delova alata (kokila) kao i generisanje upravljačkih programa za njihovu obradu na CNC mašinama uz primenu programskog sistema PTC/Creo Parametric.*

Ključne reči: *Tehnološka priprema proizvodnje, Projektovanje alata za injekciono presovanje, Tehnološki proces izrade, CNC obrada.*

Abstract – *The main subject and goal of this paper is to present the role of technological preparation of production in the manufacturing process of injection molds. The activities of constructing mold for a specific product - pipe holder for underfloor heating, designing the technological process of manufacturing vital parts of mold, as well as generating numerical control program for their machining on CNC machines using the software system PTC/Creo Parametric are described and realized.*

Keywords: *Technological preparation of production, Injection mold design, Manufacturing process planning, CNC machining.*

1. UVOD

Injekciono presovanje ili brizganje uz ekstrudiranje i duvanje predstavlja najrasprostranjeniju metodu proizvodnje proizvoda od plastike [1]. I pored visokih troškova osnovne opreme za injekciono presovanje plastike, koji se odnose na mašinu za brizganje, odgovarajućih alata i uređaja za temperiranje alata, ovaj postupak daje dobre tehnoekonomske efekte i kod serija od nekoliko hiljada proizvoda.

Postupak spada u primarnu preradu polimera tj. preoblikovanje polimera, jer se oblik otpreska dobija od polaznog materijala koji nema određenu formu (granule, komadići i sl.).

Injekcionim presovanjem se oblikuju svi polimeri: duromeri, elastomeri, elastoplastomeri, a posebno je raširena prerada plastomernih materijala [2].

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

U fazi tehnološkog i proizvodnog osvajanja proizvoda najznačajnije mesto ima tehnološka priprema proizvodnje, koja uz konstrukcionu pripremu, ima najveći uticaj na troškove i kvalitet razvoja proizvoda, odnosno proizvodnje. Veoma je bitno da se, pored projektovanja tehnoloških procesa, izvrši analiza tehnologičnosti konstrukcionih rešenja. Da bi se to postiglo mora postojati određeni nivo simultanog odvijanja faza razvoja proizvoda, što za rezultat ima skraćenje ciklusa razvoja, smanjenje troškova i ostvarenje zadovoljavajućeg kvaliteta razvijenog proizvoda [3].

Osnovni cilj ovog rada se odnosi na prikaz uloge tehnološke pripreme proizvodnje u procesu izrade alata za injekciono presovanje polimera, kroz konstruisanje alata za konkretan proizvod-držač cevi za podno grejanje i projektovanje tehnološkog procesa izrade njegovih vitalnih delova.

2. INJEKCIONO PRESOVANJE POLIMERA

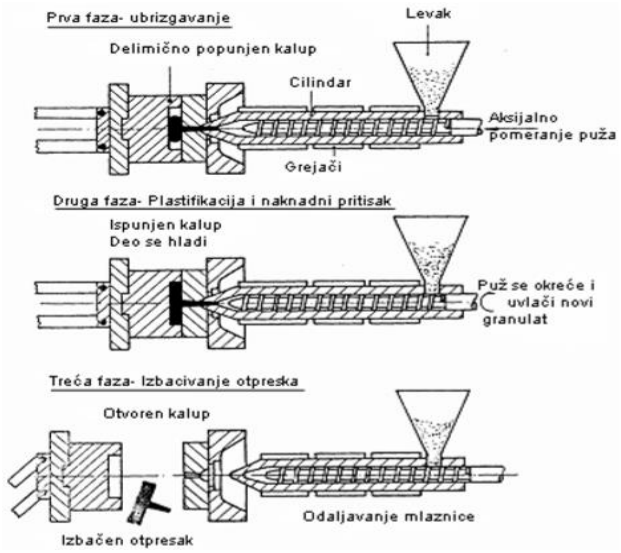
Polimeri se izrađuju u obliku praha, ljuspica, granula, zrnaca ili smole, a njihova prerada u gotove proizvode izvodi se procesima obrade kao što su: presovanje, istiskivanje (ekstruzija), ubrizgavanje, kalandrovanje, itd. Injekciono presovanje je proces kod koga materijal koji je potreban za proizvodnju dela ulazi u zagrevan cilindar, zamešan i proguran u kalupnu šupljinu gde se ohladi i dobija oblik kalupa. Brizganje se široko koristi za proizvodnju različitih delova, od najmanjih komponenta do čitavog tela automobila. Glavne prednosti prerade polimernih materijala ovim postupkom su [1, 2, 4]:

- ušteda materijala,
- kraće vreme izrade gotovog proizvoda,
- velika tačnost delova, dimenzija i oblika,
- dobijaju se sjajne i glatke površine u bojama,

Proces injekcionog oblikovanja odvija se u nekoliko faza od kojih tri osnovne (slika 1):

1. Aksijalnim pomeranjem puža rastopljeni materijal se preko ulivnih kanala ubrizgava u šupljinu kalupa.
2. Obradak se hladi uz intenzivnu cirkulaciju rashladnog sredstva kroz sistem za hlađenje alata. Pri tome, puž deluje na rastopljeni materijal naknadnim pritiskom, kako bi se nadoknadio nedostatak materijala usled skupljanja otpreska pri hlađenju. Nakon završenog hlađenja otpreska, tj. na kraju faze delovanja naknadnog pritiska, puž se vraća, rotira i uvlači novu količinu granulata, topi ga i plastificira.

3. Poslednja faza jeste otvaranje kalupa i izbacivanje obratka.



Slika 1. *Faze injekcionog presovanja* [2]

3. ALATI ZA INJEKCIONO PRESOVANJE PLASTIKE

U procesu proizvodnje proizvoda od plastike tehnologijom injekcionog presovanja-brizganja, centralni deo sistema predstavlja alat, odnosno kalup, kome su podređeni svi ostali delovi posmatranog sistema.

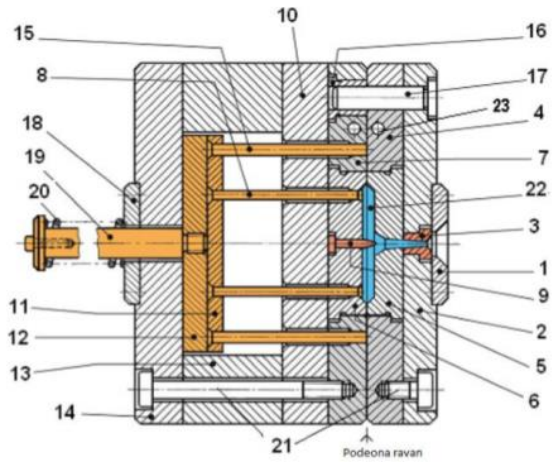
Alat predstavlja proizvod povišenog stepena tačnosti, zbog čega se njegovom projektovanju i izradi posvećuje posebna pažnja.

Kod oblikovanja plastomera postupkom injekcionog brizganja, najvažnije je postići visoko automatizovanu proizvodnju, kod koje se ne treba naknadno obavljati nikakva dorada (samo u retkim slučajevima odvajanje ulivnog kanala).

Osnovni zadaci alata su: prihvatanje rastopa iz mlaznice jedinice za ubrizgavanje i njegovo hlađenje do postizanja forme otpreska, potiskivanje otpreska iz kalupne šupljine i ciklusni rad sistema injekcionog presovanja. Sve te postavljene zadatke alat ne može da izvrši ako nema sledeće elemente [4]:

- Ulivni sistem,
- Kalupnu šupljinu,
- Sistem za izbacivanje otpreska,
- Sistem za vođenje kalupa,
- Kućište i
- Elemente za temperiranje kalupa

Na slici 2 dat je šematski prikaz alata sa osnovnim komponentama: 1. Prsten za centriranje mlaznice, 2. Nepokretna stezna ploča, 3. Ulivna čaura, 4. Nepokretni kalup, 5. Segmentni umetak nepokretne strane, 6. Segmentni umetak pokretne strane, 7. Pokretni kalup, 8. Izbacivači, 9. Jezgra, 10. Međuploča, 11./12. Izbacivačke ploče, 13. Odstojnici ili distantne letve, 14. Pomična ili pokretna stezna ploča, 15. Protivpritisna čivija, 16./17. vođice i vodeće čaure, 18. Centrirani prsten, 19. Potiskivač, 20. Povratna opruga, 21. Spojni vijci, 22. Proizvod (otpresak), 23. Kanali za hlađenje (temperiranje).

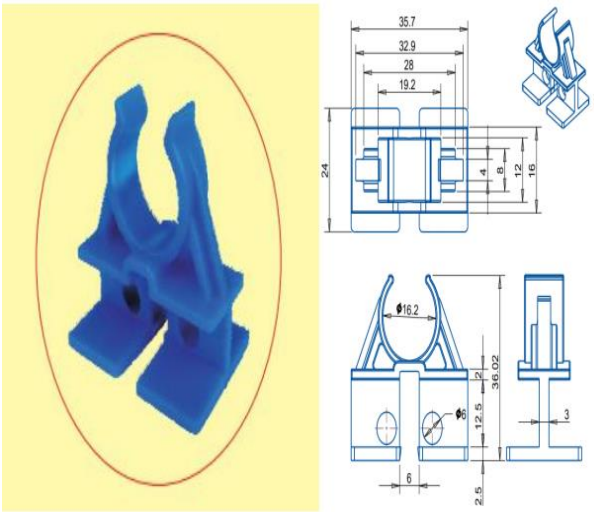


Slika 2. *Struktura alata za injekciono presovanje* [5]

4. PROJEKTOVANJE ALATA ZA INJEKCIONO PRESOVANJE PROIZVODA OD PLASTIKE

4.1 Opis proizvoda od plastike

Osnovni predmet istraživanja u ovom rada se odnosi na projektovanje i izradu alata za injekciono presovanje držača cevi za podno grejanje (slika 3). Osnovna namena proizvoda je pričvršćivanje i fiksiranje cevi na željeni razmak i distanciranje armaturne mreže od podloge.



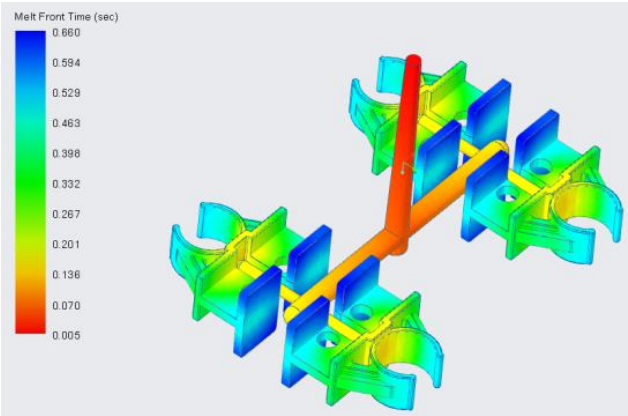
Slika 3. *Držač cevi za podno grejanje* [6]

4.2 Analiza tehnologičnosti za proces injekcionog presovanja proizvoda

Analiza tehnologičnosti konstrukcije dela je izvršena u okviru modula *Mold Analysis* programskog sistema PTC Creo Parametric. U okviru ovog modula mogu se dobiti svi neophodni podaci vezani za sam proces brizganja. Ti neophodni podaci se odnose na materijal koji se brizga, tok materijala pri brizganju, vreme punjenja pri brizganju, pritisak brizganja, pad pritiska brizganja, predviđen kvalitet, itd. Modul *Mold Analysis* omogućava analizu:

- Vremena punjenja kalupne šupljine,
- Pritiska,
- Temperature,
- Pouzdanost popunjavanja kalupa,
- Vremena hlađenja, itd.

Na slici 4 prikazani su izlazni rezultati simulacije vremena punjenja kalupa plastičnom masom.



Slika 4. *Simulacija vremena punjenja kalupa*

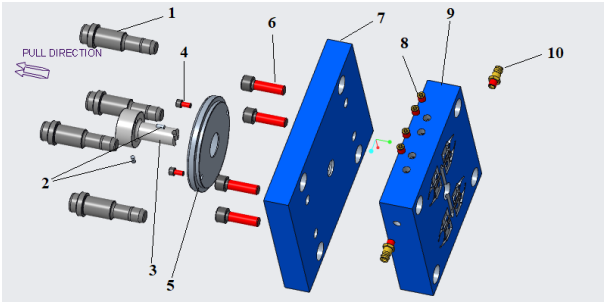
Nakon softverske analize usvojeni su sledeći podaci procesa brizganja posmatranog proizvoda, prema tabeli 1.

Tabela 1. *Parametri brizganja proizvoda*

Vreme punjenja	0.69 s
Temperatura topljenja	250 ºC
Maksimalni injekcioni pritisak	155 Mpa

4.3 Definisanje nepokretne strane alata

Sklop nepokretne strane alata prikazan je na slici 5.

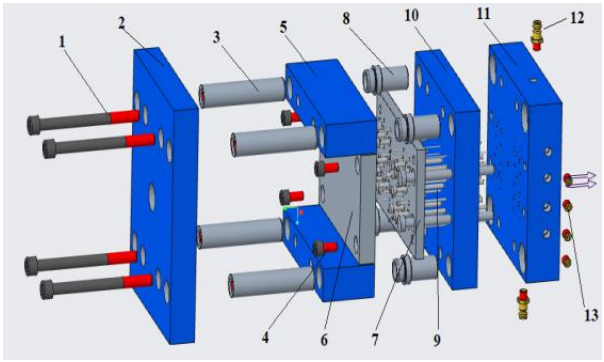


Slika 5. *Nepokretna strana alata*

Elementi sa slike 5 su: 1 – Vođice koja služi za vođenje nepokretne strane alata. 2 – Čivije (Ø4 x 8 mm), služe za fiksiranje dizne kako ne bi došlo do njenog obrtanja oko sopstvene ose. 3 – Dizna koja sprovodi rastopljeni polimer do kalupne šupljine. 4 – Vijak M5, služi za povezivanje centrirnog prstena (5) i osnovne ploče (7). 5 – Centrirni prsten na koji se dovodi mlaznica mašine. 6 – Vijak M10 povezuje osnovnu (7) i kalupnu ploču (9). 7 – Osnovna nepokretna ploča. 8 – Element sistema za hlađenje – čep. 9 – Nepokretna kokila. 10 – Element sistema za hlađenje – priključak.
Kalupna ploča je od niskolegiranog čelika Č.4742 dok je osnovna ploča od ugljeničnog čelika Č.1730.

4.4 Definisanje pokretne strane alata

Sklop pokretne strane alata prikazan je na slici 6. Elementi sa slike 6 su: 1 – Vijak M10 koji služi za povezivanje osnovne ploče (1), distantne letve (5), međuploče (10) i pokretne kokile (11). 2 – Osnovna pokretna ploča. 3 – Čaura, služi za centriranje pokretnog dela alata. 4 – Vijak M8 povezuje ploče izbacivačkog sklopa. 5 – Distantna letva, zajedno sa izbacivačkim pločama obezbeđuje dužinu hoda izbacivača koji je ovde 30 mm. 6 – Ploča oslonca izbacivača, na nju se oslanjaju izbacivači. 7 – Noseća ploča izbacivača, ploča u koju se postavljaju i koja nosi izbacivače. 8 – Čaura, služi za centriranje i vođenje pokretne strane alata. 9 – Izbacivači i povratnici. Povratnici izbacivača, u kontaktu sa nepokretnom stranom alata vraćaju izbacivački sklop u prvobitni položaj. 10 – Međuploča. 11 – Pokretna kokila. 12 – Element sistema za hlađenje – priključak. 13 – Element sistema za hlađenje – čep.
Pokretna osnovna ploča (2), distantne letve (5), izbacivačke ploče (6,7) i međuploča (10) su od ugljeničnog čelika Č.1730 dok je pokretna kokila (11) od niskolegiranog čelika Č.4742.



Slika 6. *Pokretna strana alata*

5. PROJEKTOVANJE TEHNOLOŠKOG PROCESA IZRADE KALUPNIH PLOČA

Mnoge kompanije, među kojima je i Meusburger se bave proizvodnjom i prodajom elemenata za alate za injekciono presovanje plastike, čime se omogućava kupovina već obrađenih ploča sa standardnim otvorima za vođice (čaure) i rupama za vijke koji služe za spajanje sa ostalim delovima alata. Tako da proizvođaču alata ostaje samo obrada gravure, otvora za izbacivače, ulivnih kanala i otvora za hlađenje što znatno ubrzava proces proizvodnje.
U okviru [6] projektovani su tehnološki procesi izrade pokretne i nepokretne kokile u vidu sadržaja tehnološkog procesa (slika 7) i preciziranih karti operacije izrade gravure kokila, na bazi čega su primenom CAM sistema generisne upravljačke informacije za CNC obadu.

Fakultet tehničkih nauka, Departman za proizvodno mašinstvo — **SADRŽAJ TEHNOLOŠKOG PROCESA** — Proizvod; Veličina serije [kom]

Naziv dela: **Pokretna kokila**	Broj dela	Indetifikacioni / Klasifikacioni		Ide u proizvod	Komada
Oznaka i stanje materijala		Vrsta i dimenzije pripremka		Pogon	Odelenje
Oznaka	Kod:	Oznaka:	Kod:		

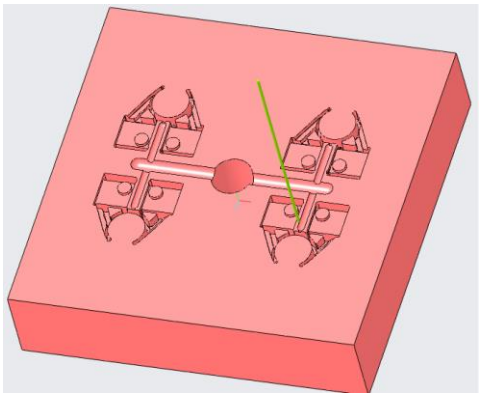
Opera-cija	NAZIV OPERACIJE	MAŠINA: Naziv i oznaka	Vreme [min]: Pripr.	Glavno	Pomoć.	Po kom	Kom/8h
10	BUŠENJE OTVORA ZA HLADJENJE	Uni. glodalica Deckel FP2	10	20	5	25	19
20	GLODANJE	CNC glodalica Maho 700W	30	200	10	210	2
30	MEĐUKONTROLA	Radni sto	1	5	2	7	68
40	TERMIČKA OBRADA	Peć	10	20	10	30	16
50	KONTROLA TERMIČKE OBRADE	Uređaj za merenje tvrdoće	1	3	1	4	120
60	ELEKTROEROZIVNA OBRADA	Erozimat Deckel DE 25	30	80	15	95	5
70	POLIRANJE	Radni sto	1	120	1	121	3
80	ZAVRŠNA KONTROLA	Radni sto	1	5	2	7	68
	Ukupno:						

Izmena					Datum	Izradio	Kontrol.	Šef teh. pr	Šef pogona	Šef kontrole	Listova / List broj

Slika 7. *Sadržaj tehnološkog procesa izrade kokile*

5.1 Izrada gravure primenom CAM modula programskog sistema Creo Parametric

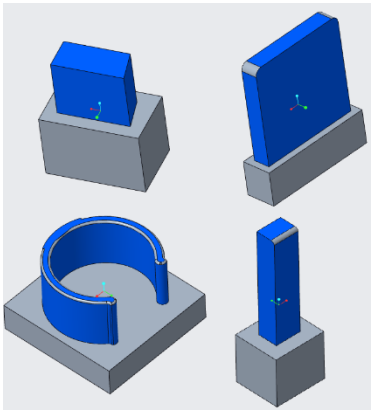
CAM (Computer-Aided Manufacturing) je računarom podržana proizvodnja i obuhvata sve faze proizvodnje u smislu optimizacije korišćenja velikog broja podataka. Osnovna ideja ovakvog načina razvoja tehnologije je da se kompjuter koristi u svrhu generisanja upravljačkih programa za CNC (Computer Numerical Control) mašine alatke. Potreba za razvojem ovakvih programskih sistema proistekla je iz potrebe da se na CNC mašinama alatkama izrađuju delovi složene geometrije brzo, kvalitetno i efikasno. Na slici 8 je prikazan izvod iz simulacije zahvata obrade finog glodanja gravure nepokretne kalupne ploče.



Slika 8. *Simulacija zahvata obrade kalupne ploče*

5.2 Obrada gravure EDM obradom

Određene površine gravure nije moguće potpuno obraditi glodanjem, zbog toga je potrebno konstruisati pune elektrode za obradu na erozimatu. U zavisnosti od vrste i kvaliteta obrade, erodiranje se izvodi sa jednom ili više elektroda. Najčešće se obrada izvodi sa dve (gruba+fina) ili tri (gruba+prethodna+fina) elektrode. Elektrodom za grubu obradu odnosi se skoro celokupni deo materijala, a za finu obradu ostavlja se samo onoliko materijala koliko je potrebno da se uklone neravnine i oštećenja u površinskom sloju koji su nastali pri gruboj obradi. Korišćene elektrode treba prethodno obraditi na CNC glodalici i njihovi 3D modeli su prikazani na slici 9.



Slika 9. *Elektrode za EDM obradu*

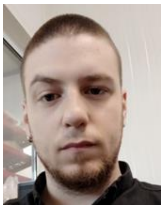
6. ZAKLJUČAK

Kao rezultat opšte prisutnih težnji prilagođavanja savremenim zahtevima tržišta, postavljaju se novi i sve složeniji zahtevi pred proizvodne sisteme, što se u velikoj meri ogleda u potrebi za visokim stepenom automatizacije svih aktivnosti, počevši od razvoja i projektovanja proizvoda pa sve do njihove proizvodnje. U poslednje vreme javlja se sve veća potražnja za proizvodima od plastike, a samim tim i za alatima za njihovu proizvodnju, koji u velikoj meri utiču na kvalitet, vreme i troškove njihove proizvodnje. Cilj ovog rada bio je opis projektnih aktivnosti tehnološke pripreme proizvodnje proizvoda od plastike: definisanje konkretnog proizvoda, u ovom slučaju držača cevi za podno grejanje, projektovanje alata za zadati proizvod i projektovanje tehnološkog procesa izrade njegovih vitalnih delova - kokila.
Alati za injekciono presovanje predstavljaju složeni sistem, sastavljen od velikog broja elemenata, što pre svega zavisi od oblika i broja predmeta koji se u njemu dobijaju. Projektovanje alata pomoću savremenih programskih paketa u velikoj meri olakšava rad projektantima.
Zahvaljujući standardizovanju elemenata alata smanjeno je vreme izrade celokupnog alata i njegove cene koštanja. Mnogobrojna istraživanja su pokazala da se 55% vremena troši za izradu delova kalupne šupljine, a 45% za izradu kućišta. Upotrebom standardnih delova postignuti su kratki rokovi izrade kalupa, visok kvalitet, kao i manji troškovi. Upotrebom CAM softvera u velikoj meri se skraćuje generisanje upravljačkih programa za obradu na CNC mašinama, a samim tim se povećava proizvodnost i ekonomičnost poslovanja.

7. LITERATURA

[1] Perošević, B.: Kalupi za injekciono presovanje plastomera (termoplasta), Naučna knjiga, Beograd, 1995.
[2] Vilotić, D.: Tehnologija injekcionog presovanja polimera, skripta sa predavanja, FTN, Novi Sad, 2015.
[3] Lukić, D., Milošević, M., Todić, V.: Integrisani CAPP sistemi i tehnološka baza podataka, skripta sa predavanja, FTN, Novi Sad, 2014.
[4] Lukić, D.: Razvoj sistema za automatizovano projektovanje tehnoloških procesa izrade alata za brizganje plastike, magistarska teza, FTN, Novi Sad 2007.
[5] Milutinović, M.: Alati za injekciono presovanje, skripta sa predavanja, FTN, Novi Sad, 2017.
[6] Kiš, S.: Tehnološka priprema proizvodnje alata za injekciono presovanje držača cevi za podno grejanje, Master rad, FTN, Novi Sad, 2020.

Kratka biografija:



Stevan Kiš rođen je u Rumi 1996. god. Master rad na Fakultetu tehničkih nauka iz oblasti Mašinstva – Proizvodno mašinstvo, smer Računarom podržane tehnologije obranio je 2020. god.
kontakt: stevan.kis1@gmail.com

Dejan Lukić rođen je u Žablju 1973. god. Doktorirao je na Fakultetu tehničkih nauka 2012. god., a od 2018 je u zvanju vanrednog profesora. Oblast interesovanja je projektovanje tehnoloških procesa, CAD/CAPP/CAM, DfM, optimizacija i logistika proizvodnje, Industrija 4.0.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 629.3.015
DOI: https://doi.org/10.24867/12AM02Jankovic

DINAMIKA VOZILA U ANALIZI SAOBRAĆAJNIH NEZGODA: MODELIRANJE PRILAZNE FAZE

VEHICLE DYNAMICS IN ANALYSIS OF ROAD TRAFFIC ACCIDENTS: MODELING OF THE APPROACHING PHASE

Robert Janković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – MEHANIKA

Kratak sadržaj – *U ovom radu je analizirana prilazna faza vozila za potrebe rekonstrukcije saobraćajnih nezgoda, pomoću Lineikinovog modela. Izvedene su diferencijalne jednačine kretanja vučnog vozila sa poluprikolicom, pomoću Lagranževih jednačina druge vrste za neholonomne sisteme. Primenom programskog paketa Mathematica izvršena je simulacija prilazne faze vozila za bočni sudar.*

Ključne reči: *dinamika vozila, Lineikinov model, sudar, analiza saobraćajnih nezgoda.*

Abstract – *In this paper we analysed the approach phase of vehicles for the needs of traffic accident reconstruction using the Lineikin model. Differential equations of motion of a trailer truck with a semi–trailer are formed using Lagrange equations of the second kind for nonholonomic systems. Simulations of the approach phase of vehicles for a side collision were performed using the Mathematica software.*

Keywords: *vehicle dynamics, Lineikin model, collision, road traffic accident reconstruction.*

1. UVOD

Ovaj rad je nastao sa idejom da u budućnosti bude od koristi prilikom veštačenja sudara dva vozila u saobraćaju. Za analizu dinamike vozila koriste se različiti matematički modeli. Korišćenje veoma složenih modela koji opisuju kretanje vozila verodostojnije od jednostavnijih modela, može dovesti do određenih matematičkih poteškoća prilikom rešavanja, kao i do mnogo dužeg vremena potrebnog za rešavanje, pa razlozi njihove upotrebe moraju biti opravdani. Da bi se došlo do željenih rezultata, u praksi se često mogu koristiti jednostavniji modeli za proučavanje kretanja vozila u zasebnim ravnima, sa manjim brojem stepeni slobode. Prema tome dinamika vozila se može podeliti na:

- Longitudinalnu (uzdužnu) dinamiku: proučava otpore kretanja, performanse i kočenje;
- Vertikalnu dinamiku: analizira oscilacije vozila usled vertikalnih opterećenja i njihov uticaj na udobnost;
- Lateralnu (poprečnu) dinamiku: analizira bočne sile, kretanje u krivini i uticaj na bezbednost putnika, videti [1].

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

U radu je proučen i prikazan slučaj bočnog sudara vučnog vozila sa poluprikolicom i automobila, pomoću Lineikinovog modela za kretanje vozila.

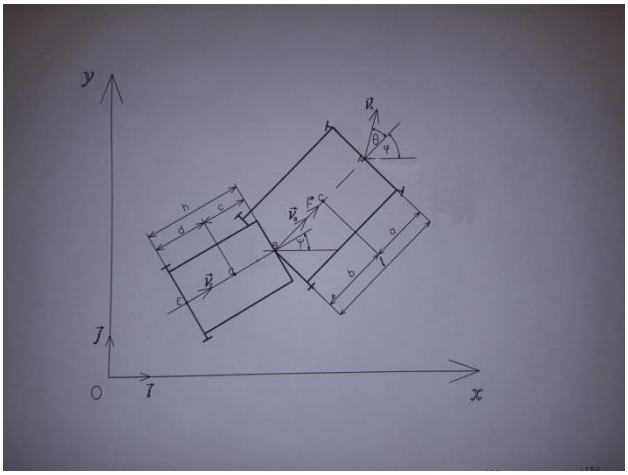
1.2. Postavka problema

Za proučavanje kretanja vučnog vozila sa poluprikolicom pomoću Lineikinovog modela, moraju se usvojiti određene predpostavke pre nego što se krene sa analizom:

- vozilo se kreće po horizontalnoj ravni
- uticaj obrtanja točkova se zanemaruje
- ne dolazi do proklizavanja točkova
- brzine tačaka *A, B* i *E* imaju pravce odgovarajućih točkova.

2. DINAMIKA VUČNOG VOZILA SA POLUPRIKOLICOM

Pri analizi ovog modela ne koristi se pokretni koordinatni sistem pa će ose nepokretnog koordinatnog sistema biti označene malim slovima *x* i *y* kao što je prikazano na slici 1.



Slika 1. *Model vučnog vozila sa poluprikolicom*.

Na slici 1 prikazan je model vučnog vozila sa poluprikolicom, koji je izveden na osnovu Lineikinovog modela automobila. Model ima sledeće karakteristike:

- *l* međuosovinsko rastojanje,
- *a* rastojanje prednje osovine od centra mase *C*,
- *b* rastojanje zadnje (pogonske) osovine od centra mase *C*,
- *h* dužina poluprikolice,
- *c* rastojanje od prednje stranice poluprikolice do centra mase *G*,

- d rastojanje od osovine poluprikolice do centra mase G,
- $\vec{v}_E$ brzina tačke E, koja se nalazi na sredini poluprikolice,
- $\vec{v}_A$ brzina tačke A koja se nalazi na sredini prednje osovine,
- $\vec{v}_B$ brzina tačke B koja se nalazi na sredini zadnje osovine,
- $\vec{F}$ pogonska sila vučnog vozila,
- φ ugao koji uzdužna osa vučnog vozila zaklapa sa x osom,
- θ ugao zakretanja prednjih točkova u odnosu na uzdužnu osu vozila,
- ψ ugao koji uzdužna osa poluprikolice zaklapa sa x osom.

Brzine tačaka A, B, G i E se mogu izraziti na sledeći način:

$$\begin{aligned}\vec{v}_A &= \dot{x}_A\vec{i} + \dot{y}_A\vec{j}\\ \vec{v}_B &= \dot{x}_B\vec{i} + \dot{y}_B\vec{j}\\ \vec{v}_G &= \dot{x}_G\vec{i} + \dot{y}_G\vec{j}\\ \vec{v}_E &= \dot{x}_E\vec{i} + \dot{y}_E\vec{j}\end{aligned} \tag{1}$$

Koordinate tačaka A, B, G i E su date na sledeći način:

$$\begin{aligned}&x_B = x_C - b\cos\varphi; y_B = y_C - b\sin\varphi,\\ &x_A = x_C - a\cos\varphi; y_A = y_C + a\sin\varphi,\\ &x_G = x_C - b\cos\varphi - c\cos\psi; y_G = y_C - b\sin\varphi - c\sin\psi,\\ &x_E = x_C - b\cos\varphi - h\cos\psi; y_E = y_C - b\sin\varphi - h\sin\psi,\end{aligned} \tag{2}$$

Za opisivanje ovog sistema koriste se Lagranževe jednačine druge vrste za neholonomne sisteme čiji je osnovni oblik:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_j} - \frac{\partial L}{\partial q_j} = Q_j^* + \sum_{\beta=1}^{l}\lambda_\beta b_{\beta j}\ (i=1,...,m), \tag{3}$$

gde su

$$q_j\ (j=1,...,m), \tag{4}$$

generalisane kooordinate, a broj generalisanih koordinata je m=5, dok su λ_β (β=1,...,l) Lagranževi množitelji, videti [2]. Generalisane koordinate određuju položaj sistema u svakom trenutku. Za generalisane koordinate ovog sistema uzete su koordinate centra mase x_c i y_c, ugao zakretanja vozila φ, ugao zakretanja prednjih točkova θ i ugao ψ koji uzdužna osa poluprikolice zaklapa sa x osom. Jednačine neholonomnih veza u opštem obliku su predstavljene sledećim izrazom:

$$\sum_{i=1}^{n} b_{\beta j}\dot{q}_j + b_\beta = 0\ (\beta=1,2,...,l), \tag{5}$$

a njihov broj iznosi l=3. To su ograničenja na brzine tačaka A, B i E, koje predstavljaju sredinu prednje osovine vučnog vozila, sredinu njegove zadnje osovine i sredinu osovine poluprikolice, respektivno.

Brzine ovih tačaka imaju pravace odgovarajućih točkova, kao što je prikazano na slici 1. Jednačine neholonomnih veza za razmatrani problem su:

$$tg\varphi = \frac{\dot{y}_B}{\dot{x}_B},\ tg(\varphi+\theta) = \frac{\dot{y}_A}{\dot{x}_A},\ tg\psi = \frac{\dot{y}_E}{\dot{x}_E}, \tag{6}$$

iz kojih se, raspisivanjem u oblik (5) određuju koeficijenti $b_{\beta j}$. Broj stepeni slobode:

$$n=m-l=5-3=2 \tag{7}$$

Lagranžijan L je:

$$L = E_{k_1} + E_{k_2} - \Pi = \frac{1}{2}Mv_c^2 + \frac{1}{2}J_C\dot{\varphi}^2 + \frac{1}{2}J_A\dot{\theta}^2 + \frac{1}{2}M_2v_G^2 + \frac{1}{2}J_G\dot{\psi}^2$$
$$\Pi = 0, \tag{8}$$

gde je:

- M masa vučnog vozila
- M_2 masa poluprikolice
- J_G aksijalni moment inercije poluprikolice za vertikalnu osu koja prolazi kroz tačku G,
- J_C aksijalni moment inercije vučnog vozila za vertikalnu osu koja prolazi kroz tačku C,
- J_A aksijalni moment inercije pri zakretanju upravljačkih točkova.

Generalisane sile nepotencijalnih dejstava su prikazane kao Q_j^*, $j \in \{x_c, y_c, \varphi, \theta, \psi\}$, a izvedene su iz izraza za virtualni rad nepotencijalnih sila.

$$Q_{xc} = F\cos\varphi; Q_{yc} = F\sin\varphi;$$
$$Q_\theta = M_A;\ Q_\varphi = 0;\ Q_\psi = 0, \tag{9}$$

Iz jednačine (3) kada se uvrste izrazi (8) i (9), uzimajući u obzir koeficijente $b_{\beta j}$ određene iz jednačina neholonomnih veza (6) dobija se sistem diferencijalnih jednačina kretanja neholonomnog sistema:

$$\begin{aligned}M\ddot{x}_c &= F\cos\varphi + \lambda_1\sin\varphi + \lambda_2\sin(\varphi+\theta) + \lambda_3\sin\psi\\ M\ddot{y}_c &= F\sin\varphi - \lambda_1\cos\varphi - \lambda_2\cos(\varphi+\theta) - \lambda_3\cos\psi\\ J_C\ddot{\varphi} &= \lambda_1 b - \lambda_2 a\cos\theta - \lambda_3 b\cos(\varphi-\psi)\\ J_A\ddot{\theta} &= M_A\\ J_E\ddot{\psi} &= \lambda_3 h\end{aligned} \tag{10}$$

pri čemu su jednačine neholonomnih veza:

$$\begin{aligned}&\dot{x}_c\sin\varphi - \dot{y}_c\cos\varphi + b\dot{\varphi} = 0\\ &\dot{x}_c\sin(\varphi+\theta) - \dot{y}_c\cos(\varphi+\theta) - a\cos\theta\dot{\varphi} = 0\\ &\dot{x}_c\sin\psi - \dot{y}_c\cos\psi + b\cos(\varphi-\psi)\dot{\varphi} + h\dot{\psi} = 0\end{aligned} \tag{11}$$

Eliminacijom Lagranževih množitelja iz jednačina (10), koristeći sistem (11) dobijaju se diferencijalne jednačine kretanja vučnog vozila sa poluprikolicom, koje će biti rešene numerički. U nastavku sledi rešavanje takvog sistema diferencijalnih jednačina za izabrane vrednosti parametara vučnog vozila sa poluprikolicom i početne uslove.

Pored tog vozila razmotriće se i kretanje automobila čije je kretanje modelirano Lineikinovim modelom, videti [3]. Generalisane koordinate automobila su označene kao generalisane koordinate vučnog vozila, samo što imaju dodat indeks 2.

3. REŠENJE PROBLEMA I GRAFIČKI PRIKAZ

Razmotriće se slučaj u kojem dolazi do sudara vučnog vozila sa poluprikolicom i automobila, gde će se anali-

zirati samo prilazna faza. Izabrani parametri za vučno vozilo sa poluprikolicom su:

M=4000 kg; l=5.81 m; a=2.3 m; b=2.51 m; M_p=5000 kg, c=2.1 m, d=3.2 m,

$$J_C = 8000 kgm^2; J_G = 9500 kgm^2; J_A = 5.2 kgm^2, \quad (12)$$

dok su izabrani parametri za automobil koji se kreće u susret vučnom vozilu:

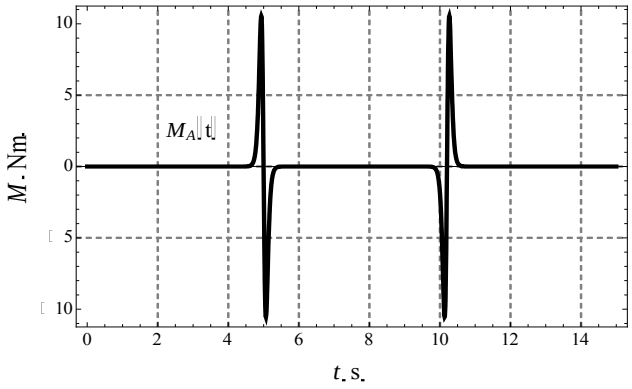
$$M_2 = 1750 \text{ kg}; l_2 = 2.75 \text{ m}; a_2 = 1.25 \text{ m}; b_2 = 1.5 \text{ m}; J_{c2} = 2750 \text{ kgm}^2; J_{A2} = 1.2 \text{ kgm}^2 \quad (13)$$

Analiziran je slučaj bočnog sudara. Vučnim vozilom sa poluprikolicom se upravlja pomoću pogonske sile F i obrtnog momenta M_A, koji utiče na ugao zakretanja pogonskih točkova θ, dok na automobil deluje pogonska sila F_2 i moment M_{A2}.

3.1. Slučaj bočnog sudara

Vučno vozilo sa poluprikolicom skreće ulevo na raskrsnici pod uglom od 90°, kada automobil iz suprotnog smera, koje se kreće translatorno pravolinijski, udara u njega bez prethodnog kočenja.

Pogonska sila F vučnog vozila sa poluprikolicom je konstantna i iznosi F=7500 N, moment M_A je zadat u funkciji vremena tako da se izvrši željeno kretanje, slika 2. Pogonska sila vozila koje se kreće iz suprotnog smera F_2 je takođe konstantna i iznosi F_2=500 N, dok je moment M_{A2} jednak nuli kako bi se vozilo kretalo duž saobraćajne trake.



Slika 2. *Moment upravljanja tokom skretanja vučnog vozila sa poluprikolicom pod uglom od* 90°.

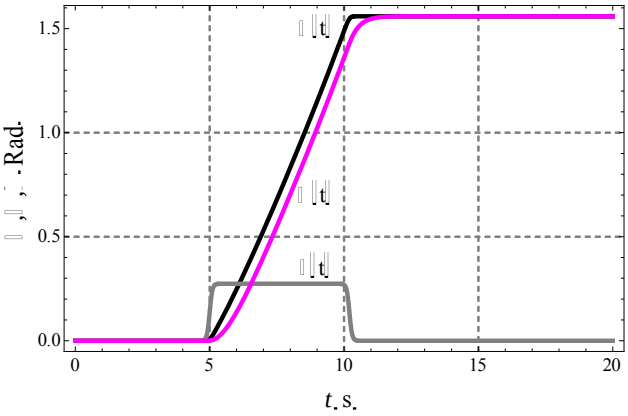
Na slici 3 se vidi da dolazi do zakretanja upravljačkih točkova u skladu sa zadatom funkcijom momenta upravljanja. Uglovi φ i ψ za to vreme rastu od 0° do π/2.

Na slici 4 prikazane su koordinate centra mase vučnog vozila tokom vremena, dok je trajektorija centra mase data na slici 5.

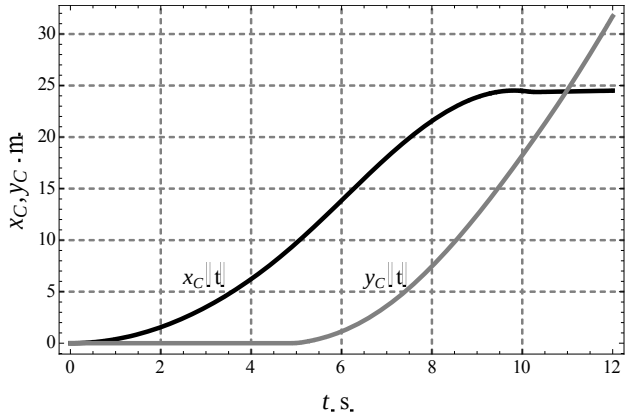
Automobil koji ide u susret vučnom vozilu sa poluprikolicom, kreće se translatorno pravolinijski u skladu sa zadatom pogonskom silom i momentom upravljanja pogonskih točkova, kao i homogenim početnim uslovima.

Generalisana koordinata x_{C2} centra mase automobila je jedina generalisana koordinata automobila koja se u posmatranom slučaju menja tokom prilazne faze vozila i ovde se neće prikazivati.

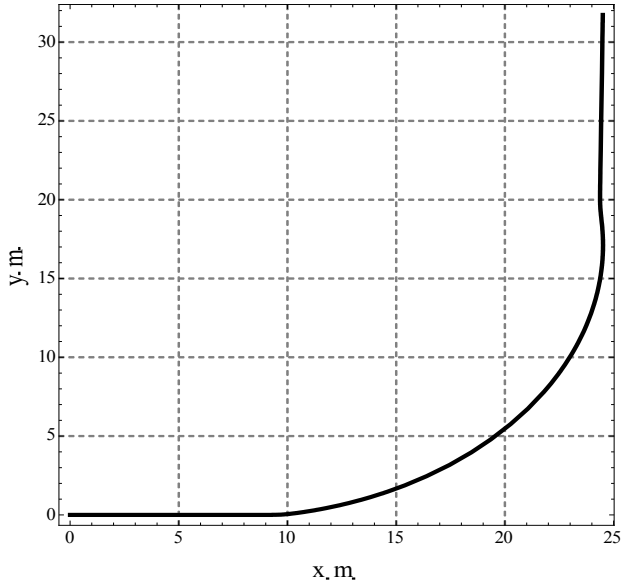
Na slici 6 prikazan je položaj vozila u trenutku sudara automobila u vučno vozilo sa poluprikolicom. Oba vozila su vršila zadate manevre do trenutka sudara. Slika je deo simulacije dobijene uz pomoć programskog paketa *Mathematica.*



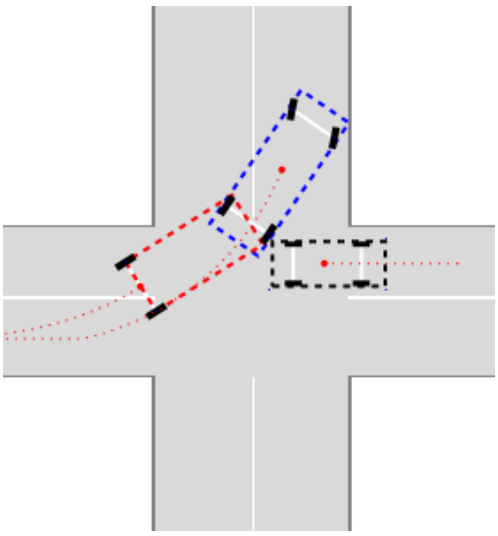
Slika 3. *Uglovi* φ, θ i ψ *vučnog vozila sa poluprikolicom*.



Slika 4. *Koordinate centra mase vučnog vozila sa poluprikolicom tokom zadatog manevra*.



Slika 5. *Trajektorija centra mase vučnog vozila tokom zadatog manevra: skretanje u raskrsnici pod uglom od* 90°.



Slika 6. *Položaj vučnog vozila sa poluprikolicom i automobila u raskrsnici u trenutku sudara*.

4. ZAKLJUČAK

U ovom radu su izvedene i rešene diferencijalne jednačine kretanja vučnog vozila sa poluprikolicom i rešenja su prikazana grafički. Rešene su i diferencijalne jednačine kretanja automobila za zadate početne uslove.

Za rešavanje su korišćene Lagranževe jednačine druge vrste za neholonomne sisteme. Analiziran je slučaj u kojem automobil naleće na vučno vozilo sa poluprikolicom koje skreće na raskrsnici ulevo ne poštujući pravo prvenstva prolaza automobila.

Proučena je prilazna faza kretanja vozila do trenutka njihovog kontakta, dok sama faza sudara i faza kretanja vozila nakon sudara do njihovog zaustavljanja ovde nije proučena jer prevazilazi okvire ovog rada i predstavlja temu za dalje istraživanje. Lineikinov model, iako dosta jednostavan, daje pouzdane rezultate i vrlo je upotrebljiv za modeliranje kretanja vozila za određenu klasu problema u analizi saobraćajnih nezgoda.

5. LITERATURA

[1] W. Chen, H. Xiao, "*Integrated vehicle dynamics and control",* Hefei, Wiley, 2016.

[2] D.T.Spasić, "*Mehanika: osnove, opšte, proširenja",* Univerzitet u Novom Sadu, u pripremi.

[3] В.В.Добронравов, "Основы механики неголономных систем", Издательствовысшая школа, Москва, 1970.

Kratka biografija:

Robert Janković rođen u Doboju 1987. god. Bečelor rad na Fakultetu tehničkih nauka odbranio 2011. god. Iz oblasti Mašinstva – Tehnička mehanika i dizajn u tehnici

Зборник радова Факултета техничких наука, Нови Сад

UDK: 621.43
DOI: https://doi.org/10.24867/12AM03Milojkovic

НОВЕ ГЕНЕРАЦИЈЕ ДИРЕКТНОГ УБРИЗГАВАЊА

NEW GENERATION OF DIRECT INJECTION

Иван Милојковић, *Факултет техничких наука, Нови Сад*

Област –МАШИНСТВО

Кратак садржај – *У овом раду су описани системи директног убризгавања горива у радном простору мотора. Повећање ефикасности СУС мотора одувек је била тема константног истраживања и усавршавања ради максималног искоришћења термодинамичког потенцијала, како би се реални циклус приближио идеалном. Новим концептом директног убризгавања постигнуто је боље образовање, мешање, сагоревање и хлађење смеше услед целокупног убризгавања и испаравања у комори сагоревања. Подложност детонантном сагоревању је смањена, услед чега је повећан степен сабијања. Ови и други фактори утичу да се у односу на конвенционалне системе убризгавања повећају излазни параметри мотора (снага и момент), док је потрошња горива и емисија издувних гасова смањена. Слојевита смеша карактеристична за ове моторе се формира на три начина вођења пуњења: зидом, ваздухом и млазом.*

Кључне речи: *СУС мотор, Директно убризгавање, Повећање ефикасности, Слојевита смеша, ЕГР.*

Abstract – *This paper describes direct fuel injection systems in the engine working space. Increasing the efficiency of SUS engines has always been a topic of constant research and improvement in order to maximize the thermodynamic potential, in order to bring the real cycle closer to ideal. The new concept of direct injection has achieved better formation, mixing, combustion and cooling of the mixture due to the entire injection and evaporation in the combustion chamber. The susceptibility to detonation combustion is reduced, as a result of which the degree of compression is increased. These and other factors affect the increase of engine output parameters (power and torque) compared to conventional injection systems, while fuel consumption and exhaust emissions are reduced. The layered mixture characteristic of these engines is formed in three ways of conducting charge: wall, air and jet.*

Keywords: *IC Engine, Direct injection, Efficiency increase, Layered mixture, EGR.*

1. УВОД

Убризгавање горива је првобитно реализовано механичким, затим комбинацијом механичког и електронског, да би се потпуно прешло на електронске системе.

Прекретница у развоју и већој серијској производњи ових система имала је јапанска компанија *Mitsubishi*, 1996. године. Постоји три начина убризгавања горива са различитим резултатима:

- директно убризгавање са хомогеном смешом,
- директно убризгавање са слојевитом смешом,
- директно убризгавање са слојевитом смешом и компресијским сагоревањем.

Код слојевитог образовања смеше, гориво се убризгава током процеса сабијања. Неопходно је да се изврши раслојавање веома сиромашне смеше у комори сагоревања, тако да се увек у близини свећице налази таква концентрација мешавине горива и ваздуха која ће омогућити сигурно и стабилно упаљење електричном варницом. Потребан је додатан конвертор издувних гасова, због присуства азотових оксида који настају на граници између горивог и негоривог слоја где је смеша сиромашна, јер конвенционални трокомпонентни каталитички конвертори немају значај при учесталом вишку ваздуха при мањим и делимичним оптерећењима када се убризгавање врши пред крај такта сабијања [1].

Постоји три концепта у виду вођења процеса (слика 1):

- зидом,
- ваздухом и
- млазом вођеним процесом. [1]

Слика 1. *Класификација процеса сагоревања код мотора са директним убризгавањем*

Млазом вођени процеси имају највећи потенцијал за искоришћење предности слојевитог пуњења и повећање економичности мотора.

2. ИСТОРИЈАТ РАЗВОЈА И СПЕЦИФИЧНА КОНСТРУКЦИЈСКА РЕШЕЊА

Историја Ото мотора са директним убризгавањем бензина дели се на неколико етапа, од периода основних идеја и концепција, преко првог Ото мотора 1878. године са основним деловима конструкције и законима измене притиска у цилиндру сличним данашњим моторима [4], до последњег периода са убрзаном применом електронике у управљању процесима мотора СУС.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији је ментор био др Јован Дорић, ванр. проф.

Daimler Benz је представио четвороцилиндарни, тролитарски мотор са директним убризгавањем и два вентила по цилиндру почетком 1954. године. За разлику од данашњих електричних, овде се користила механичка пумпа за гориво. Због учесталих проблема по питању стварања филма горива у цилинду, клипу мотора, као и спирања уљног филма, поготово приликом старта мотора, у доњем кућишту је складиштено чак 15 литара уља како би мотор избегао прегревања приликом учесталих кратких релација.

Године 1996., аутомобилска компанија *Mitsubishi* представља *GDI* концепт директног убризгавања горива, и то са 1.8 литарским *4G93* мотором, који се одликовао са 4 вентила по цилиндру и два брегаста вратила постављена изнад коморе сагоревања. Неки модели су имали и променљив рад вентилског склопа *MIVEC*.

Године 2000., *Volkswagen* је развио 1.4 литарски *FSI* мотор са директним убризгавањем, регулисан од стране *Bosch Motronic Med 7* јединице, која управља оптималним решењима за образовање мешавине горива и ваздуха, пратећи константно различите режиме рада и оптерећења мотора [2]. Омогућено је раслојавање и стварање хомогене смеше, дупло убризгавање по такту ради повећања обртног момента и температуре издувних гасова приликом рада са сиромашном смешом.

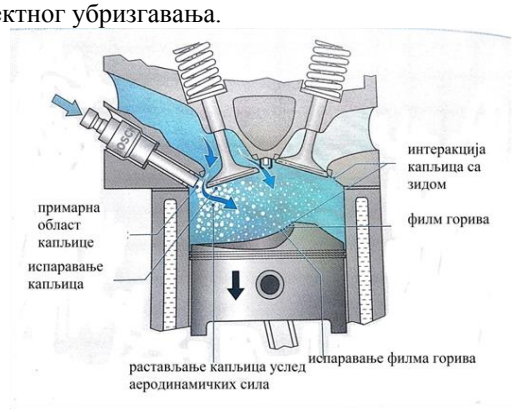
Mercedes-Benz 2007. године развија директно убризгавање горива у модел *CLS 350 CGI*, са првим пијезоелектричним бризгачима и млазом вођеним процесом.

3. КОНЦЕПТ МОТОРА СА ДИРЕКТНИМ УБРИЗГАВАЊЕМ БЕНЗИНА У КОМОРИ ЗА САГОРЕВАЊЕ

Bosch Motronic Med систем, управљајући радом мотора, повећава динамику вожње уз очување или смањење радне запремине, где се уз електронску савремену контролу управљања постиже ефикасан процес сагоревања у зависности од броја обртаја, оптерећења и режима рада мотора [2]. При нижим обртајима мотора (слојевити мод), где је брзина усисног ваздуха мања, неопходно је обезбедити ефикасније мешање горива и ваздуха, што се обезбеђује интензивнијим вртложењем помоћу вентила у усисној грани. Како је краћи период од места убризгавања и упаљења горива, битна функција вентила вртложења омогућава остварење стабилног мешања радне смеше, сагоревања и ниске емисије штетних материја приликом рада у слојевитом моду, што за резултат има циљану нижу потрошњу горива. Под одређеним условима, ради ефикасније криве сагоревања, у састав свежег ваздуха додаје се и одређена количина издувних гасова помоћу вентила за рециркулацију издувних гасова, чијим прецизним радом управља ЕУЈ [2].

3.1. Директно убризгавање са хомогеном смешом

Током рада мотора са директним убризгавањем и хомогеним модом, одликују се режими виших бројева обртаја са већим обртним моментом и снагом. Хомогенизација се постиже због довољног времена за припрему смеше која је присутна код убризгавања у раним фазама такта усисавања (слика 2), када се клип креће ка УМТ, како се одвија и код индиректног убризгавања. Смеша има сличну концентрацију, уз боље излазне параметаре мотора (снаге и момента), као и редукције емисије штетних честица у корист директног убризгавања.



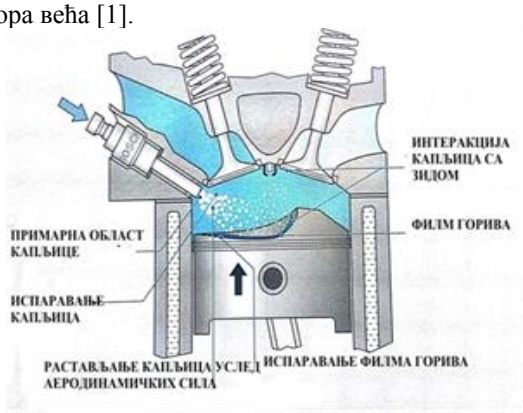
Слика 2. *Убризгавање у такту усисавања*

Предности директног у односу на индиректно убризгавање се по питању степена сабијања може објаснити следећим факторима:

- повећано је хлађење смеше услед целокупног испаравања у комори сагоревања,
- мања подложност ка детонантном сагоревању,
- постојање изолирајућег слоја ваздуха,
- могућност бржег образовања смеше услед виших релативних брзина и притисака.

3.2. Директно убризгавање са слојевитом смешом

Други мод формирања смеше горива и ваздуха је слојевити мод, са задатком остваривања специфичних услова рада у нижим и делимичним отптерећењима, са краћим временом формирања смеше. Гориво се убризгава током процеса сабијања (слика 3). Смеша је веома сиромашна, раслојена, па је економичност мотора већа [1].



Слика 3. *Убризгавање у такту сабијања*

Неопходно је да увек у близини свећице буде концентрација радне материје која ће омогућити сигурно и стабилно упаљење електричном варницом, тј. да упаљиви део смеше буде у близини електроде свећице у тачки паљења (λ=1). У области мало даље од свећице је веома сиромашна (λ=3-5), али и даље запаљива смеша, чист ваздух, или издувни гасови из предходног циклуса [1].

Зидом вођеним процесом, образовање и кретање смеше од бризгача до свећице одвија се посредством зида коморе цилиндра, а чело клипа са својом конструкцијом специјалног удубљења, обликованог као усме-

равач млаза горива, потпомаже да се очува слој са смешом од експанзије по радном простору у цилиндру [1].

Тешко је одвојити процес вођења зидом и ваздухом, јер су увек у већој или мањој мери присутна оба процеса [1].

За разлику од зидова вођених процеса, контакт између горива и зидова коморе се избегава ради смањења токсичности, и у идеалним случајевима не постоје депозити горива на зидовима коморе за сагоревање. Вртлог ваздуха ствара ваздушни јастук који спречава поменути контакт чела клипа и горива.

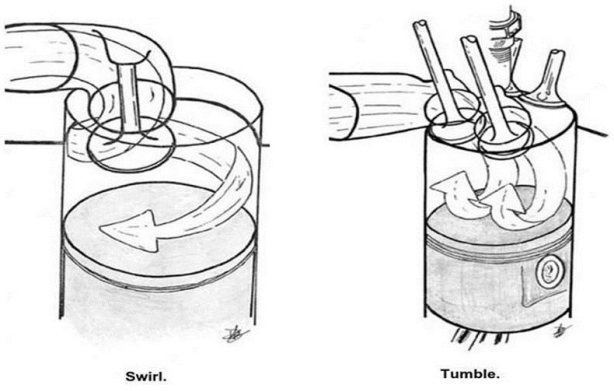
Код млаза вођених процеса, путања убризгавања и кретања горивог слоја која иде директно према свећици је знатно краћа [5]. Највећи проблеми мотора овог типа, са највећим потенцијалом искоришћења предности слојевитог пуњења, јесу релативно мала растојања бризгача и свећице, позиционираних између вентила [5].

Чело клипа је конструисано да кретањем ваздуха чува слој са смешом.

4. КАРАКТЕРИСТИКЕ СМЕШЕ ГОРИВА И ВАЗДУХА

Један од најбитнијих процеса код мотора СУС је процес образовања смеше, где долази до мешања ваздуха и горива, при чему се формира упаљива смеша која ће сагоревати у циљу добијања корисног рада у такту ширења.

Подешавање образовања смеше зависи од броја обртаја мотора, радне температуре, постигнуте величине притиска и геометрије убризганог горива, интензитета вртложења усисног ваздуха, облика коморе сагоревања, степена сабијања.
Различито усмерено кретање пуњења на улазном каналу може бити реализовано на два начина ротационог усмерења. Код вртлога (*swirl*), оса ротације је паралелна са осом цилиндра. Код другог вида (*tumble*), оса ротирања је управна на осу цилиндра (слика 4).



Слика 4. *Различито усмерено кретање пуњења на улазном каналу лево (swirl) и десно (tumble).*

Након убризгавања горива, фаза образовања смеше је одређена кидањем млаза, формирањем капљица, испаравањем капљица горива и мешањем са ваздухом.
Кидање млаза се може постићи формирањем капљица, таласа и атомизацијом.

5. ТРЕТМАН ИЗДУВНИХ ГАСОВА И КАРАКТЕРИСТИЧНА РЕШЕЊА

Садржај токсичних компоненти у продуктима сагоревања зависи од састава смеше, угла претпаљења и конструкције мотора. У зависности од тога да ли је смеша богата, сиромашна или негде између, јавља се у одређеној мери концентрација CO и CxHy [1].

5.1 Емисија

Поред свих наведених предности директног у односу на индиректно убризгавање, недостаци у виду неповољне емисије HC и NOx, представљају проблеме са третманом издувних гасова, поготово у режимима нижих обртаја.
Није довољан више само конвенционалан тростепени каталитички неутрализатор, који је и даље ефикасан у хомогеном моду са стехиометријском смешом, већ је неопходан додатни конвертор за неутрализацију азотових оксида, који настају на граници између горивог и негоривог слоја, где је смеша сиромашна [1].
Проблем високе температуре издувних гасова може бити решен смањењем протока свежег ваздуха помоћу делимичног пригушења, уз веће струјне губитке. Овај принцип не представља адекватно решење, као што је постигнуто рециркулацијом издувних гасова, тј. са ЕГР вентилом, где се одвија мешање издувних гасова са свежим пуњењем за време такта усисавања.

5.2 Истраживање на екстремним притисцима убризгавања

Повећање притиска свежег пуњења доводи до увећаног оптерећења компоненти мотора, а ради одржања поузданог рада, неопходно је извршити увећање димензије мотора, масе као и примену квалитетнијих материјала [1]. Увећани топлотни губици који су предати зидовима цилиндра утичу негативно на поузданост рада мотора, погоршавањем услова подмазивања уљем и другим факторима који су ограничавајућег карактера за имплементацију повећаног притиска убризгавања горива у цилиндар, који би обезбедили ефикаснији процес рада мотора [1].

Остварени су притисци убризгавања од око 200 бара, услед ограничене издржљивости система убризгавања (200 бара код млазом вођеним, а до 150 бара код зидом и ваздухом вођеним процесима) [1]. Испитивања са повећањем притиска око 600 бара спроведена су у лабораторијским условима, где је гориво убризгивано у специјалну комору. Резултати су показали да је са екстремним притисцима остварено смањење величине капљица горива, потрошње горива и формирања чађи.

5.3 Упаљење и ток сагоревања сиромашне смеше

Системи паљења морају задовољити посебне захтеве код слојевитог мода, као што су: стабилно место упаљења веома сиромашне смеше, неосетљивост на течну фазу горива тј. квашење чела клипа и неосетљивост на депозите. Високе вредности брзине струјања пуњења у близини електрода свећице додатно отежавају формирање језгра пламена.

Квашење свећице горивом не може се искључити због кратког времена за образовање смеше током слојевитог мода пуњења [1]. Због великог имплуса млаза горива у

области паљења, посебно у слојевитом моду, долази до великих брзина струјања у процепу између електрода свећице. Главни недостатак сагоревања је тај да је за смањење емисије NOx потребан сложени каталитички систем претварача.

5.4 Детонантно сагоревање

Детонантно сагоревање се јавља у оним деловима простора за сагоревање који су далеко од свећице, а близу загрејаних зидова цилиндра [1]. Осим рада код стабилног тока сагоревања, где су испуњени услови за правилно простирање фронта пламена, постоји и други вид сагоревања а то је ненормално сагоревање, које може бити [3]:

- детонантно
- површинско (неконтролисано)
- дисоцијација продуката сагоревања.

Мотори са директним убризгавањем могу радити са већим степеном сабијања у односу на конвенционалне моторе. Детонантно сагоревање је карактеристично и по томе што се, за разлику од нормалног сагоревања и простирања фронта пламена, овде јавља самопаљење као код дизел мотора на једној или више локација у комори сагоревања, претежно при крају такта сабијања. При крају такта сабијања код СМТ, спонтано паљење неупаљене смеше резултира брзом конверзијом горива, уз јављање јаког таласа притиска, механичких оштећења и манифестације звука сличном ударању метал о метал [3].

6. ОТКЛАЊАЊЕ ГРЕШАКА ДИЈАГНОСТИЧКИМ УРЕЂАЈИМА

Аутомобили савремених генерација опремљени су модулима електронске контроле управљања на готово свим системима моторног возила. Управља се мотором, трансмисијом, управљачем, системима активне и пасивне безбедности (АБС, АСР, ЕСП, ваздушним јастуцима). Рад савремених возила контролише електронска управљачка јединица, која информише возача у случају неисправности, режиму рада појединих система (укључен и искључен систем против проклизавања, деактивација одређеног ваздушног јастука), тако да је возило већ опремљено сопственом дијагностичком опремом. Сигнализација која упозорава возача може се уочити светлећим, звучним или комбинованим сигналом.

Опрема за електронску дијагностику (комуникациони уређаји), доступна на тржишту, а дели се у три категорије:

- опрема ниског квалитета,
- опрема доброг квалитета (ограничене функције) и
- професионална опрема за ауто-сервисе.

Три основна статуса грешака су:

- АТТ - стална грешка тј. сигуран квар, односно неисправност,
- МЕМ - меморисана грешка тј. није активна али са повременим појављивањем и
- грешка коју систем није једнозначно препознао.

7. ЗАКЉУЧАК

У последњих 25. година, дошло је до велике експанзије и константног усавршавања система директног убризгавања горива, који у све већој мери замењује конвенционалне индиректне моторе. Циљ данашње експанзије развоја ових система је технолошка борба ауто-компанија, како би понудили на тржишту што ефикаснији, економичнији и поузданији мотор, са циљем што већег профита.

Даљи развој Ото мотора иде ка усавршавању постојећих решења са акцентом на истраживање високих притиска убризгавања и до три пута већим него постигнутим до сада у серијској производњи. Будућа решења представљају изузетно важан корак за функционисање у слојевитом моду и за остварење што већег потенцијала рада са сиромашном смешом.

8. ЛИТЕРАТУРА

[1] Јован Дорић, *Теорија мотора СУС*, Нови Сад, 2015.

[2] Robert Bosch GmbH: *Gasoline-Engine Management*, 2006.

[3] Клинар Ј. Иван, *Мотори са унутрашњим сагоревањем*, Нови Сад, 2013.

[4] Ратко Николић, Лазар Савин, Мирко Симикић, Милан Томић, *Погонске машине мотори СУС*, Нови Сад, 2014.

[5] Robert Bosch GmbH: *Gasoline-Engine Management*, 2001.

Кратка биографија:

Иван Милојковић, рођен у Пожаревцу 1993. године, након завршене средње школе у Великом Градишту, своје образовање наставља на Војној Академији у Београду, где је 2016. године стекао звање дипломираног инжењера машинства.

контакт: imilojkovic62@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK:539.3
DOI: https://doi.org/10.24867/12AM04Petkovic

ANALITIČKI I NUMERIČKI POSTUPCI ANALIZE NAPONA I DEFORMACIJA U CILINDRIČNIM SUDOVIMA POD PRITISKOM

ANALYTICAL AND NUMERICAL PROCEDURES FOR STRESS AND STRAIN ANALYSIS IN CYLINDRICAL PRESSURE VESSELS

Strahinja Petković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – MAŠINSKO INŽENJERSTVO

Kratak sadržaj – *U ovom radu analizirana su naponska i deformacijska stanja cilindričnih sudova pod pritiskom. Različiti slučajevi dobijeni su odabirom različitih modela cilindričnih sudova koji se najčešće sreću u praktičnim primenama. Izvršena je detaljna analiza i poređenje teorijskih i numeričkih rezultata za slučajeve kada je cilindrični sud opterećen unutrašnjim pritiskom. Takođe, izvršena je analiza uticaja toplotnog opterećenja na raspodelu karakterističnih napona i deformacija. Analitička rešenja dobijena su na osnovu klasične teorije za beskonačno dugačke elastične cevi konačne debljine i slučaj ravanskog stanja deformacije. Posebna pažnja posvećena je analizi napona i deformacija u okolini spojeva (poklopaca) i njihovom poređenju sa rezultatima za otvoreni cilindrični sud. Za modele realnih sudova, približna numerička rešenja dobijena su korišćenjem aplikacije Solidworksi adekvatne numeričke metode konačnih elemenata (MKE). Poređenje rezultata i prikaz rešenja dat je grafički u vidu dijagrama i odgovarajućih 3D modela.*

Ključne reči:*sudovi pod pritiskom, teorija elastičnosti, MKE metode.*

Abstract – *In this paper, the stress and strain of cylindrical pressure vessels are analyzed. Different cases were obtained by selecting different models of cylindrical vessels that are most commonly encountered in practical applications. A detailed analysis and comparison of theoretical and numerical results for the cases when the cylindrical vessel is loaded with internal pressure was performed. Also, the influence of the thermal load on the distribution of characteristic stresses an deformationis given. Analytical solutions were obtained on the basis of the classical theory for infinitely long elastic tubes of finite wall thickness and in the case of the plane state of deformation. Special attention is dedicated to the analysis of stresses and strains in the vicinity of joints (caps). For real world models, approximate numerical solutions were obtained using Solidworks application and adequate FEM procedure. The comparison of results and presentation of solutions is given graphically in the form of diagrams and corresponding 3D models.*

Keywords: *pressure vessels, theory of elasticity, FEM.*

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Damir Mađarević, van. prof.

1. UVOD

Uređaji i konstrukcije (rezervoari, cevovodi, silosi, kotlovi, reaktori) koji prenose, skladište ili primaju fluide (tečnosti i gasove) se nazivaju sudovi pod pritiskom [1,2]. Unutrašnji pritisak kod ovakvih konstrukcija je obično veći od spoljašnjeg pritiska. Sudovi pod pritiskom često su izloženi visokom pritisku kao i visokoj temperaturi, a u nekim slučajevim zapaljivim fluidima kao i radioaktivnim materijalima. Takođe, fluid u sudu može menjati agregatno stanje kao što je to slučaj kod parnih bojlera i kotlova. Zbog velike opasnosti veoma je važno da se sudovi pod pritiskom konstruišu tako da se predvidi svaka mogućnost nastanka havarije u vidu curenja ili ekstremnih situacija (eksplozija), odnosno da se projektuju tako da mogu podneti visoke radne temperature i pritiske. Veličina i oblik sudova pod pritiskom varira od velikih cilindričnih sudova koji služe sa skladištenje gasova pod velikim pritiskom do malih hidrauličnih cilindara na avionima za upravljanje elementima krila i točkova. Neki se nalaze smešteni duboko ispod površine zemlje ili okeana, dok se većina koristi na površini zemlje. Sudovi pod pritiskom su obično sfernog ili cilindričnog oblika.
Cilindrični sudovi se najčešće prave tako što se limovi odgovarajuće debljine savijaju i spajaju u konačni oblik. Spajanje se vrši zavarivanjem pod strogo kontrolisanim uslovima. Cilindrični sudovi pod pritiskom manjih dimenzija izrađuju se tako što se na cev određene dužine na oba kraja zavare završeci (krajevi, poklopci) koji mogu biti različitog oblika (ravni ili zaobljeni poklopci). U praksi se sudovi pod pritiskom najčešće izrađuju u obliku cilindričnog suda zbog lakše proizvodnje i boljeg iskorišćenja radnog prostora, iako je poznato da sferni sudovi imaju povoljnije naponsko stanje tj. bolje podnose radno opterećenje (kotlovi, izmenjivači toplote, hemijski reaktori).

Slika 1. *Primeri cilindričnih sudova*

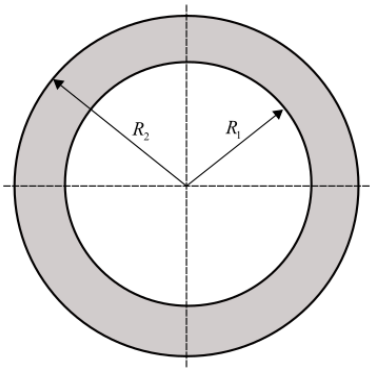
U radu je sprovedena detaljna analiza analitičkih rezultata dobijenih korišćenjem klasične teorije za beskonačno dugačke elastične cevi konačne debljine i slučaj ravanskog stanja deformacije [3]. Rezultati teorijske analize su upoređeni sa pažljivo pripremljenim numeričkim rezultatima. Numerička analiza sprovedena je korišćenjem metode konačnih razlika i metode konačnih elemenata [4].

Sve analize i poređenje rezultata izvršene su za dva karakteristična slučaja radnog opterećenja:

1. Unutrašnji radni pritisak (1Mpa -10 bara)
2. Toplotno opterećenje (razlika temperatura unutrašnjeg i spoljašnjeg zida suda ΔT=3°C, temperatura radnog fluida iznosi 90°C).

Analizirano je nekoliko karakterističnih konstriktivnih slučajeva (Slika 2):

1. Otvoreni cilindrični sud (teorijski slučaj - beskonačno dugačka cev),
2. Cilindrični sud sa ravnim poklopcima,
3. Cilindrični sud sa hemisferičnim poklopcima,
4. Sferni sud,



Slika 2: *Konstruktivne mere analiziranih sudova pod pritiskom: unutrašnji poluprečnik*: R_1=250mm, *spoljašnji poluprečnik:*R_2=300mm.

Dužina svih analiziranih sudova iznosi L=1,5m (Slika 3).

U svakom od prethodno nabrojanih slučajeva izvršene su sledeće analize:

- analiza radijalnih i cirkularnih napona u zidu suda,
- analiza deformacija suda (radijalnog pomeranja)
- termo-mehanička analiza (analiza napona i deformacija pod dejstvom unutrašnjeg pritiska i toplotnog opterećenja).

Termo-mehaničke analize koriste se za proučavanje toplotnih opterećenja, toplotnog fluksa, napona i deformacija. Proučavanje toplotnih opterećenja sudova pod pritiskom je od velikog značaja imajući u vidu da pored toga što sudovi služe za skladištenje i transport fluida čija se temperatura značajno razlikuje od spoljašnje temperature ambijenta, velike temperaturne promene mogu nastati prilikom punjenja i pražnjenja suda.

Postoje dve vrste ovih analiza: statičke i dinamičke (stacionarne i tranzijentne). U cilju lakšeg poređenja uticaja različitih vrsta opterećenja i rezultata, u radu je sprovedena stacionarna termo-mehanička. Ona je u skladu sa stacionarnim mehaničkim opterećenjem u vidu konstantnog unutrašnjeg hidrostatičkog pritiska.

Poređenje sa teorijskim rezultatima za otvoreni cilindrični sud izvršeno je na sredini suda i u neposrednoj blizini spoja sa odgovarajućim poklopcima.

U poglavlju 2 ovog rada dat je kratak prikaz teorijskih postavki ravanske analize deformacija sa naponskom analizom. U praksi se prilikom projektovanja sudova pod pritiskom analiziraju naponi i deformacije korišćenjem osnovne teorije za tankozidne cevi opterećene unutrašnjim pritiskom. Iz tog razloga u uvodnom delu opisana je naponska i deformacijska analiza tankozidnih sudova pod pritiskom. U nastavku je dat prikaz teorije koja se koristi pri analizi. sudova pod pritiskom konačne debljine zida.

U poglavlju 3 dat je prikaz numeričkih rešenja i prikaz poređenja sa teorijskim rezultatima.

U poslednjem delu ovog rada dat je kratak zaključak rada sa smernicama za dalji rad.

2. TEORIJSKE OSNOVE SUDOVA POD PRITISKOM

Do loma konstrukcija dolazi na mestima lokalne koncentracije napona tj. kada vrednosti ekvivalentnog napona prekorače dozvoljene vrednosti napona materijala suda. Kod sudova pod pritiskom se to najčešće dešava na mestima geometrijskih singulariteta: spojevi sa drugim elementima i zavareni spojevi. Kako su sigurnost i bezbednost funkcionisanja kod ovakvih konstrukcija od presudne važnosti, pred inženjere se postavljaizazov modeliranja i određivanja napona i deformacija koji se mogu očekivati u praktičnim uslovima.

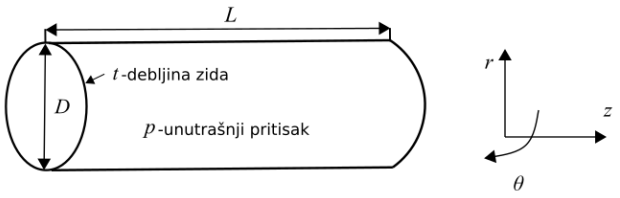
Većina standardnih proračuna sudova pod pritiskom zasniva se na korišćenju jednostavnih izraza za određivanje napona i deformacija za slučaj statičkog opterećenja i činjenice da je debljina suda mala u odnosu na dimenzije suda.

Međutim, uzimanjem u obzir debljine suda, analiza je složenija ali nam omogućava precizniji uvid u promenu karakterističnih napona u unutar zida suda.

U slučaju da cilindrični sud opteretimo konstantnim unutrašnjim pritiskom p imaćemo prostorno naponsko stanje u zidu suda gde su sa označeni naponi σ_r radijalni napon, σ_z aksijalni napon, a σ_θ cirkularni napon.

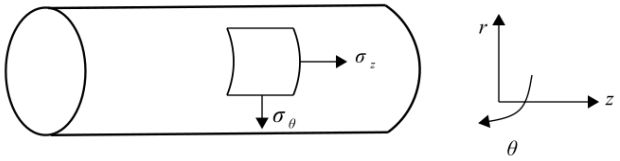
2.1. Tankozidni cilindrični sudovi pod pritiskom

Teorijska analiza napona kod tankozidnih cilindričnih sudova pod pritiskom podrazumeva zanemarivanje uticaja težine suda kao i težina fluida u sudu.

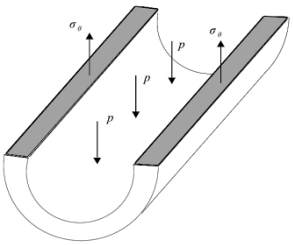


Slika 3: *Cilindrični sud pod pritiskom*

Na Slici 3 sa r je označen radijalni pravac, θ je cirkularni pravac, z aksijalni pravac, a L predstavlja dužinu suda.$t$označava debljinu suda, D unutrašnji prečnik suda, p predstavlja unutrašnji pritisak.



Slika 4: *Naponi u tankozidnom cilindričnom sudu*



Slika 5: *Tankozidni sud - određivanje cirkularnog napona*

Iz uslova ravnoteže spoljašnjeg opterećenja i unutrašnjih sila određujemo cirkularni napon:

$$DLp = 2\sigma_\theta Lt\,, \tag{1}$$

$$\sigma_\theta = \frac{pD}{2t}\,. \tag{2}$$

Na sličan način analizom ravnoteže duž ose suda, dobijamo izraz za aksijalni napon:

$$\sigma_z = \frac{pD}{4t}\,. \tag{3}$$

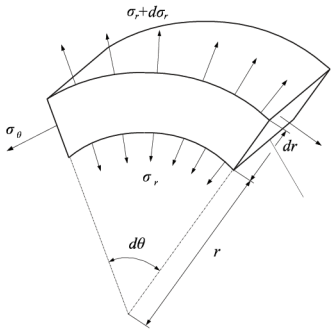
Kod tankozidnih sudova pod pritiskom prečnik suda je mnogo veći od debljine cilindra $D >> t$, pa se može zaključiti da je $\sigma_\theta, \sigma_z >> \sigma_r$, tako da vrednost σ_r je zanemarljiva, odnosno da je u ovom slučaju naponsko stanje ravansko (Slika 4).

U nastavku rada, aksijalni napon σ_z nije bio predmet analize.

2.2. Sudovi pod pritiskom konačne debljine zida

Posmatrajmo elementarni segment cilindičnog suda konačne debljine zida na Slici 7. Na osnovu analize napona možemo zapisati jednačinu ravnoteže u radijalnom pravcu:

$$(\sigma_r + d\sigma_r)(r + dr)d\theta \times 1 - \sigma_r \times rd\theta \times 1 = \\ = 2\sigma_\theta \times dr \times 1 \times \sin\frac{d\theta}{2} \tag{4}$$



Slika 6: *Analiza radijalnog napona*

Zanemarivanjem članova drugog reda jednačina (4) se može napisati na sledeći način:

$$\sigma_\theta - \sigma_r = r\frac{d\sigma_r}{dr}\,. \tag{5}$$

Imajući u vidu pretpostavku o ravanskom stanju deformacije ε_z=0, prethodna diferencijalna jednačina se može napisatu u sledećem obliku:

$$\frac{d}{dr}\left(\sigma_r r^2 - Ar^2\right) = 0\,. \tag{6}$$

Nakon integracije dobijamo:

$$\sigma_r r^2 - Ar^2 = \text{const.} = -B\,. \tag{7}$$

odnosno raspored radijalnog i cirkularnog napona u zidu suda:

$$\sigma_\theta = A + \frac{B}{r^2}\,, \qquad \sigma_r = A - \frac{B}{r^2}\,, \tag{8}$$

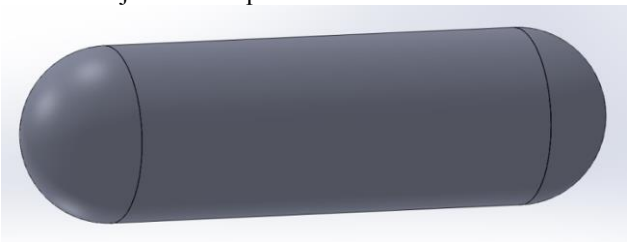
gde konstante A i B određujemo iz uslova opterećenja suda.

3. ANALIZA NAPONA I DEFORMACIJA ZA CILINDRIČNI SUD KONAČNE DEBLJINE SA HEMISFERIČNIM POKLOPCIMA

Za prikaz rezultata rada izabran je slučaj cilindričnog suda sa hemisferičnim poklopcima gde smo izvršili detaljnu FEM analizu i prikazali napone u poprečnim presecima na sredini suda i neposredno u blizini hemisferičnih poklopaca.

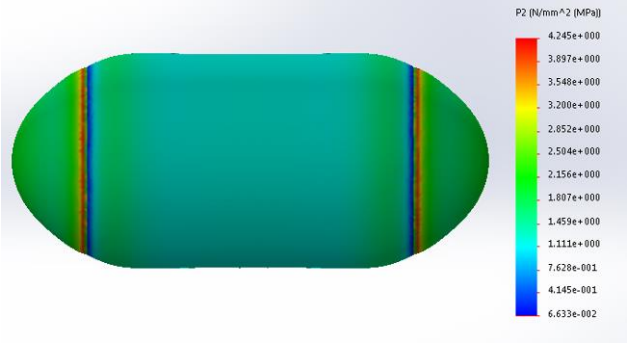
Numerička analiza je sprovedena na sledeći način:

- kreiranje geometrije sudova pod pritiskom,
- definisanje svojstava materijala elemenata konstrukcije,
- diskretizacija geometrije - generisanje adekvatne prostorne mreže imajući u vidu osno-simetričnost elemenata konstrukcije suda,
- postavljanje različitih graničnih uslova i uslova opterećenja,
- rešavanje modela i prikaz rezultata.

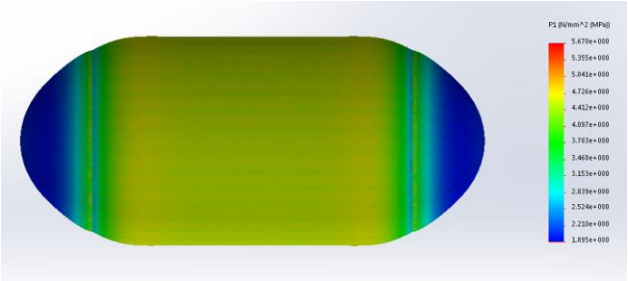


Slika 7: *Geometrija cilindričnog suda sa hemisferičnim krajevima*

Na Slici 8 prikazan je prostorni raspored radijalnih napona pod dejstvom unutrašnjeg pritiska p gde je jasno uočljiva pojava koncentracije napona u delu hemisferičnog poklopca dok cirkularni naponi u ovom slučaju imaju ravnomerniju raspodelu (Slika 9). Ekstremne vrednosti napona date su su u Tabeli 1.



Slika 8: *Radijalni napon za unutrašnji radni pritisak* p_1=1MPa



Slika 9: *Cirkularni napon za unutrašnji radni pritisak* p_1=1MPa

Tabela 1: *Ekstremne vrednosti napona u cilindričnom sudu sa hemisferičnim krajevima p=1MPa*

p_1=1MPa	σ_r	σ_c
min.	0,066	1,895
max.	4,245	5,670

U Tabeli 2 prikazano je poređenje maksimalnih i minimalnih vrednosti napona u karakterističnim presecima sa teorijskim vrednostima za beskonačno dugačku cev.

Tabela 2: *Vrednosti napona u cilindričnom sudu sa hemisferičnim krajevima za različite slučajeve opterećenja*

	Teorijska analiza (MPa)		Numerička analiza (MPa)	
p=1MPa	σ_r	σ_c	σ_r	σ_c
Sredina suda (min.)	-1	4,545	-0,96	4,534
Sredina suda (max.)	0	5,545	-0,0164	5,567
Spoj poklopca (min.)	-1	4,545	-0,975	3,637
Spoj poklopca (max.)	0	5,545	-0,034	4,253
p=1MPa +ΔT=3°C	σ_r	σ_c	σ_r	σ_c
Sredina suda(min.)	-1	-0,659	-2,208	1,048
Sredina suda (max.)	0	10,041	0,209	8,478
Spoj poklopca (min.)	-1	-0,659	-2,943	-0,636
Spoj poklopca (max.)	0	10,041	0,307	7,622

4. ZAKLJUČAK I PRAVCI DALJIH ISTRAŽIVANJA

U radu su analizirana naponska i deformacijska stanja cilindričnih sudova pod pritiskom. Izvršena je detaljna analiza i poređenje teorijskih i numeričkih rezultata za slučajeve kada je cilindrični sud opterećen unutrašnjim pritiskom i toplotnim opterećenjem od zagrejanog radnog fluida. Numerička analiza sprovedena je korišćenjeg programskih paketa *Mathematica* i *Solidworks.*

Na osnovu poređenja rezultata u dva karakteristična poprečna preseka suda (sredina i neposredna blizina spoja sa poklopcima) utvđeno je da se teorijske vrednosti napona ne odstupaju značajno od numeričkih tj. da se teorijski proračuni mogu koristiti za kvalitativnu ocenu stvarnog opterećenja suda. Kada je u pitanju uticaj toplotnog opterećenja, čak i pri minimalnim temperaturnim razlikama spoljašnjeg i unutrašnjeg zida suda, vrednosti napona u ovom slučaju se značajno razlikuju od teorijskih vrednosti, što prilikom projektovanja treba uzeti u obzir.

5. LITERATURA

[1] A.P. Boresi, R.J. Schmidt, *"*Advanced mechanics of materials*"*, New York, Wiley, 1993.

[2] S. Chattopadhyay, *"*Pressure Vessels*"*, Boca Raton, CRC Press, 2004.

[3] M. H. Sadd - *"*Elasticity: Theory, Applications, and Numerics*"* 1st Edition 2005.

[4] A. Kaw, S. Ho, *"*On Introducing Approximate Solution Methods in Theory of Elasticity*" Wiley Periodicals, Inc. Comput Appl Eng Educ Vol 14: pp. 120-134, 2006; DOI 10.1002/cae.20070*

Kratka biografija:

Strahinja Petković rođen je u Novom Sadu 1991. godine. Diplomski (bečelor) rad na Fakultetu tehničkih nauka iz oblasti Mašinstva - Tehnička mehanika i dizajn u tehnici odbranio je 2017.godine.
kontakt: strale0000@yahoo.com

Damir Mađarević rođen je u Novom Sadu 1981. Doktorirao je na Fakultetu tehničkih nauka 2013. godine., a od 2020. godine je zvanju vanredni profesor. Oblast interesovanja: termo-mehanika, prostiranje talasa u gasovima

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 531/534(043.3)
DOI: https://doi.org/10.24867/12AM05Smiljanic

OSCILOVANJE I STABILNOST AKSIJALNO OPTEREĆENE I UKLEŠTENE PRAVOUGAONE PLOČE NA NEHOMOGENOJ ELASTIČNOJ PODLOZI

VIBRATION AND STABILITY OF AN AXIALLY LOADED CLAMPED RECTANGULAR PLATES ON A NON-HOMOGENEOUS ELASTIC BASE

Radivoj Smiljanić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – MAŠINSTVO - TEHNIČKA MEHANIKA

Kratak sadržaj – *U radu je analiziran način nalaženja frekvencije oscilovanja pravougaone ploče kao i vrednost kritične sile pri kojoj dolazi do gubitka stabilnosti ploče. Nađene osnovne jednačine postavljenog problema su prebačene u bezdimenzijski oblik i kao takve numerički rešavane. Dobijeni su rezultati kružne frekvencije oscilovanja ploče kao i rezultati kritične sile pri kojoj dolazi do savijanja ploče.*

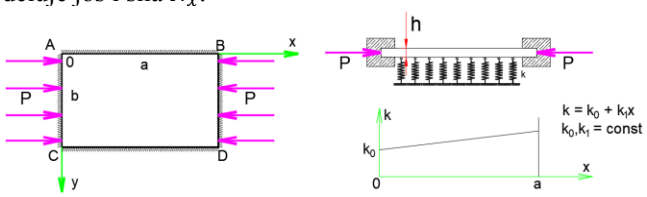
Ključne reči: *oscilovanje i stabilnost ploče, kružna frekvencija, kritična sila*

Abstract – *The paper analyzes the way of finding the oscillation frequency of a rectangular plate as well as the value of the critical force at which it comes to the loss of plate stability. The basic equations of the posed problem were transferred to the dimensionless form and as such numerically solved. The results of the natural frequency of the plate as well as the results of the critical force at which the plate bending occurs were obtained.*

Keywords: *vibration and stability of plates, natural frequency, critical force*

1. UVOD

Tema ovog rada je analiza problema nalaženja frekvencije oscilacija pravougaone ploče kao i vrednost kritične sile pri kojoj dolazi do gubitka stabilnosti ploče. Sistem, prikazan na slici 1, sastoji se od pravougaone ploče, stranica a i b, koja je ukleštena sa sve četiri strane. Osim uklještenja, ploča je oslonjena na elastičnu podlogu čija se krutost podloge menja prema jednačini $k = k_0 + k_1$, gde su $k_0, k_1 = const$. Na posmatranu ploču, duž stranice b, deluje još i sila N_x.



Slika 1. *Pravougaona ploča*

U ovom radu prikazaće se matematički model za rešavanje zadatog problema ploče.

Odrediće se konačni oblik diferencijalne jednačine za postavljeni problem kao i odgovarajući granični uslovi. Zatim će se pomenute jednačine prebaciti u bezdimenzijski oblik i kao takve rešavati pomoću softvera Mathematica. Za razne, proizvoljno odabrane, parametre predstaviće se rešenje postavljenog problema, tj. prikazaće se za koje vrednosti parametara dolazi do gubitka stabilnosti ploče. Suština rada je da se pokaže kakav uticaj na stabilnost ploče imaju kružna frekvencija, sila u oprugama i sila P koja deluje u pravcu osce x.

Cela analiza odradiće se za ploču kvadratnog oblika (kada je odnos stranica $\frac{a}{b} = 1$) i ploču pravougaonog oblika (kada je odnos stranica $\frac{a}{b} = \frac{1}{2}$). Rešavanje sličnih problema može se videti u radovima [1]-[5].

2. MATEMATIČKI MODEL AKSIJALNO OPTEREĆENE I UKLEŠTENE PLOČE POSTAVLJENE NA ELASTIČNU PODLOGU

Za postavljeni problem koji je prikazan na slici 1, potrebno je odrediti frekvencije oscilacija i kritičnu silu pri kojoj dolazi do gubitka stabilnosti ploče.

Bitne stvari koje je potrebno uočiti sa slike 1, su da je ploča (stranica a i b) sa sve četiri strane ukleštena i da je oslonjena na elastičnu podlogu. Zatim se vidi da na ploču u pravcu ose x deluju određene sile P. Da bi se odredila frekvencija oscilacija i kritična sila pri kojoj dolazi do gubitka stabilnosti, neophodno je izvesti diferencijalnu jednačinu koja opisuje postavljeni problem. Pa tako, polazimo od osnovne diferencijalne jednačine savijanja ploče koja je poprečno opterećena, jednačina (1).

Diferencijalna jednačina savijanja ploče koja je poprečno opterećena ima oblik:

$$D\nabla^4 w = q \qquad (1)$$

U ovoj jednačini D predstavlja savojnu krutost, ∇^4 bilaplasijan od w(w - ugib ploče), dok je sa$q$ predstavljen intenzitet neprekidno podeljenog opterećenja. Savojna krutost D, data je jednačinom:

$$D = \frac{Eh^3}{12(1-\nu^2)} \qquad (2)$$

gde h označava debljinu ploče, ν Poisson-ov koeficijent, a E Young-ov modul elastičnosti. ∇^4 bilaplasijan od w je dat jednačinom:

$$\nabla^4 w = \frac{\partial^4 w}{\partial x^4} + 2\frac{\partial^4 w}{\partial x^2 y^2} + \frac{\partial^4 w}{\partial y^4} \qquad (3)$$

Desna strana znaka jednakosti jednačine (1) tj. intenzitet neprekidno podeljenog opterećenja q predstavlja sumu svih opterećenja koja deluju na posmatranu ploču. Oscilovanje ploče, sile koje deluju uz celu dužinu stranice b kao i sile u oprugama, su opterećenja koja utiču nastabilnost poče. Navedena opterećenja prikazaće se kroz naredne jednačine.

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

Osnovna diferencijalna jednačina oscilovanja ploče koja se dobija primenom Dalamberov-og principa ima oblik:

$$D\nabla^4 w = -\rho h\ddot{w} \tag{4}$$

Opšta jednačina pomoću koje se opisuju problemi prikazani na slici 2 je oblika:

$$D\nabla^4 w = -\rho h\ddot{w} - cw + N_x\frac{\partial^2 w}{\partial x^2} + 2N_{xy}\frac{\partial^2 w}{\partial x\partial y} + N_y\frac{\partial^2 w}{\partial y^2} \tag{5}$$

Sa slike 2 vidi se da sledeći članovi nemaju nikakav uticaj na dalje rešavanje diferencijalne jednačine. Kako nema opterećenja u pravcu y-ose, dobija se:

$$N_y = 0 \tag{6}$$

kao i u pravcu xy odnosno yx sledi da je:

$$N_{xy} = N_{yx} = 0 \tag{7}$$

U pravcu x-ose postoji opterećenje i ono je u obliku:

$$N_x = -P \tag{8}$$

Zadatkom je zadato da se krutost podloge menja prema relaciji:

$$c = k = k_0 + k_1 x \tag{9}$$

gde su k_0, k_1konstante.

Znajući sve navedene jednačine, jednačina (5) postaje:

$$D\nabla^4 w = -kw - P\frac{\partial^2 w}{\partial x^2} - \rho h\ddot{w} \tag{10}$$

Jednačina (10) je diferencijalna jednačina kojom se rešavaju zadati problemi sa slike 2. Za napisanu jednačinu primeniće se Galerkinova metoda pomoću koje će se dalje rešavati diferencijalna jednačina.

Primenom Galerkinove metode pretpostavlja se rešenje u obliku:

$$w = F(x,y)\sin\omega t \tag{11}$$

Zatim se za jednačinu (11) odredi drugi izvod po vremenu pa postaje:

$$\ddot{w} = \frac{\partial^2 w}{\partial t^2} = \omega^2 F(x,y)\sin\omega t \tag{12}$$

Raspisivanjem bilaplasijana u jednačini (10) dobijamo istu jednačinu zapisanu u obliku:

$$D\left(\frac{\partial^4 w}{\partial x^4} + 2\frac{\partial^4 w}{\partial x^2\partial y^2} + \frac{\partial^4 w}{\partial y^4}\right) = -kw - P\frac{\partial^2 w}{\partial x^2} - \rho h\ddot{w} \tag{13}$$

Uvrštavanjem jedančina (8),(9) i (12) u jednačinu (13), jednačina (13) postaje:

$$D\left(\frac{\partial^4 F}{\partial x^4} + 2\frac{\partial^4 F}{\partial x^2\partial y^2} + \frac{\partial^4 F}{\partial y^4}\right)\sin\omega t = -(k_0 + k_1 x)\,F\sin\omega t - P\frac{\partial^2 F}{\partial x^2} + \rho h\omega^2 F(x,y)\sin\omega t \tag{14}$$

gde se deljenjem sa $\sin\omega t$ dobija:

$$D\left(\frac{\partial^4 F}{\partial x^4} + 2\frac{\partial^4 F}{\partial x^2\partial y^2} + \frac{\partial^4 F}{\partial y^4}\right) = -(k_0 + k_1 x)F - P\frac{\partial^2 F}{\partial x^2} + \rho h\omega^2 F \tag{15}$$

Jednačina (15) je konačna jednačina aksijalno opterećene i ukleštene pravougaone ploče na nehomogenoj elastičnoj podlozi. Pored dobijene diferencijalne jednačine potrebno je ispisati granične uslove pomoću kojih će se rešiti jednačina. Sa slike 2 će se ispisati granični uslovi za posmatranu ploču. Sve četiri stranice su ukleštene i na osnovu toga imamo da su granični uslovi:

za stranicu AB:

$$w(x,0) = 0,\quad \frac{\partial w}{\partial y}(x,0) = 0 \tag{16}$$

za stranicu AC:

$$w(0,y) = 0,\quad \frac{\partial w}{\partial x}(0,y) = 0 \tag{17}$$

za stranicu BD:

$$w(a,y) = 0,\quad \frac{\partial w}{\partial x}(a,y) = 0 \tag{18}$$

za stranicu CD:

$$w(x,b) = 0,\quad \frac{\partial w}{\partial y}(x,b) = 0 \tag{19}$$

čime je predstavljeno da su za sve četiri stranice pomeranje i ugao nagiba tangente jednaki nula.

Diferencijalnu jednačinu za aksijalno opterećenu i ukleštenu pravougaonu ploču na nehomogenoj elastičnoj podlozi (15) kao i granične uslove (16-19) potrebno je prebaciti u bezdimenzijski oblik radi lakšeg rešavanja problema.

3. BEZDIMENZIJSKI OBLIK POSTAVLJENOG PROBLEMA PLOČE

Prebacivanjem konačne diferencijalne jednačine kao i graničnih uslova u bezdimenzijski oblik pojednostavljuje se dalje rešavanje posmatranog problema. Jednačinama u bazdimenzijskom obliku smanjuje se ukupan broj parametara za koje je potrebno pronaći rešenje.

Da bi se diferencijalna jednačina (15) napisala u bezdimenzijskom obliku, neophodno je za početak uvesti bezdimenzijske veličine u obliku:

$$\xi = \frac{x}{a};\eta = \frac{y}{b};f = \frac{F}{h};\beta = \frac{a}{b};\lambda = \frac{Pa^2}{D};\Omega^2 = \frac{\omega^2\rho h a^4}{D};B_1 = \frac{k_0 a^4}{D};B_2 = \frac{k_1 a^5}{D};X = \frac{w}{a};\ Y = \frac{w}{b}; \tag{20}$$

Uvedene bezdimenzijske veličine, jednačine $(20)_{1,2,3}$ uvrštavaju se u jednačinu (15) i dobija se:

$$D\left(\frac{\partial^4 fh}{\partial\xi^4 a^4} + 2\frac{\partial^4 fh}{\partial\xi^2 a^2\partial\eta^2 b^2} + \frac{\partial^4 fh}{\partial\eta^4 b^4}\right) = -(k_0 + k_1\xi a)fh - P\frac{\partial^2 fh}{\partial\xi^2 a^2} + \rho h\omega^2 fh \tag{21}$$

Zatim se množenjem prethodne jednačine sa $\frac{a^4}{h}$ i korišćenjem jednačine $(20)_4$dobija:

$$D\left(\frac{\partial^4 f}{\partial\xi^4} + 2\frac{\partial^4 f}{\partial\xi^2\partial\eta^2}\beta^2 + \beta^4\frac{\partial^4 f}{\partial\eta^4}\right) = -(k_0 + k_1\xi a)fa^4 - P\frac{\partial^2 f}{\partial\xi^2}a^2 + \rho h\omega^2 fh \tag{22}$$

A na kraju se primenom jednačina $(20)_{5,6,7,8}$ na jednačinu (22) dobija konačna diferencijalna jednačina u bezdimenzijskom obliku:

$$\frac{\partial^4 f}{\partial\xi^4} + 2\frac{\partial^4 f}{\partial\xi^2\partial\eta^2}\beta^2 + \beta^4\frac{\partial^4 f}{\partial\eta^4} + \lambda\frac{\partial^2 f}{\partial\xi^2} - \Omega^2 f + B_1 f + B_2 f\xi = 0 \tag{23}$$

Sada je potrebno uz primenu jednačina (20) zapisati granične uslove u bezdimenzijskom obliku:
stranica AB:

$$w(x,0) = 0 \;\Rightarrow\; Y(0) = 0$$
$$\frac{\partial w}{\partial y}(x,0) = 0 \;\Rightarrow\; Y`(0) = 0 \qquad (24)$$

stranica AC:

$$w(0,y) = 0 \;\Rightarrow\; X(0) = 0$$
$$\frac{\partial w}{\partial x}(0,y) = 0 \;\Rightarrow\; X`(0) = 0 \qquad (25)$$

stranica BD:

$$w(a,y) = 0 \;\Rightarrow\; X(1) = 0$$
$$\frac{\partial w}{\partial x}(a,y) = 0 \;\Rightarrow\; X`(1) = 0 \qquad (26)$$

stranica CD:

$$w(x,b) = 0 \;\Rightarrow\; Y(1) = 0$$
$$\frac{\partial w}{\partial y}(x,b) = 0 \;\Rightarrow\; Y`(1) = 0 \qquad (27)$$

Bezdimenzijska diferencijalna jednačina (23) kao i granični uslovi u bezdimenzijskom obliku (24-27), rešavaju se uz korišćenje numeričkih metoda uz pomoć softvera Wolfram Mathematica. Navedenim jednačinama potrebno je dodati i odabrane poznate funkcije kako bi se došlo do rešenja traženih parametara.

4. ANALIZA REZULTATA – ZAVISNOST FREKVENCIJE OSCILOVANJA OD PARAMETARA KOJI DELUJU NA PLOČU

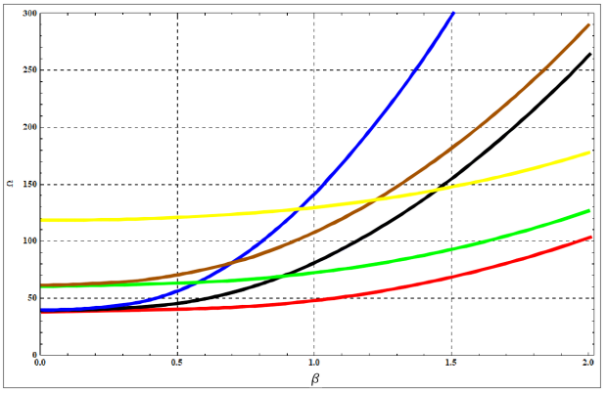
U ovom delu predstaviće se zavisnost kružne frekvencije od ostalih parametara koji deluju na posmatranu ploču (slika 2). Na vrednost bezdimenzijske kružne frekvencije Ω utiče: bezdimenzijski odnos stranica pločeβ, bezdimenzijska kritična sila u aksijalnom pravcu pločeλ, bezdimenzijske sile u oprugama B_1, B_2.
U tabeli 1 predstaviće se dobijene vrednosti bezdimenzijske kružne frekvencije oscilovanja ploče po modovima za različit odnos stranica poloče. Uzete vrednosti za bezdimenzijsku kritičnu silu u aksijalnom pravcu kao i bezdimenzijske sile u oprugama iznose $\lambda = 40, B_1 = B_2 = 1000$. Ovakav odabir vrednosti parametara govori da sve veličine koje deluju na posmatranu ploču imaju uticaj na istu.

Tabela 1. *Bezdimenzijska kružna frekvencija po modovima za različit odnos stranica ploče kada su ostali parametri* $\lambda = 40, B_1 = B_2 = 1000$.

	Ω_{m1}	Ω_{m2}
$\beta = 0.5$	39.9216	45.0004
$\beta = 1$	47.9439	72.5443
$\beta = 1.5$	68.6453	92.8958
$\beta = 2$	103.447	126.453

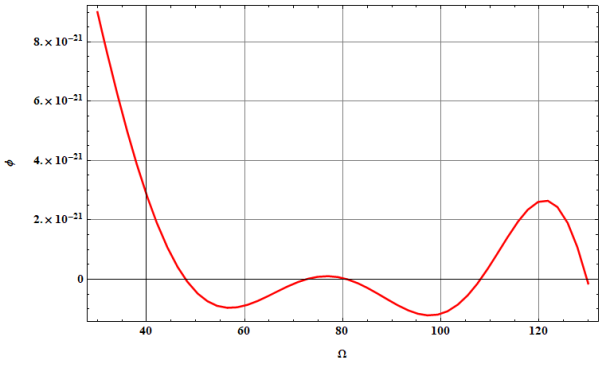
Iz tabele 1 vidi se kako se ponaša bezdimenzijska kružna frekvencija za prva dva moda u zavisnosti da li je posmatrana ploča pravougaonog (odnos stranica dat: β=0.5, β=1.5, β=2) ili kvadratnog oblika (odnos stranica: β=1), a da na nju ne deluje niti sila u aksijalnom pravcu niti sile u oprugama.

Može se videti da i u prvom i u drugom modu kružna frekvencija raste prilikom povećanja odnosa između dve stranice ploče. Kako se jedna strana ploče povećava u odnosu na drugu tako dolazi do povećanja bezdimenzijske kružne frekvencije. Navedeni rezultati iz tabele 1 predstavljeni su na slici 3.



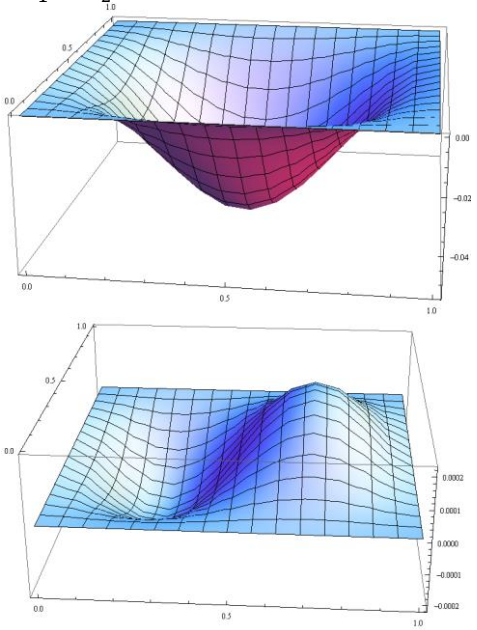
Slika 3. *Grafički prikaz kružne frekvencije i odnosa stranica ploče*

U nastavku slede dobijeni rezultati za ploču kvadratnog oblika β=1.



Slika 4. *Dijagram bezdimenzijske kružne frekvencije*Ω

Na slici 4 prikazan je dijagram bezdimenzijske kružne frekvencije za slučaj kada vrednost bezdimenzijske kritične sile u aksijalnom pravcu ploče iznosi$\lambda = 40$, dok je vrednost bezdimenzijske sile u oprugama koje deluju na ploču $B_1 = B_2 = 1000$



Slika 5. *Oblik oscilovanja ploče – prva dva moda*

Sa slike 5 vidi se kako posmatrana ploča osciluje, prva dva moda, kada na nju deluju sve posmatrane sile sa svojim vrednostima $\lambda = 40, B_1 = B_2 = 100$, za vrednost kružne frekvencije $\Omega_{m1} = 47.9439$ kod prvog moda i vrednost kružne frekvencije $\Omega_{m2} = 72.5443$ kod drugog moda.

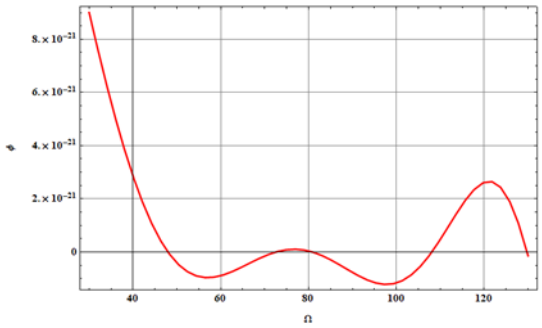
5. ANALIZA REZULTATA - ZAVISNOST KRITIČNE SILE OD PARAMETARA KOJI DELUJU NA PLOČU

U ovom delu predstaviće se zavisnost kritične sile koja deluje u aksijalnom pravcu od ostalih parametara koji deluju na posmatranu ploču (slika 2). Na vrednost bezdimenzijske kritične sileλutiče: bezdimenzijski odnos stranica ploče β, bezdimenzijske sile u oprugama B_1, B_2. Bezdimenzijska kružna frekvencija Ω u ovom slučaju je jednaka nuli.
U tabeli 2 predstaviće se dobijene vrednosti bezdimenzijske kritične sile ploče za prva dva moda kod primera pravougaone i kvadratne ploče. Uzete vrednosti za bezdimenzijske sile u oprugama iznose $B_1 = B_2 = 100$.

Tabela 2. *Bezdimenzijska kritična sila po modovima za različit odnos stranica ploče kada su ostali parametri* $B_1 = B_2 = 100$.

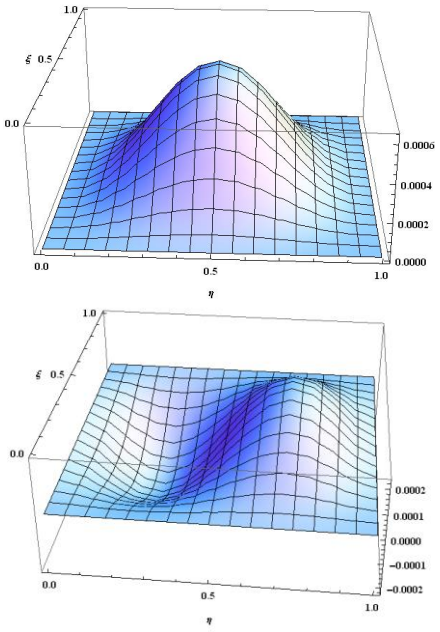
	λ_{m1}	λ_{m2}
$\beta = 0.5$	58.9968	96.5497
$\beta = 1$	109.974	128.851

U nastavku slede dobijeni rezultati za ploču kvadratnog oblika β=1.



Slika 6. *Dijagram bezdimenzijske kritične sile* λ

Na slici 6 prikazan je dijagram bezdimenzijske kritične sile za slučaj kada je vrednost bezdimenzijske sile u oprugama koje deluju na ploču $B_1 = B_2 = 100$.



Slika 7. *Oblik oscilovanja ploče – prva dva moda*

Tražene vrednosti prva dva moda bezdimenzijske kritične sile su one tačke na dijagramu gde crvena linija preseca nultu liniju za razne vrednosti parametara koji utiču na pločuB_1, B_2, β. Dobijene vrednosti su vrednosti pri kojima dolazi do gubitka stabilnosti ploče, tj. ploča više nije stabilna. Kritična sila u oba moda raste prilikom povećanja odnosa između dve stranice ploče. Što je veća vrednost bezdimenzijske sile u oprugama to je vrednost bezdimenzijske kritične sile veća.

Sa slike 7 vidi se kako se posmatrana ploča savija, prva dva moda, kada na nju deluju sve posmatrane sile sa svojim vrednostima $B_1 = B_2 = 100$, za vrednost kritične sile $\lambda_{m1} = 109.974$kod prvog moda i vrednost kritične sile $\lambda_{m2} = 128.851$kod drugog moda.

6. ZAKLJUČAK

Zadatak rada je bio da se odredi vrednost bezdimenzijske frekvencije oscilovanja ploče i vrednost bezdimenzijske kritične sile pri kojoj dolazi do gubitka stabilnosti aksijalno opterećene i ukleštene pravougaone ploče na nehomogenoj elastičnoj podlozi.

Prvo je prikazana diferencijalna jednačina savijanja ploče koja je poprečno opterećena (1).Zatim se uz primenu Dalamberovog principa došlo do jednačine u kojoj se opisuje oscilovanje ploče (4). Potom se za postavljeni problem sa slike 1 uz primenu Galerkinove metode izvela diferencijalna jednačina (15) koja opisuje isti, kao i granični uslovi koji odgovaraju postavljenom problemu (16-19). Pomenute jednačine su prebačene u bezdimenzijski oblik (23, 24-27) i kao takve rešavane primenom numeričkih metoda pomoću softvera Wolfram Mathematica. Analizirani su dobijeni rezultati pri kojima dolazi do gubitka stabilnosti posmatrane ploče sa fokusom na bezdimenzijsku frekvenciju oscilovanja i bezdimenzijsku kritičnu silu. Rezultati su dati u vidu slika i tabela.

7. LITERATURA

[1] Arthur W. Leissa, Vibration of plates, (NASA Report SP-160).
[2] Bhalchandra Y. B., Vibration of Rectangular Plates, Houston, Texas, July [1966].
[3] Hajdin N., Teorija Površinskih Nosača, (Naučna Knjiga, Građevinski Fakultet, Beograd, 1989).
[4] S. Timošenko, S. Vojnovski-Kriger, Teorija ploča i ljuski, Građevinska knjiga - Beograd, [1962].
[5] SzilardR.,Theories and Applications of Plate Analysis, (John Wiley & Sons Inc. New Jersey) [2004].

Kratka biografija:

Radivoj Smiljanić; rođen je u Sremskoj Mitrovici 1991. god. Master rad na Fakultetu tehničkih nauka iz oblasti Mašinstvo – Tehnička mehanika i dizajn u tehnici odbranio je 2020.god. kontakt: radivojsmiljanic@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 007.52
DOI: https://doi.org/10.24867/12AM06Varga

ANALIZA MOGUĆNOSTI OPTIMIZACIJE PUTANJE ALATA ZA POZICIONU PETOOSNU OBRADU NA CNC GLODALICAMA

ANALYSIS OF TOOL PATH OPTIMIZATION POSSIBILITIES FOR POSITIONAL FIVE-AXIS MACHINING ON CNC MILLING MACHINES

Čongor Varga, Slobodan Tabaković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – MAŠINSTVO

Kratak sadržaj – *Rad predstavlja analizu mogućnosti optimizacije putanje alata na savremenim petoosnim obradnim centrima za obradu glodanjem u SolidCAM softveru sa podrškom iMachining modula u cilju smanjenja obrade, optimalne upotrebe alata i materijala, a samim tim i puno novca.*

Ključne reči: *Numerički upravljane mašine alatke, poziciona petoosna obrada, CAM sistemi, SolidCAM, iMachining*

Abstract – *The work presents an analysis of the tool path optimization possibilities on modern five - axis CNC milling machines in SolidCAM software with the support of the iMachining module in order to save time and tools, and thus a lot of money.*

Keywords: *CNC machine, positional five-axis machining, CAM systems, SolidCAM, iMachining*

1. UVOD

Razvoj numerički upravljanih mašina alatki (NUMA) je započeo tokom drugog svetskog rata. Napredak na svim područjima vojnog i komercijalnog razvoja je bio toliko brz da se nivo automatizacije i tačnosti koji zahteva moderan industrijski svet nije mogao postići pomoću tada dostupnih konvencionalnih mašina alatki. Nakon nekoliko neuspešnih projekata američke vlade, kompanija Parsons na čelu sa Džonom Parsonom (John Parsons) i MIT (eng. Massachusetts Institute of Technology) je u periodu od 1949. do 1953. godine razvila prvu numerički upravljanu mašinu alatku (NUMA) [1]. Upravljačka jedinica ove mašine alatke omogućavala je simultano kretanje duž tri koordinatne ose, obezbeđujući zadovoljavajuću tačnost izrade radnih predmeta kompleksne geometrije sa uskim tolerancijama. Dalji razvoj na ovom polju je omogućio razvoj savremenih CNC mašina alatki kod kojih je za operacije glodanja omogućena petoosna obrada.

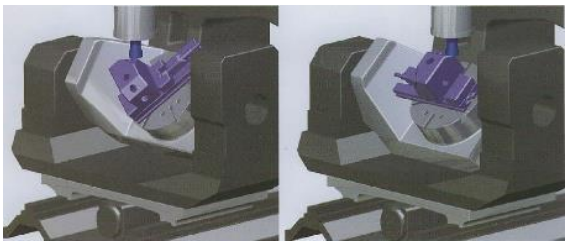
Radom je obuhvaćena analiza mogućnosti obrade glodanjem u savremenim SolidWorks i SolidCAM softverima. Zadatkom je obuhvaćen i postupak definisanja putanje alata i upravljačkih programa za NU obradne centre za petoosnu obradu glodanjem. Opis postupka definisanja putanje alata i upravljačkih programa je realizovan primenom programskog sistema SolidWorks i SolidCAM softvera, koji su u industrijskim uslovima najčešće korišćeni kompleksni CAD/CAE/CAM programski sistemi. Praktični deo rada obuhvata izradu jednog primera programiranja u softveru SolidCAM sa konvencionalnom metodom u odnosu na drugi primer koji je obrađen u SolidCAM pomoću iMachining modula za koje je potrebno opisati postupak definisanja putanja alata tokom obrade, i na kraju kritički sagledati opisanu problematiku i zaključiti šta su glavne prednosti i nedostaci automatizovanog programiranja NUMA sa pet osa konvencionalnom metodom u odnosu na iMachining obradu.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Slobodan Tabaković, red. prof.

2. OPERACIJE POZICIONE VIŠEOSNE OBRADE I POSTUPCI OPTIMIZOVANJA PUTANJE ALATA

Postupci obrade za pozicioniranjem radnog predmeta pomoću indeksnog rotacionog kretanja obezbeđuju zadovoljavajuću krutost i preciznost. Drugi uobičajen naziv koji se koristi za ovakvo pozicioniranje je 2+3 obrada. Kod ovog tipa obrade, rotacione/okretne ose koriste se samo za pozicioniranje, a obrada se odvija samo sa kretanjem tri linearne ose. Ovom metodom je moguće izraditi veliki broj različitih tipova radnih predmeta i to je najosnovniji višeosni koncept. Time se obezbeđuje laka tranzicija od pozicioniranja radnog predmeta do neposredne obrade. Na slici 1 prikazano je kako se jedan deo može rezati iz različitih uglova bez otpuštanja i prepozicioniranja [2].



Slika 1. *Slike koje pokazuju kako se jedan deo može rezati iz različitih uglova, bez uklanjanja iz stezanja [2]*

Ovakva obrada se koristi za komplikovanije radne predmeti. Ekonomski je isplativa zbog kratkog vremena pripreme mašine i upotrebe samo jednog pribora za stezanje radnog predmeta.

3. SAVREMENI CAM PROGRAMSKI SISTEMI

Princip automatizovanog programiranja NUMA se zasniva na korišćenju geometrijskog opisa pripremka i

izradka dobijenih u CAD sistemima. Tako definisana geometrija proizvoda se u nekom od CAM modula obrađuje u cilju dobijanja putanje alata (CL datoteku – Cutter location data). Izbor redosleda operacija i zahvata kao i tehnoloških parametara obrade određuje tehnolog - ”programer”.

Nakon dobijenog NU programa (opšti oblik) potrebno je izvršiti postprocesiranje podataka za određenu upravljačku jedinicu alatne mašine. Nakon toga upravljački program je spreman za upotrebu na mašini alatki.

Osnovne funkcije CAM sistema vezane su za planiranje proizvodno-tehnoloških procesa. Među njima su [1]:

- generisanje pripremka,
- generisanje i optimizacija putanja alata,
- kreiranje i korišćenje baza podataka i kataloga režima i alata,
- proračun vremena izrade,
- generisanje NC programa,
- simulacija i vizuelizacija procesa izrade,
- generisanje proizvodne dokumentacije,
- brza izrada prototipova (rapid prototyping)

Neki od najpoznatijih CAM sistema su: Catia, Cimatron, Siemens NX, Creo, SolidCAM, Mastercam, HyperMill, WorkNC [3].

3.1. SolidCAM

Danas na tržištu postoji veliki broj programskih rešenja koja omogućuju primenu i razvoj visokobrzinskih i visokoproduktivnih obrada rezanjem. Jedno od programskih rešenja koje prati trendove savremene obrade rezanjem je SolidCAM koji predstavlja zasebno rešenje u području CAM tehnologije.

SolidCAM sačinjava familija podmodula koji su napravljeni u cilju zadovoljenja potreba kako malih tako i velikih proizvodnih sistema. SolidCAM sačinjavaju 4 modula kojima se realizuju operacije obrade.To su:

- Glodanje (Milling)
- Struganje (Turning)
- Hibridne obrade (Mill-Turn)
- Elektroerozivna obrada žicom (Wire Cut).

U okviru ovih modula postoji više tehnoloških zahvata obrade, od kojih su kod glodanja najznačajnije:

- Čeono glodanje (Face)
- Profilno glodanje (Profil)
- Glodanje džepova (Pocket)
- Bušenje (Drilling)
- Glodanje navoja (Thread)
- Glodanje žljebova (Slot)
- Glodanje T-žljebova (T-Slot)...

Kod struganja su najznačajniji zahvati:

- Konturno struganje (Turning)
- Čeono struganje (Face Turning)
- Struganje navoja (Threading)
- Struganje žljebova (Grooving)...

3.1.1. Modul iMachining

iMachining je modul SolidCAM softverskog sistema koji omogućuje konstantno opterećenje reznog alata.

Smanjenje glavnog vremena obrade je moguće postići ako se poveća efikasnost skidanja materijala a to se može postići:

- Smanjenjem vremena praznih hodova, odnosno smanjenjem vremena repozicioniranja alata i vraćanja u bezbednosnu ravan, cilj je da alat bude uvek u zahvatu čime se obezbeđuje konstantnost skidanja materijala, a samim tim i manje glavno vreme obrade,
- Ugrađenom optimizacionom logikom koja omogućava da se alat, ako to nije potrebno, ne vraća u bezbednosnu ravan, već da se repozicioniranje alata vrši na novoj bezbednosnoj ravni koja je generisana u odnosu na već skinuti materijal,
- Povećanjem zapremine skidanja strugotine, koja se povećava povećanjem aksijalne dubine rezanja, širine rezanja i dužine strugotine.

Povećanje životnog veka alata se postiže konstantnim, umerenim mehaničkim i termičkim opterećenjima, odnosno:

- Eliminisanjem prinudnih vibracija što ima značajan uticaj na životni vek alata. Usled naglih promena sila rezanja dolazi do velikih oscilacija alata što nepovoljno utiče na postojanost alata,
- Treba voditi računa o površini režućeg sloja odnosno treba voditi računa da zapremina materijala koja se skida bude približno, koliko to uslovi rezanja dozvoljavaju, konstantna.
- Jedan od uzroka nastajanja vibracija su i termička opterećenja. Primenom režima visokoproduktivnih obrada i putanja koje pruža iMachining, eksperimenti su potvrdili i činjenicu da se skoro 95% ukupne temperature rezanja odvodi strugotinom, a samo mali procenat odlazi u alat.
- Vibracije alata se takođe uklanjaju upotrebom mašina alatki visoke krutosti.

4. PRIMER PROGRAMIRANJE PETOOSNE NUMA PRIMENOM PROGRAMSKOG SISTEMA SOLIDCAM UZ PODRŠKU MODULA IMACHINING

Cilj ovog rada je da se predhodno pomenute teorije iskoriste za izradu konkretnog radnog predmeta primenom programskog sistema SolidCAM. Pored toga, cilj predstavlja i poređenje postupka programiranja NUMA u programskom sistemu SolidCAM konvencionalnom metodom sa postupkom koji obuhvata priemnu modula iMachining. Verifikacija putanje alata je urađena u simulaciji SolidCAM-a, a G-kod je generisan sa postprocesorom za NUMA Hermle C400 (Slika 2.). Ceo obradni proces je urađen u Adi u preduzeću Termometal.

Obradni centar Hermle C400 ima sledeće karakteristike:

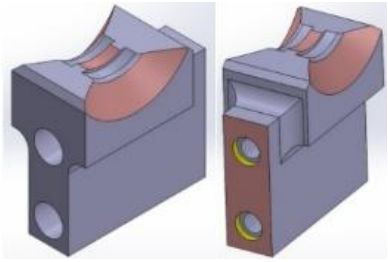
- pet upravljačkih osa sa hodovima 850mm za X osu, 700mm za Y osu, 500mm za Z osu i opseg

okretanja upravljačke ose A +91°/-139°, ose C 360°,
- maksimalnu brzinu vretena 15000 obr./min.,
- maximalni brzi linearni hod 30000 mm/min,
- upravljačku jedinicu Heidenhain TNC 640,
- maximalnu snagu vretena 20 kW,
- rotacioni sto sa steznom površinom Ø440mm i maksimalnim opterećenjem 450kg,
- maksimalni prečnik obratka Ø650mm,
- maksimalnu visinu radnog predmeta 500mm,
- magacin alata od 38 komada [4].

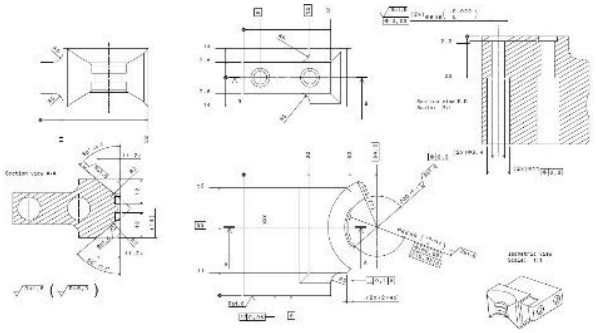
Slika 2. *Petoosna CNC mašina alatka Hermle C400 [5]*

Za radni predmet je izabran deo steznog pribora iz proizvodnje. Za definisanje 3D modela korišćen je programski sistem SolidWorks. 3D model radnog predmeta se nalazi na slici 3.

Za konkretni radni predmet, gabaritnih dimenzija 28x52x52mm na osnovu 2D radioničkog crteža (Slika 4.), izabran je pločasti pripremak od čelika Č.4830, 50CrV4 dimenzija 30x58x58mm.



Slika 3. *3D model obratka*



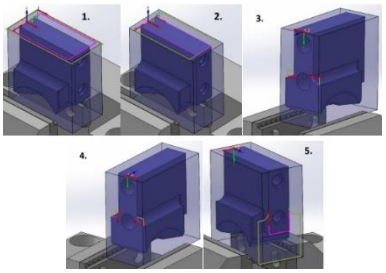
Slika 4. *2D radionički crtež radnog komada*

Za izradu radnog komada korišćeno je ukupno 12 alata.

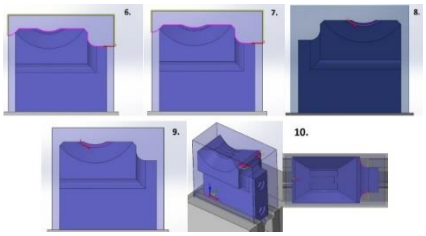
5. ANALIZA REZULTATA PRIMENE OPTIMIZACIJE

Osnovni cilj ovog rada je upoređivanje programiranja u softverskom sistemu SolidCAM sa konvencionalnom metodom u odnosu na SolidCAM pomoću primene iMachining modula.

Glavni cilj rada nije da se predstavi programiranje NUMA pomoću softverskog sistema, pa nije potrebno prikazati sve zahvate obrade. Zbog toga je potrebno izabrati nekoliko kontura. Za upoređivanje su izabrane konture koje su navedene na slikama 5 i 6.



Slika 5. *Praćene konture u prvom stezanju*



Slika 6. *Praćene konture u drugom stezanju*

Tehnološkim postupkom obrade predviđeno je da operacije obrade glodanjem obuhvati sledeće zahvate:

1. Obrada spoljašnje strane 1
2. Obrada spoljašnje strane 2
3. R5mm radijusa 1
4. R5mm radijusa 2
5. Obrada konture desne strane
6. Obrada gornjeg dela
7. Skidanje ostatka materijala na gornjem delu
8. R15mm radijusa 1
9. R15 mm radijusa 2
10. R5mm radijusa 3-4

5.1. Ukupno vreme obrade za petoosno glodanje pre termičkog obrade

Dobijeni rezultati ukupnog vremena obrade glodanja na CNC petoosnoj glodalici kod konvencionalnog programiranja za dve varijante i kod iMachining programiranja prikazani su u sledećem dijagramu 1:

Dijagram 1. *Ukupno vreme obrade za petoosno glodanje pre termičkog obrade*

Sa dijagrama se jasno vidi da iMachining programiranje daje kraće vreme obrade u odnosu na konvencionalno programiranje.

Na sledećim slikama prikazan je gotov komad (Slika 7.).

Slika 7. *Gotov komad*

6. ZAKLJUČAK

Cilj ovog rada jeste da utvrdi realne prednosti primene na najnovije generacije integrisanih programskih rešenja koji pružaju efikasno i brzo projektovanje tehnoloških operacija obrade uz generisanje savremenih putanja alata i automatizovano proračunavanje tehnoloških parametara.

Tema rada obuhvata analizu mogućnosti petoosne obrade glodanjem u savremenim SolidWorks i SolidCAM softverima. Praktični deo rada obuhvata izradu jednog primera programiranja u softveru SolidCAM sa konvencionalnom metodom u odnosu na drugi primer koji je obrađen u SolidCAM pomoću iMachining modula za koje je potrebno opisati postupak definisanja putanja alata tokom obrade, i na kraju kritički sagledati opisanu problematiku i zaključiti šta su glavne prednosti i nedostaci automatizovanog programiranja NUMA sa pet osa konvencionalnom metodom u odnosu na iMachining obradu.

Kod programskog sistema SolidCAM za programiranje NUMA, nova generacija programskih rešenja se zove iMachining za generisanje složenih putanja alata i automatizovano proračunavanje tehnoloških parametara obrade, smanjuje vreme projektovanja proizvodnje, a pored toga što je mnogo važnije, smanjuje glavno vreme obrade u odnosu na konvencionalno programiranje čak i do 70%. Takve strategije sigurno pomeraju svest da su konvencionalne putanje alata prošlost.

U ovom radu se ukazuje na mogućnosti ovog programskog sistema, i uz pomoć istog izradi konkretni radni predmet. Snima se vreme obrade pojedinih zahvata obrade i upoređuje se sa konvencionalnim programiranjem NUMA sa pet osa. Upoređivanje se odnosi isključivo na glavno vreme obrade, ali optimizacija ovog programskog rešenja se odnosi i na druge karakteristike obrade kao što su npr. postojanost, habanje alata, sile rezanja, mehaničko, termičko opterećenje alata, povećava se tačnost izrade jer se uklanjaju vibracije, povećava se proizvodnost i još neki parametri.

Od 3D modela radnog predmeta do gotovog radnog predmeta potrebno je proći određene faze projektovanja koje su prikazane u prethodnim poglavljima. Na osnovu dobijenih rezultata mogu se doneti sledeći zaključci:

- SolidCAM predstavlja programski sistem koji prati trendove tržišta. To dokazuje i činjenica o njegovoj obimnosti kojom pokriva veliki deo obrada skidanjem materijala.
- Pored trendova tržista, u koraku je i sa trendovima programskih sistema. To dokazuje i činjenica da je on integrisano programsko rešenje koje je moguće „uneti“ u razna CAD rešenja. Dokaz tome su dva konkurentska CAD programska rešenja SolidWorks i AutoDesk Inventor, pri čemu je SolidCAM integrisan u oba.
- Nove strategije za putanje alata i automatizovano proračunavanje tehnoloških parametara su takođe integrisani, ovaj slučaj ne postoji ni kod jednog programskog sistema za programiranje NUMA.
- U prethodnom poglavlju dobijeni rezultati dokazuju da modul iMachining u većini slučajeva skraćuje glavno vreme obrade.

7. LITERATURA

[1] Todorović, M.: NUMERIČKI UPRAVLJANE ALATNEMAŠINE - CAM SISTEMI. https://www.scribd.com/doc/57986455/PROGRAMIRANJE-NUMERI%C4%8CKI-UPRAVLJANIH-GLODALICA-POMO%C4%86U-CAM-SISTEMA (pristupljeno u maju 2020.)

[2] Apro, K.: Secrets of 5-Axis Machining, Industrial Press, Inc., New York, 2008.

[3] Zeljković, M., Tabaković, S., Živković, A.: CAD/CAE/CAM i CIM sistemi (Autorizovani rukopis predavanja), Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad, 2015/2016.

[4] Svojstva Hermle C400 mašina alatka.https://www.hermle.de/en/machining_centres/models/c_400_machining_centre (pristupljeno u maju 2020.)

[5] Petoosna CNC mašina alatka - Hermle C400.https://www.google.com/search?q=hermle%20c400&tbm=isch&tbs=rimg:CY6pBpn7MTZ_1YbbHjLidqXD4&hl=hu&sa=X&ved=0CBsQuIIBahcKEwigj5aUjaDsAhUAAAAAHQAAAAAQaQ&biw=1583&bih=789#imgrc=zpOmhG1B0hXWQM (pristupljeno u maju 2020.)

Kratka biografija:

Čongor Varga rođen je u Senti 1995. god. Master rad na Fakultetu tehničkih nauka iz oblasti Proizvodno mašinstvo – Savremeni prilazi u projektovanju proizvoda odbranio je 2020.god.
kontakt: vcsongor1995@gmail.com

Slobodan Tabaković rođen je 1974. god. Doktorirao je na Fakultetu tehničkih nauka 2008. god., a od 2018 je zvanju redovni profesor. Oblast interesovanja su Mašine alatke, Fleksibilni tehnološki sistemi i automatizacija postupaka projektovanja.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: <621.9>
DOI: https://doi.org/10.24867/12AM07Mandic

ANALIZA POSTUPKA PROGRAMIRANJA OBRADNIH CENTARA ZA STRUGANJE-GLODANJE

ANALYSIS OF THE PROGRAMMING PROCEDURE OF TURN-MILL MACHINING CENTERS

Nebojša Mandić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – MAŠINSTVO

Kratak sadržaj – *Savremeni trendovi gradnje mašina alatki ogledaju se u razvoju višefunkcionalnih mašina alatki, kao što su obradni centri za struganje-glodanje, koji imaju mogućnost izvođenja više pomoćnih kretanja alata u odnosu na obradak. U radu su opisani karakteristični postupci obrade na obradnim centrima za struganje-glodanje kroz tri grupe zahvata. Verifikacija postupaka programiranja kao i analiza karakteristika tehnoloških zahvata su izvršeni na obradnom centru za struganje-glodanje sa četiri numerički upravljane ose.*

Ključne reči: *Mašine alatke, Obradni centri za struganje-glodanje, Zahvati obrade, CAM programski sistemi, SolidCAM*

Abstract – *Modern trends in the construction of machine tools are reflected in the development of multifunctional machine tools, such as turn-mill machining centers, which have the ability to perform more auxiliary tool movements in relation to the workpiece. The paper describes the characteristic machining procedures at turn-mill machining centers through three groups of operations. Verification of programming procedures as well as analysis of the characteristics of technological operations were performed on a turn-mill machining center with four numerically controlled axes*

Keywords: *Machine tools, Turn-Mill machining centers, Processing operations, CAM software systems, SolidCAM*

1. UVOD

Čovek je oduvek težio obavljanju delatnosti sa što manje napora. Rezultat tih težnji jeste mehanizacija i automatizacija procesa proizvodnje, odnosno mašina i sistema. Osnovni ciljevi mehanizacije i automatizacije su smanjenje psihičkog i fizičkog naprezanja, posluživanje mašina sa veće udaljenosti, povećanje proizvodnosti i kvaliteta proizvoda [1].

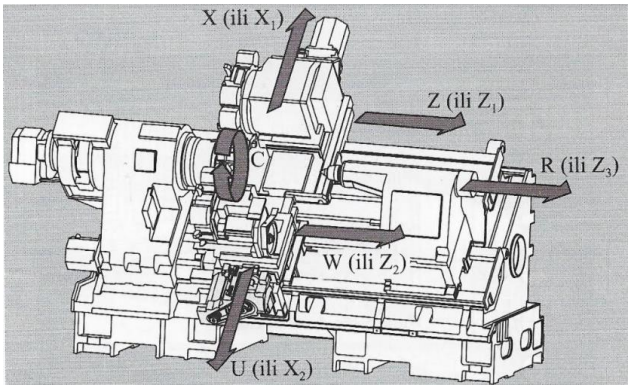
Tehnologija numeričkog upravljanja (engl. *Numerical Control*) pojavila se sredinom dvadesetog veka sa ciljem automatizacije serijske proizvodnje proizvoda složene geometrije i unapređenja tačnosti i kvaliteta [3].

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

Zadatak rada jeste da se kroz automatizovano programiranje četvoroosnog obradnog centra za struganje-glodanje izvrši analiza karakteristika tehnoloških zahvata i da se na taj način uoče razlike određenih zahvata obrade. Shodno tome, osmišljen je test radni predmet, čija se izrada izvodi u dve podoperacije, tj. iz dva stezanja. Kroz simulaciju i verifikaciju obradnog procesa prilikom programiranja primenom savremenog CAM programskog sistema, obezbeđena je provera svih putanja alata, čime je omogućeno otkrivanje neželjenih pojava i njihovo eliminisanje uz brzo vidljive efekte sprovedenih izmena.

2. OBRADNI CENTRI ZA STRUGANJE-GLODANJE

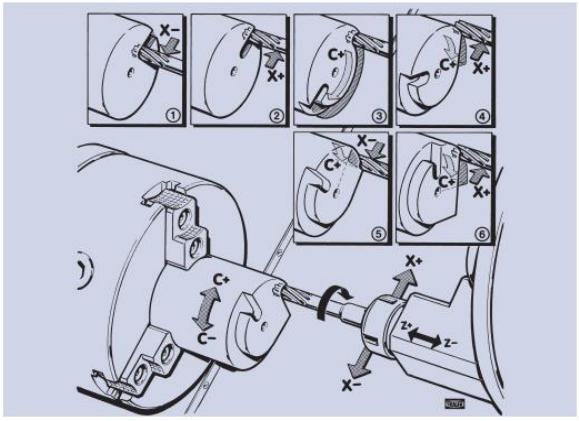
Prve mašine za obradu struganjem sa numeričkim upravljanjem su koncipirane i izvedene kao mašine sa dve numerički upravljane ose. Povećanjem broja upravljanih osa i opremanjem višepozicionih nosača alata gonjenim alatima, dolazi do nastanka obradnih centara za struganje-glodanje, koje su vrlo fleksibilne, tačne i ekonomične. Time su na jednu mašinu integrisane operacije struganja, bušenja, glodanja, izrade ozubljenja, kao i određene merno-kontrolne operacije. Na slici 2.1 je prikazan obradni centar za struganje-glodanje sa šest numerički upravljanih osa [2,4].



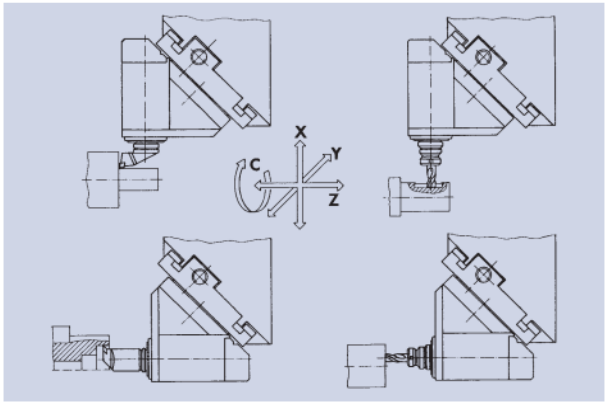
Slika 2.1 *Obradni centar za struganje-glodanje sa šest NU osa [4]*

Višefunkcionalne mašine alatke danas se uspešno koriste za obradu delova sa ravnim, cilindričnim i prostorno složenim površinama, obezbeđujući pri tome visok nivo produktivnosti i ekonomičnosti. Na slici 2.2 je prikazan strug sa tri numerički upravljane ose, na kom se izvodi obrada čela pomoću alata za glodanje, dok su na slici 2.3 prikazani od strugarskih zahvata, spoljašnje uzdužno i unutrašnje uzdužno struganje i od glodačkih zahvata,

radijalno i aksijalno glodanje na obradnom centru za struganje-glodanje sa četiri numerički upravljane ose.



Slika 2.2 *Obrada čela pomoću alata za glodanje [5]*



Slika 2.3 *Zahvati obrade na obradnom centru za struganje-glodanje sa četiri NU ose [5]*

3. PODELA ZAHVATA OBRADE

Karakteristični zahvati obrade na obradnim centrima za struganje-glodanje se mogu opisati kroz tri grupe zahvata obrade i tu spadaju: struganje, bušenje i glodanje.

Struganje je postupak obrade kod koga je glavno kretanje kružno i izvodi ga obradak, dok pomoćno kretanje izvodi alat. Grupa strugarskih zahvata obuhvata:

- poprečnu obradu,
- uzdužnu obradu,
- konturnu obradu,
- ukopavanje žljebova,
- narezivanje navoja i
- odsecanje.

Bušenje je postupak obrade sa glavnim kružnim i pomoćnim pravolinijskim kretanjem koje izvodi alat. Grupa zahvata bušenjem obuhvata:

- zabušivanje,
- bušenje,
- proširivanje,
- razvrtanje,
- duboko bušenje,
- upuštanje i
- urezivanje navoja.

Glodanje na obradnim centrima za struganje-glodanje je postupak obrade sa glavnim kružnim kretanjem koje izvodi alat i pomoćnim kretanjem koje mogu da izvode i obradak i alat u zavisnosti od koncepcije mašine. Kod obradnog centra za struganje-glodanje pomoćno kretanje u većini slučajeva izvodi alat po X, Y i Z osi, dok je C osa pomoćno kretanje koje izvodi obradak pomoću glavnog vretena mašine.

Kod obradnih centara za struganje-glodanje sa Y osom ograničavajući faktor je dužina hoda iste. Prilikom simultane obrade sa C osom, obrada se sprovodi konvertovanjem pravouglih koordinata u polarni koordinatni sistem. Grupa glodačkih zahvata obuhvata u radijalnom i aksijalnom pravcu obrade:

- izradu džepova,
- izradu žljebova,
- izradu ostrva,
- profilno glodanje,
- glodanje navoja i
- izradu ozubljenja.

4. PROGRAMIRANJE PRIMENOM CAM SISTEMA

Pod pojmom „programiranje numerički upravljanih mašina alatki" podrazumeva se niz aktivnosti koje je neophodno obaviti da bi se od konkretnog radnog zadatka došlo do upravljačkog programa za upravljanje numerički upravljanom mašinom alatkom [2].

Programiranje NUMA primenom CAM sistema predstavlja osnovu inženjerske pripreme savremene i automatizovane proizvodnje. Razvojem i primenom ove oblasti su u najvećoj meri otklonjeni nedostaci koji prate programiranje NUMA u radioničkim uslovima [3].

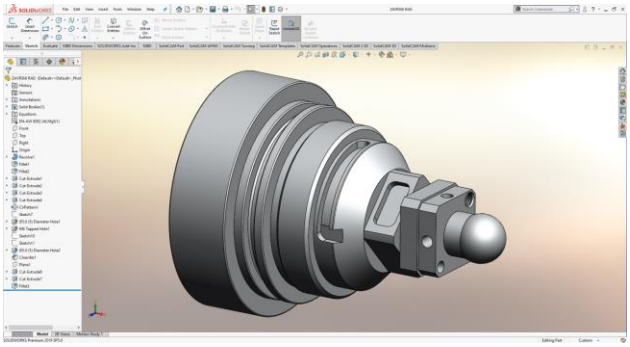
U poređenju sa ostalim metodama, programiranje NUMA primenom CAM sistema obezbeđuje niz prednosti, od kojih se kao najvažnije mogu izdvojiti [3]:

- intenzivnim korišćenjem podataka o proizvodu iz CAD sistema eliminiše se potreba za redefinisanjem geometrije proizvoda, što rezultuje smanjenjem grešaka i uštedama u vremenu,
- simulacijom i verifikacijom NC programa omogućeno je pravovremeno otkrivanje potencijalnih grešaka i njihova korekcija pre početka stvarne proizvodnje, čime se obezbeđuje podizanje nivoa sigurnosti i ekonomičnosti procesa proizvodnje,
- mogućnost automatizacije postupka programiranja korišćenjem programskih rutina, te ugradnje specifičnih znanja i ekspertnosti u CAM sistem, što ima za rezultat značajne uštede u vremenu i povećanje kvaliteta gotovih proizvoda,
- relativno jednostavno programiranje proizvoda sa složenom geometrijom,
- kreiranjem i korišćenjem baza podataka o proizvodnim resursima, režimima obrade i drugim elementima proizvodnog procesa, značajno se ubrzava postupak programiranja i
- mogućnost integracije sa drugim CAx sistemima u cilju provođenja koncepta konkurentnog inženjerstva.

Integracijom programskih sistema SolidWorks i SolidCAM dobija se specijalizovani programski paket koji ima primenu u području proizvodnog inženjerstva,

gde se teži integrisanju poslova konstrukcije (CAD) i pripremi proizvodnje kroz definisanje tehnološkog postupka i programiranje CNC mašina (CAM).
Pogodnost CAD/CAM sistema jeste činjenica da se svaka greška koja nastane u procesu definisanja programa za obradu može uočiti u toku simulacije i verifikacije.

SolidWorks je jedan od najpopularnijih programa za projektovanje proizvoda parametarskog solid modeliranja. Parametarsko modeliranje je oblik opisivanja trodimenzionih objekata u okviru koga se geometrijski model fizičkog tela može opisati skupom geometrijskih (paralelnost, normalnost i sl.) i dimenzionih (dužine, prečnici, uglovi i sl.) parametara.



Slika 4.1 *Korisnički interfejs programa SolidWorks*

SolidCAM je specijalizovani programski system koji se implementira u okruženje SolidWorks-a, tako da s njim čini funkcionalnu celinu. Na taj način je omogućeno istovremeno modeliranje mašinskih delova, izrada njihove tehničke dokumentacije, kao i izrada tehnologije za obradu i generisanje izvršnog programa za upravljanje CNC mašinama koje taj deo treba da izrade.

Tehnologija izrade proizvoda se zasniva na definisanom modelu koji je modeliran u CAD programu i koji se u radno okruženje SolidCAM-a uvozi direktno iz SolidWorks-a, gde se definišu svi parametri tehnologije za izradu proizvoda.

5. VERIFIKACIJA UPRAVLJAČKOG PROGRAMA

Analiza rezultata programiranja obradnih centara za struganje-glodanje primenom programskog sistema za automatizovano programiranje NUMA predstavlja drugi cilj istraživanja u radu. Realizacija ovog cilja je ostvarena kroz koncipiranje specijalizovanog test radnog predmeta, a zatim automatizovano programiranje SolidCAM programskim sistemom izradi NC program i proveru izradom na samoj mašini.

Polazni test komad je definisan na osnovu test radnog predmeta koji se koristi za verifikaciju zahvata obrade na strugovima sa tri numerički upravljane ose (X, Y i Z) razvijenog od strane kompanije Siemens za testiranje novih mašina alatki.

Za potrebe istraživanja test komad je proširen sa nekoliko geometrijskih oblika, prvenstveno radi poređenja zahvata glodanja koji se izvode sa C, odnosno Y osom. Konačna verzija test komada prikazana je na slici 5.1. Za pripremak je izabran cilindar dimenzija Ø100x130 [mm] od legure aluminijuma AlMgSi1.

Slika 5.1 *Test komad*

Nakon što je test komad definisan, bilo je potrebno definisati tehnologiju obrade. Detaljni opisi tehnološkog postupka, karakteristike usvojenih alata i vremena trajanja obrade su navedeni u master radu.

Verifikacija obrade je realizovana na obradnom centru za struganje-glodanje marke OKUMA, oznake LB3000 EX II, koji ima četiri ose, od kojih su tri translatorne (X, Y i Z) i jedna rotaciona (C) .

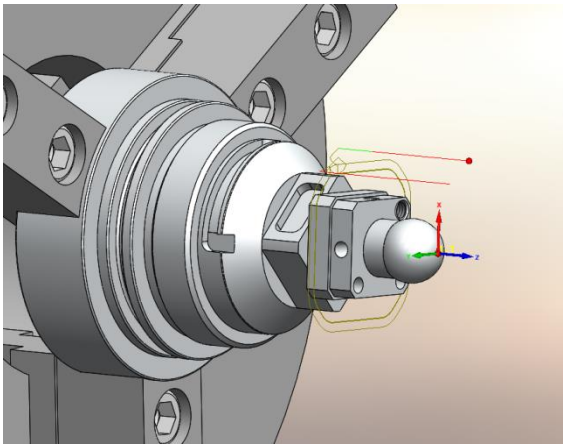
Slika 5.2 *OKUMA LB3000 EX II*

Zahvati aksijalnog glodanja se primenjuju pri obradi prvog i drugog kvadratnog ostrva. Obrada prvog kvadratnog ostrva izvodi se sa C osom čije vreme iznosi 34 [s], dok se obrada drugog kvadratnog ostrva izvodi sa Y osom čije vreme iznosi 1 [min] i 25 [s].

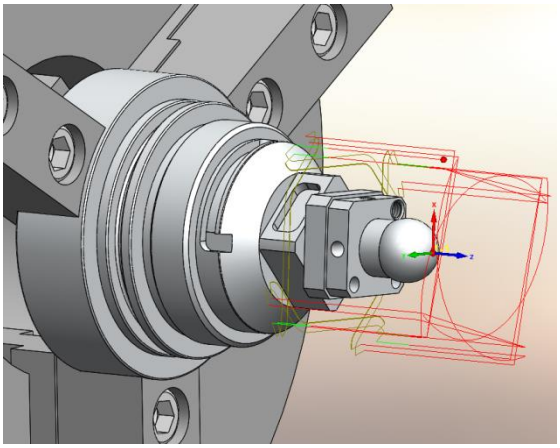
Pri zahvatu glodanja sa C osom kretanja se prevode na polarni koordinatni sistem, što znači da sve vreme obradak vrši pomoćno kretanje (rotaciju) pomoću C ose, a alat vrši pomoćno kretanje po X osi, dok Z osa služi samo za zauzimanje dubine prolaza.

Pri zahvatu glodanja sa Y osom, obradak miruje i obrada se izvodi klasičnim glodanjem. U ovom slučaju C osa služi samo da obradak dovede u radnu poziciju, nakon čega alat po Z osi zauzima dubinu prolaza i vrši obradu sa Y osom. Nakon završene prve strane, alat vrši odmicanje do sigurnosne ravni, dok se obradak rotira za 90º i

postupak se ponavlja još tri puta. Verifikacije putanja alata oba zahvata su prikazane na slikama 5.3 i 5.4.



Slika 5.3 *Verifikacija putanje alata kod zahvata glodanja prvog kvadratnog ostrva*



Slika 5.4 *Verifikacija putanje alata kod zahvata glodanja drugog kvadratnog ostrva*

6. ZAKLJUČAK

Osnovni cilj rada je da se kroz izradu test komada izvrši analiza tehnoloških zahvata obrade i da se izvedu zaključci o njihovoj primenljivosti proizvodnji na osnovu vremena obrade. Nakon prve pripremne podoperacije, realizovana je druga koja se realizuje prema predviđenom tehnološkom postupku. Analizom karakteristika i njihovim poređenjem kod zahvata glodanja koji se izvode sa C i Y osom, kao i vremena obrade, izvedeni su sledeći zaključci.

Pošto se izvedene obrade aksijalnog glodanja razlikuju u pomoćnim kretanjima, odnosno različitim strategijama putanja alata, korišćeni su i različiti koordinatni sistemi (pravougli i polarni). Kada se posmatraju oba ova slučaja uočava se da se vremena trajanja zahvata obrade bitno razlikuju. Glodanje prvog kvadratnog ostrva traje 34 [s], a glodanje drugog kvadratnog ostrva traje 1 [min] i 25 [s], iz čega vidimo da glodanje prvog kvadratnog ostrva traje 51 [s] kraće, odnosno dva i po puta manje. Na osnovu analize obrađenih površina se zaključuje da u kvalitetu obrade nisu uočene nikakve razlike i zaključak je da je u ovom slučaju efikasnije izvoditi obradu sa C osom kao pomoćno kretanje.

Ovakvo poređenje nije moguće uraditi pri radijalnom glodanju, odnosno pri obradi glodanjem trećeg kvadratnog ostrva i izradi džepova, jer nije izvodljivo sa C osom kao pomoćnim kretanjem. Isto važi i za zahvat glodanja radijalnog žljeba za čiju izradu Y osa nije potrebna, što znači da se takav zahvat može izvesti i na strugu sa tri numerički upravljane ose.

Savremeni proizvodni sistemi danas su praktično nezamislivi bez primene CAD/CAM sistema. CAD sistemi su obezbedili uspešno rešavanje pitanja automatizacije u području projektovanja proizvoda, dok se primenom numerički upravljanih mašina alatki realizuje automatizacija u području izrade proizvoda. Razvojem i primenom CAM sistema danas je uspešno rešen i problem automatizacije projektovanja tehnologije za numerički upravljane mašine alatke. Automatsko generisanje NC programa primenom CAD/CAM sistema predstavlja preovladavajući metod programiranja, koji je potisnuo ostale.

Dalje istraživanje predstavljene tematike bi moglo obuhvatiti upotrebu naprednih programskih modula CAM programskog sistema, kao i specijalizovanih softvera za optimizaciju putanje alata kroz definisanje tehnologije obrade, programiranje obradnog centra, izradu drugih radnih predmeta i analizu drugih tehnoloških zahvata obrade dostupnih na obradnim centrima za struganje-glodanje.

7. LITERATURA

[1] Zeljković, M., Tabaković, S., Živković, A., Živanović, S., Mlađenović, C., Knežev M.: *Osnove CAD/CAE/CAM tehnologija*, Univerzitetski udžbenik, Fakultet tehničkih nauka, Novi Sad, 2018.

[2] Zeljković, M., Tabaković, S., Antić, A.: *Programiranje numerički upravljanih obradnih sistema,* Fakultet tehničkih nauka, Novi Sad, 2015.

[3] Čiča, Đ., Jokanović, S.: *Programiranje numerički upravljanih mašina alatki*, Univerzitetski udžbenik, Univerzitet u Banjoj Luci, Mašinski fakultet, Banja Luka, 2014.

[4] Čiča, Đ.: *Mašine alatke*, Univerzitetski udžbenik, Univerzitet u Banjoj Luci, Mašinski fakultet, Banja Luka, 2016.

[5] Kief, H., Roschiwal, H.: *CNC-Handbuch*, Carl Hanser Verlag Munchen, 2017.

Kratka biografija:

Nebojša Mandić rođen je 1995. godine u Beogradu. Master rad na Fakultetu tehničkih nauka iz oblasti mašinstva odbranio je 2020. godine.
kontakt: mandicn95@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 007:004.056
DOI: https://doi.org/10.24867/12BE01Sunjka

ПРОШИРЕЊЕ АЛАТА ЗА АУТОМАТСКУ ДЕТЕКЦИЈУ ПРЕТЊИ СА *CWE* БАЗОМ ЗНАЊА И ЊЕГОВА УПОТРЕБА У РАЗВОЈУ БЕЗБЕДНОГ СОФТВЕРА

EXTENSION OF AN AUTOMATIC THREATS DETECTION TOOL WITH CWE KNOWLEDGE BASE AND ITS USE IN THE DEVELOPMENT OF SECURE SOFTWARE

Јово Шуњка, *Факултет техничких наука, Нови Сад*

Област – СОФТВЕРСКО ИНЖЕЊЕРСТВО И ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Кратак садржај – *У овом раду описан је алат за подршку SDL процеса и смернице за његову интеграцију у процес развоја безбедног софтвера.*

Кључне речи: *Претња, рањивост, слабост, SDL, моделовање претњи.*

Abstract – *This paper presents support tool for SDL process and guidelines for its integration into the development process of secure software.*

Keywords: *Threat, vulnerability, weakness, SDL, threat modeling.*

1. УВОД

Људи су одвајкада морали да обезбеде и заштите себе и своје ресурсе које поседују, тако је и у данашње време, само што се све дешава и у реалном и у дигиталном свету. Поред личних (приватних) података за које је неопходно обезбедити приватност, данас постоје и разне компаније које поседују одређене ресурсе или пружају одређене услуге у дигиталном свету, које желе да заштите од сваког вида злоупотребе. Да би се постигао висок степен безбедности, неопходно је приступити обезбеђивању софтверских система на систематичан и стандардизован начин, што спада у домен развоја безбедног софтвера (енгл. *Security Development Lifecycle, SDL*).

У контексту софтвера, *SDL* је процес који проширује животни циклус развоја софтвера (енгл. *Software Development Lifecycle)* тако што сваку његову фазу проширује безбедносним активностима (праксама) [1]. Најједноставније речено, *SDL* помаже инжењерима софтвера да развију безбеднији софтвер смањењем броја и озбиљности рањивости софтвера, истовремено смањујући трошкове развоја [1].

Кроз овај рад је описан алат за подршку *SDL* процеса уз смернице за његову интеграцију у процес развоја безбедног софтвера. Алат представља веб базирану апликацију за управљање рањивостима и слабостима у софтверу, са акцентом на *CWE* (енг. Common Weakness Enumeration) модулу.

Наведени модул помаже инжењерима да моделују претње, и дизајнирају, имплементирају и тестирају прикладне безбедносне механизме за посматрани софтвер.

2. *SDL*

SDL (енгл. *Security Development Lifecycle*) је процес који проширује Животни циклус развоја софтвера (енгл. *Software Development Lifecycle*) тако што сваку његову фазу проширује безбедносним активностима (праксама) [1]. Циљ *SDL*-а је да помогне инжењерима софтвера да развију безбеднији софтвер. Један од лидера у индустрији који дефинише и практикује *SDL* јесте *Microsoft*. Колико је *SDL* важан, говори и то да постоје индустријски стандарди (*IEC* [5], *NIST* [6]) који га стандардизују.

2.1. Преглед *SDL* приступа у индустрији

Microsoft дефинише *SDL* [7], као скуп од 16 безбедносних активности (пракси) које се интегришу у животни циклус развоја софтвера. Наведени скуп пракси није једини [7] и SDL процес предвиђа проширивање скупа пракси спрам потреба пројектног тима и предлога саветника за безбедност, како би допринеле повећању безбедности софтвера. Једну од обавезних пракси представља безбедносни тренинг (енгл. *security training*) који треба да претходи осталим праксама, а остале обавезне праксе су груписане по фазама традиционалног животног циклуса развоја софтвера.

Без обзира која методологија развоја софтвера се користи, едукација и тренинзи о безбедности су пресудни. Стога би требало следити стандардну *SDL* политику и најмање једном годишње обучити све инжењере о основним безбедносним питањима.

Фаза захтева састоји се од три обавезне праксе: безбедносни захтеви, капије квалитета и *bug bars*, и процена ризика за безбедност и приватност. У дизајн фази примењују се три обавезне праксе: захтеви за дизајн, смањивање површине напада и моделовање претњи. Фаза имплементације састоји се од три обавезне праксе: коришћење одобрених алата, застареле небезбедне функције и статичка анализа. У фаза верификације примењују се три обавезне праксе: динамичка анализа програма, *fuzz* тестирање и модел претњи и преглед површине за напад. *Release* фаза састоји се од три обавезне праксе: план реаговања на инциденте, завршни преглед безбедности и *release / archive*.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био доц. др Никола Лубурић.

Група *IEC* 62443 стандарда пружа флексибилан оквир за решавање и ублажавање тренутних и будућих безбедносних рањивости у индустријски аутоматизованим и управљачким системима (енгл. *industrial automation and control systems, IACS*). Конкретно, *IEC* 62443-4-1 стандард дефинише захтеве за безбедан развој производа који се користе у *IACS*-у и дефинише *SDL* за развој и одржавање безбедних производа [5]. По *IEC*-у, *SDL* треба да се састоји од следећих пракси: дефиниција безбедносних захтева, безбедан дизајн, безбедна имплементација (укључујући смернице за кодирање), верификација и валидација, *defect management*, *patch management* и крај животног века производа [5]. Ове праксе могу бити примењене на нове или на постојеће процесе за развој.

NIST SP 800-160 публикација говори о перспективи вођеној инжењерингом и радњама неопходним за развој система који ће бити више одбрањиви и одрживи. Идеја је да се безбедносна питања решавају тако што ће се одговорити на потребе и захтеве заинтересованих страна и да се користе проверени (устаљени) инжењерски процеси како би се постигло решавање са одређеним нивоом поузданости и одрживости система током целог животног циклуса [6]. Такође као и *Microsoft* и *IEC*, и *NIST* уводи безбедносне активности и задатке, који су развијени као безбедносна проширења процеса животног циклуса система. Активности и задаци заснивају се на принципима и концептима безбедности и поверења.

2.2. Преглед моделовања претњи

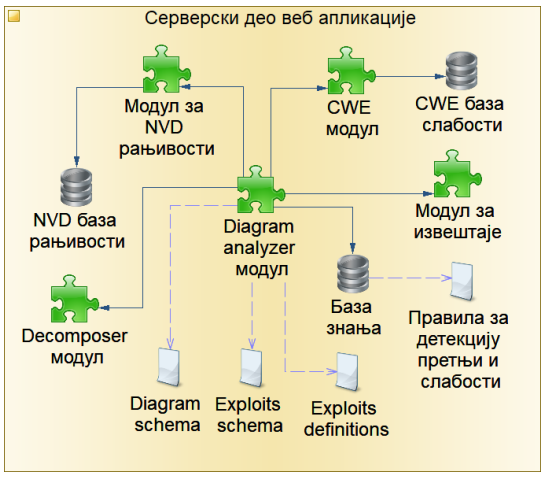
У овом раду, посебна пажња ће бити посвећена моделовању претњи, односно алату за моделовање претњи који је развијен како би се што више олакшало извршавање ове активности.

Моделовање претњи може се извести на више начина. Моделовање базирано на дијаграмима тока података (ДТП) је вероватно најбољи избор, јер често се визуелно могу „изговорити“ ствари које би било тешко објаснити речима, а оне су корисне и за техничку и за нетехничку публику, од програмера до извршног директора. У моделовању претњи могу учествовати чланови тима са различитим улогама. Најчешће улоге које учествују су: *product owner*, архитекта, *developer* и тестер. На сваку од ових улога модел претњи и генерално *SDL* имају значајан утицај. Задатак *product owner*-а је да дефинише потребну безбедност за производ.

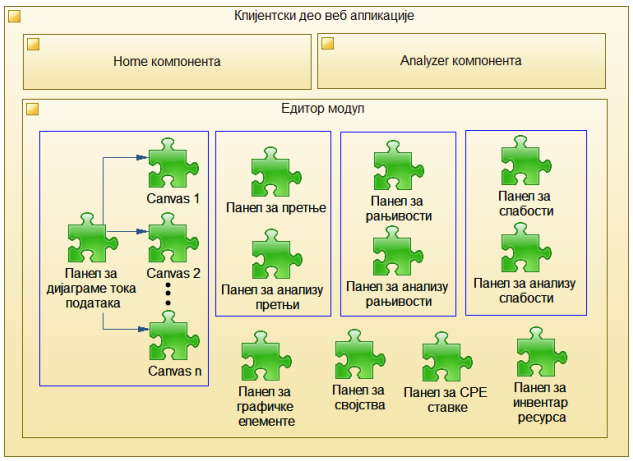
Његове одговорности су да одреди изворе безбедносних захтева и дефинише безбедносне захтеве за развојне тимове. Такође, *product owner* је тај који одлучује да ли ће тренутно већи приоритет имати нпр. развијање неке нове функционалности или решавање неког безбедносног проблема. Софтверски архитекта је задужен да од почетка прави *secure by design* софтвер. Његове одговорности су да: дизајнира вишеслојну одбрану, примењује *secure design* принципе и декомпонује производ за формирање модела претњи. *Secure design* принципи који су написани 1975. године, и данас важе и посебно се односе на безбедносни софтвер [7]. Сваки *developer* да би направио квалитетан софтвер мора да тежи ка писању безбедног кода. Безбедно кодирање се постиже тако што ће се усвојити најбоље праксе безбедног кодирања које су дефинисане у *SDL*-у [7]. У моделу претњи је могуће наћи потенцијалне слабости и рањивости, као и решења за њих. *Developer*-у је објашњено које конфигурације и методе из одговарајућих библиотека треба да користи, као и значење и утицај улазних параметара тих метода на целокупну безбедност софтвера. Да би софтвер био што квалитетнији неопходно га је тестирати, како због проналаска обичних багова, тако и због безбедносних рањивости. Модели претњи су непроцењиви документи које треба користити током тестирања безбедности софтвера. Моделе претњи треба користити за вођење и информисање о плановима за безбедносно тестирање. Такође, најризичнији делови апликације, обично они са највећом површином за нападе и претњама које су највећи ризици, морају се најтемељније тестирати. Кроз тестове је неопходно проверити да ли су исправно ублажене или у потпуности уклоњене слабости и рањивости које су претходно пронађене.

3. ПРЕЗЕНТАЦИЈА ИНТЕГРИСАНОГ АЛАТА

Апликација (алат) је настала интеграцијом две веб апликације: Систем за управљање рањивостима у софтверу (СУРС) [2] и Веб апликација за цртање дијаграма тока података (ВАЦДТП) [3], као и додавањем *CWE* модула. СУРС на улазу прима ДТП, декомпонује га на компоненте (шаблоне), затим анализира шаблоне и на крају креира извештај са потенцијалним нападима на софтвер и листом јавно идентификованих рањивости у софтверу [2]. Треба напоменути да је ова апликација настала ослањајући се *desktop* апликацију - Систем за проналажење напада [9]. ВАЦДТП је направљена са циљем да програмерима и софтверским инжењерима поједностави и олакша активност моделовања претњи користећи стандардне, врло интуитивне графичке елементе за приказивање ентитета из било ког софтверског система [3]. На сликама 1 и 2 може се видети архитектура серверског и клијентског дела интегрисаног алата након интеграције и након свих проширивања функционалности.



Слика 1. *Приказ архитектуре серверског дела апликације*



Слика 2. *Приказ архитектуре клијентског дела апликације*

3.1. Кратак преглед старих функционалности

Diagram analyzer је главни модул у серверском делу апликације. На почетку се врши валидација дијаграма тока података и *exploits* дефиниција по одговарајућим *XML* шемама. Затим се декомпоновање препушта *decomposer* модулу. Резултат декомпоновања су шаблони над којима се примењују правила из базе знања. Применом правила биће пронађени потенцијални напади. Затим се проналажење рањивости препушта модулу за *NVD* рањивости. Шаблони, потенцијални напади и рањивости се прослеђују модулу за извештаје како би креира извештај који ће бити испоручен на клијентски део апликације.

Decomposer служи за рашчлањавање дијаграма тока података на шаблоне, како би се смањила сложеност и извршило ефикасније детектовање и анализа претњи.

Модул за *NVD* (енгл. *National Vulnerability Database*) рањивости има две главне функционалности: скидање рањивости са *NVD*-а и претраживање скинутих рањивости по *CPE*-у (енгл. *Common Platform Enumeration*) производа (хардвер, софтвер).

Модул за извештаје се бави креирањем извештаја за анализиране дијаграме тока података. На почетку извештај буде формиран у *XML* формату, уз помоћ *XSLT* преводи се у *XHTML*, а затим се тај *XHTML* трансформише у *PDF* датотеку. У извештају се налазе пронађене претње, рањивости и слабости, као и предлози безбедносних контрола за њихово регулисање.

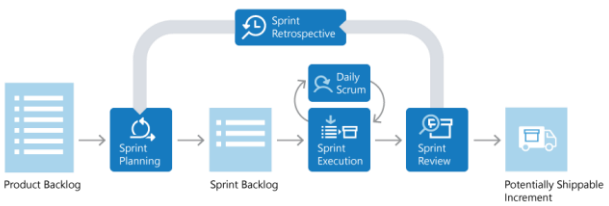
Едитор за дијаграме тока података састоји се од: панела за графичке елементе, панела за дијаграме тока података, *canvas*-а и панел за својства. Панел за графичке елементе садржи графичке елементе који се учитавају из конфигурационе *JSON* датотеке. Дефинисано је пар основних графичких елемената и то су: *process*, *complex – process*, *data – flow*, *external – entity* и *data - store*. Панел за дијаграме тока података је организован кроз табове. *Canvas* је површина на коју се могу превлачити графички елементи и на тај начин се може представити одређени систем. Могуће је изводити стандардне функционалности над графичким елементима. На панелу за својства приказана су својства селектованог графичког елемента и њихове вредности.

3.2. Преглед *CWE* модула

CWE модул има две главне функционалности: скрејповање података о *CWE*-овима и претраживање *CWE*-ова по *id*-ију. Скрејповање података о *CWE*-овима врши се са *cwe.mitre* сајта, а потом ће ти подаци бити кеширани. Треба истаћи да су *potential mitigations*, *detection methods* и *common consequences* подаци из којих *developer*-и, тестери и остали могу извући много корисних ствари. У *potential mitigations* је објашњено за сваку фазу Животног циклуса развоја софтвера шта се може урадити како би се ублажила одређена слабост система. У *detection methods* се препоручује коришћење разних метода као што су: ручна статичка анализа кода, аутоматска статичка анализа кода, аутоматска динамичка анализа кода, *fuzz* тестирање, како би се на време детектовала одређена слабост система. У *common consequences* се наводе различите последице повезане са слабошћу, идентификују се подручја безбедности апликације која су нарушена, описује се негативни технички утицај који ће настати ако нападач успе да искористи ову слабост, као и вероватноћа да ће се видети конкретна последица у односу на остале последице са списка. Тражење *CWE*-ова проузроковаће база знања, односно правила за детектовање слабости и претњи у систему, у току анализе шаблона екстрахованих из дијаграма тока података. По тренутној имплементацији, у тренутку када *CWE* буде затражен по *id*-ију, прво ће бити проверено да ли се тражени *CWE* налази међу кешираним *CWE*-овима, уколико не буде пронађен, биће извршено скрејповање података за тражени *CWE* са сајта. Уколико тражени CWE буде пронађен у кешу, он се допуњује додатним атрибутима: *diagramName*, *elementOrFlow* и *severity* (који ће бити израчунат по формули).

4. ИНТЕГРАЦИЈА АЛАТА СА *SDL* ПРАКСАМА У ОКВИРУ АГИЛНОГ РАЗВОЈА СОФТВЕРА

У тренутку када су агилне методологије постале најпопуларније и почеле да се користе у великом броју тимова широм света, појавила се потреба да се *SDL* интегрише са агилним методологијама тако да буду задржани принципи и агилних методологија и *SDL* процеса. Да би *SDL* захтеве (праксе) уклопио у агилни развој, *Microsoft* је сваку праксу из свог класичног *SDL*-а сврстао у једну од три категорије дефинисане по важности за безбедност, а самим тим и по учесталости извршавања тих пракси [9]. У питању су следеће категорије: *every-sprint* праксе, *bucket* праксе (примењују се периодично кроз неколико спринтова) и *one-time* праксе (примењују се на почетку пројекта). У наставку следи пролазак кроз ток рада (енгл. *workflow*) *Scrum* развојног оквира са фокусом на примени алата који је развијен, у циљу реализовања *SDL* пракси (слика 3).



Слика 3. *Приказ тока рада Scrum развојног оквира*

У фази планирања спринта пројектни тим се најпре договара око циља спринта, затим развојни тим одређује ставке из *product backlog*-а које су у складу са циљем спринта, а за које је реално да могу да се реализују у оквиру спринта.
Моделовање претњи је једна од *SDL* пракси која треба да буде примењена за сваки спринт по *SDL*-у за агилни развој. На самом почетку извршавања спринта најбоље би било убацити моделовање претњи које ће се односити само на ставке изабране за овај спринт (енгл. *sprint backlog*). Алат који је описан у овом раду је направљен да олакша моделовање претњи, тако да је ово прави тренутак да буде искоришћен. У едитору алата могуће је скицирати компоненте које се односе на ставке из *sprint backlog*-а, затим те компоненте детаљно описати у панелима за својства и *CPE* ставке, као и кроз панел за ресурсе који се односе на те компоненте, и на крају започети анализу. Алат ће у наставку покушати да детектује претње у систему. Резултат моделовања може бити *export*-овани ДТП који је у току спринта могуће мењати, као и извештај о детектованим претњама, рањивостима и слабостима система.
У току извршавања спринта, било би добро водити се саветима и препорукама које су дефинисане у моделу претњи, било да су генерисане од стране алата или их је нека особа стручна за безбедност унела кроз кориснички интерфејс. Конкретно, током имплементације и тестирања ставки из *sprint backlog*-а корисно би било погледати нпр. секцију за слабости (*CWE*-ове) у моделу претњи, и посебно обратити пажњу на текст који се односи на: потенцијалне митигације, методе детектовања и уобичајене консеквенце.
У току *sprint review*-а особа стручна за безбедност требало би да провери: да ли су *every-sprint* праксе испуњене или су одобрени изузеци за те праксе, да ли је примењена бар једна пракса из сваке категорије *bucket* пракси, да ни за једну *bucket* праксу није прошло дуже од шест месеци а да није примењена, да ниједна *one-time* пракса није премашила рок грејс периода и да нема отворен ниједан безбедносни баг изнад назначеног прага озбиљности (енгл. *severity*). Неки од ових задатака могу захтевати ручни напор стручне особе за безбедност како би се осигурало да је све завршено на задовољавајући начин. Пример једног таквог задатка био би прегледање модела претњи.
У току спринт ретроспективе развојни тим, *ScrumMaster* и *product owner* разматрају да ли у самом *Scrum* процесу који се примењује постоји простора за унапређење. У оквиру ове фазе тим би могао да се консултује и са саветником за безбедност око евентуалних измена на *SDL*-у, тј. око додавања, мењања или избацивања одређених *SDL* пракси, на основу искуства из претходног спринта.

5. ЗАКЉУЧАК

У овом раду описан је алат који представља подршку *SDL* процесу, као и смернице за његову интеграцију у процес развоја безбедног софтвера. Објашњено је шта представља *SDL*, дат је преглед *SDL* пракси и начин на које се оне постижу. Такође, објашњено је како моделовање претњи, као активност, утиче на све учеснике животног циклуса развоја софтвера. У трећем поглављу је презентован интегрисани алат са фокусом на *CWE* модул. Затим је представљено како се алат може интегрисати са *SDL* праксама из другог поглавља о описан је пословни процес употребе алата и *SDL* пракси у оквиру агилне методологије развоја.

Као први предлог за даљи развој ове апликације, предлаже се омогућавање креирања *template*-а на клијентском делу апликације и *upload*-овање *template*-а на серверски део апликације. *Template* би требало да садржи *stencils* (графичке елементе), дефинисане претње, рањивости и слабости. Такође, *template* би могао садржати и правила за детектовање претњи и слабости. Та правила би могла бити описана релативно једноставним домен - специфичним језиком (ДСЈ) који би требало дефинисати. ДСЈ би се преводио у *Drools* правила која се користе и у садашњем систему. На клијентском делу апликације требало би омогућити једноставно креирање правила за детектовање претњи кроз *GUI* (енгл. *Graphical User Interface*), а затим би тај *GUI* био повезан са ДСЈ изразима. Као други предлог, предлаже се детаљно изучавање *CWSS*-а и *CWRAF*-а, као и њихова примена у оквиру ове апликације.

6. ЛИТЕРАТУРА

[1] https://www.microsoft.com/en-us/securityengineering/sdl/practices (приступљено у септембру 2020)
[2] Марија Ковачевић, „Систем за управљање рањивостима у софтверу“, 2019.
[3] Јово Шуњка, „Веб апликација за цртање дијаграма тока података“, 2019.
[4] IEC, International Electrotechnical Commission, „62443-4-1: Security for industrial automation and control systems, part 4-1: Product security development life-cycle requirements“, 2018.
[5] Ron Ross, Michael McEvilley и Janet Carrier Oren, „Nist special publication 800-160: Systems security engineering considerations for a multidisciplinary approach in the engineering of trustworthy secure systems“, 2016.
[6] Michael Howard и Steve Lipner, „The Security Development Lifecycle“, 2006.
[7] Немања Миладиновић, „Проналажење рањивости у софтверу на основу дијаграма тока података“, 2017.
[8] https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ee790620(v=msdn.10) (приступљено у септембру 2020)

Кратка биографија:

Јово Шуњка рођен је 30. новембра 1996. године у Бања Луци, у Републици Српској (БиХ). Основне академске студије на Факултету техничких наука, на студијском програму Софтверско инжењерство и информационе технологије завршио је 2019. године. Тренутно завршава мастер академске студије на истом факултету.
контакт: sunjkajovo@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE02Novak

BIBLIOTEKA ZA GENERISANJE STANDARDNIH FORMI U ANGULARU

LIBRARY FOR GENERATING STANDARD FORMS IN ANGULAR

Kim Novak, Milan Vidaković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisana je specifikacija i implementacija biblioteke za generisanje standardnih formi u Angularu, kao i jednostavne veb aplikacije u koju je biblioteka integrisana.*

Ključne reči: *Angular, Web, REST, JSON*

Abstract – *This paper contains specification and implementation of the library for generating standard forms in Angular, as well as the specification and implementation of a simple web application which integrates the library.*

Keywords: *Angular, Web, REST, JSON*

1. UVOD

Manipulisanje podacima je sastavni deo skoro svakog informacionog sistema. Grafički korisnički interfejs pomoću kojeg se manipuliše podacima se najčešće sastoji od polja za unos grupisanih unutar forme. Ukoliko informacioni sistem koji razvijamo ima veliki broj entiteta, razvoj formi za svakog od njih je dug proces koji može rezultovati neodrživim sistemom. Da bi se ovaj problem rešio, potrebno je napraviti ponovno iskoristive forme koje su generične i implementiraju set funkcionalnosti koje su zajedničke za svaki od entiteta.

Standardne forme predstavljaju jedno rešenje tog problema i na njima se bazira rešenje opisano u ovom radu. Kako bi se razvoj informacionog sistema još više ubrzao potrebno je izdvojiti implementaciju standardnih formi u modul koji može da se integriše u bilo koji drugi informacioni sistem baziran na istim tehnologijama i da se na taj način standardne forme generišu, bez potrebe da se razvijaju ispočetka.

2. OPIS KORIŠĆENIH TEHNOLOGIJA

Biblioteka za generisanje standardnih formi napisana je u TypeScript jeziku, koristeći Angular radni okvir.

Klijentska aplikacija razvijena za potrebe demonstracije funkcionalnosti biblioteke, napisana je u istim tehnologijama, dok je serverska strana napisana koristeći JavaScript jezik, NodeJS i ExpressJS radni okvir.

2.1. TypeScript

JavaScript jezik je jedan od najpopularnijih jezika korišćenih za razvoj veb aplikacija.

JavaScript je dinamičan, slabo tipiziran jezik, koji je ili just-in-time kompajliran ili interpretiran. Ova tri svojstva JavaScript-a ostavljaju dosta prostora da programeri naprave greške prilikom programiranja koje se neće ispoljiti pre vremena izvršavanja. Te greške najčešće su vezane za implicitnu konverziju tipa promenljive.

TypeScript omogućava da se broj ovakvih grešaka svede na minimum, dodavanjem sloja oko JavaScript-a, a taj sloj je TypeScript-ov sistem tipova. TypeScript je i jezik i set alatki za generisanje JavaScript-a.

2.2. Angular

Angular je radni okvir baziran na TypeScript jeziku koji omogućava razvoj klijentskih aplikacija za različite platforme. Moguće je upotrebiti ga za razvoj:

- veb aplikacija,
- mobilnih aplikacija i
- desktop aplikacija.

Fundamentalni koncepti u Angularu su:

- NgModules,
- komponente,
- šabloni,
- direktive i
- umetanje zavisnosti

Arhitektura Angular aplikacije oslanja se na neke od fundamentalnih koncepata. Osnovni gradivni blokovi su NgModules, koji pružaju kontekst prevođenja (eng. compilation context) komponentama [1].

2.3. NodeJS

NodeJS je JavaScript izvršno okruženje. Pošto NodeJS omogućava da se JavaScript izvršava van internet pretraživača, moguće je koristiti ga za razvoj serverske strane veb aplikacije. Da bi se ubrzao razvoj serverske aplikacije u NodeJS-u koristi se neki od radnih okvira namenjenih razvoju veb aplikacija. Najpopularnije rešenje je ExpressJS.

Pomoću ExpressJS-a moguće je jednostavno i brzo razviti serversku aplikaciju spremnu za upotrebu.

3. SPECIFIKACIJA APLIKACIJE

Informacioni sistemi često zahtevaju implementiranje CRUD (Create, Read, Update, Delete) operacija za entitete koje sadrže. Entiteta može biti veliki broj i ukoliko bi se za svaki od entiteta svaki put ispočetka implementirale CRUD operacije, razvoj bi trajao dugo i sistem bi postao teško održiv.

Neophodno je prepoznati delove koji se mogu generalizovati i njih izdvojiti u ponovno iskoristivi modul.

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

Aplikacija rešava navedeni problem upotrebom standardnih formi. Standardne forme predstavljaju grafički korisnički interfejs koji omogućava jednostavan rad sa podacima. Operacije koje standardne forme pružaju su:

- pregled podataka,
- dodavanje novih podataka,
- izmena postojećih podataka,
- brisanje postojećih podataka i
- pretraga nad podacima.

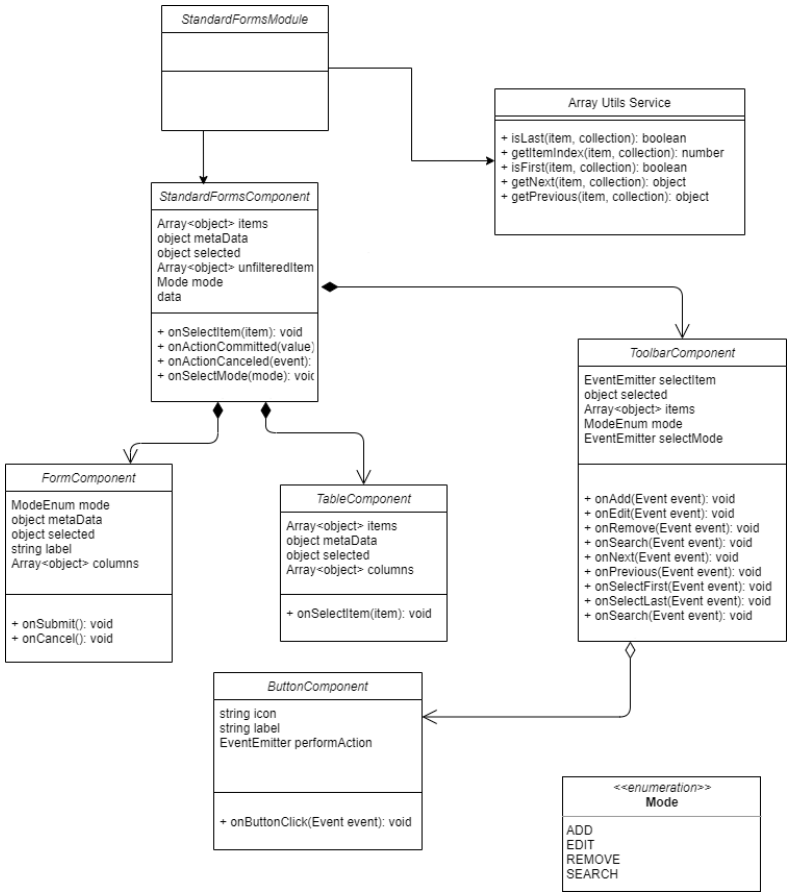
Sa aspekta funkcionalne analize zahteva, sistem će omogućiti korisniku sledeće funkcionalnosti:

- odabir režima rada, sistem se može nalaziti u režimu izmene, unosa, brisanja ili pretrage,
- unos podataka,
- čuvanje izmena i
- odbacivanje izmena

Na slici 1 prikazan je dijagram klasa. Biblioteka za generisanje standardnih formi je modul koji se sastoji od komponenti i servisa. Korenska komponenta je StandardFormsComponent koja u sebi sadrži ToolbarComponent, TableComponent i FormComponent.

ToolbarComponent sadrži više ButtonComponent-i.

Biblioteka za generisanje standardnih formi ima definisan konačan broj režima rada u kom forma može da bude, navedenih kroz nabrajanje. Ovo nabrajanje prikazano je u dijagramu pod nazivom Mode.



Slika 1. *Dijagram klasa*

4. IMPLEMENTACIJA

Biblioteka za generisanje standardnih formi implementirana je kao Angular biblioteka. Da bi se ispitala funkcionalnost ove biblioteke, integrisana je u jednostavnu Angular klijentsku aplikaciju koja komunicira sa serverom kako bi čitala i skladištila podatke.

4.1. Arhitektura

Arhitektura rešenja prati principe slojevite arhitekture (eng. Layered Architecture). Slojevita arhitektura podrazumeva organizaciju komponenti u slojeve, gde svaki sloj obavlja određenu ulogu u sistemu. Iako slojevita arhitektura ne određuje koliko slojeva mora postojati, većina slojevitih arhitektura sastoji se iz četiri sloja:

- prezentacioni sloj,
- sloj biznis logike,
- perzistencioni sloj i
- sloj baza podataka [2].

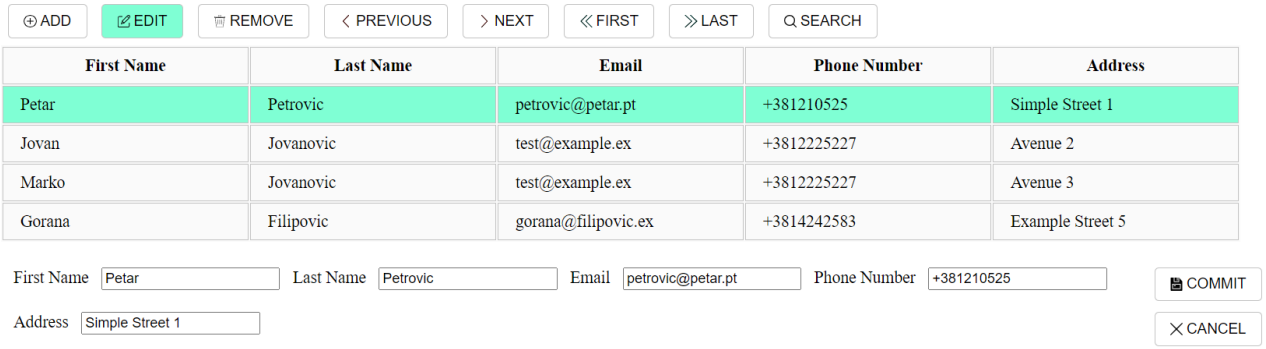
Aplikacija koja je razvijena za potrebe integracije biblioteke za generisanje standardnih formi spaja sloj biznis logike i perzistencioni sloj u jedan iz razloga što ne predstavlja rešenje namenjeno za komercijalnu ili širu upotrebu, već samo za demonstraciju funkcionalnosti biblioteke. Komunikacija između klijenta i servera obavlja se razmenom HTTP zahteva i odgovora. Serverska strana aplikacije implementirana je kao skup RESTful Web Service-a. Klijent i server razmenjuju podatke međusobno u JSON formatu.

4.2. Biblioteka za generisanje standardnih formi

Standardna forma omogućava korisniku da čita, unosi, menja, briše i pretražuje podatke. Sastoji se iz palete alatki, u kojoj korisnik može da izabere režim rada forme, kao i

alatke za jednostavnije i brže kretanje kroz podatke. Mogući režimi rada forme su režim izmene, režim unosa, režim brisanja i režim pretrage. Alatke za kretanje kroz podatke koje standardna forma omogućava su odabir prethodnog elementa, odabir sledećeg elementa, odabir prvog elementa i odabir poslednjeg elementa. Drugi sastavni deo standardne forme jeste tabela koja služi za prikaz podataka. Treći deo čini forma, koja se sastoji od polja za unos i dva dugmeta, dugmeta za potvrdu izmena i dugmeta za odbacivanje izmena. U zavisnosti od režima rada u kom se standardna forma nalazi, forma ima drugačije ponašanje. Ako se forma nalazi u režimu unosa, prilikom odabira režima, sva polja u formi će biti ispražnjena kako bi korisnik mogao lakše da unosi nove podatke. Ako se forma nalazi u režimu rada za izmenu, brisanje ili pretragu, polja za unos biće popunjena podacima trenutno odabranog elementa. Početno stanje forme je režim za izmene. Prikaz početnog stanja standardne forme dat je na slici 2. Na slici se vidi da je forma u režimu izmene i da je odabran prvi element u tabeli. Polja forme popunjena su podacima odabranog elementa.

Implementacija biblioteke za generisanje standardnih formi razlikuje se od već postojećih rešenja kao što je Kendo UI, po tome što uvodi još jedan nivo apstrakcije i time pojednostavljuje upotrebu biblioteke. Odnosno, meta podaci se prosleđuju biblioteci zajedno sa podacima, a dobavljaju se sa servera. Pomoću meta podataka generišu se tabela i forma, čime upotreba biblioteke postaje jednostavnija jer ne moraju eksplicitno da se definišu kolone i oznake.



Slika 2. *Početno stanje standardne forme*

Standardna forma implementirana je tako da se promene čuvaju tek nakon što korisnik potvrdi izmene klikom na dugme za potvrdu izmena. Nakon što korisnik klikne na dugme za potvrdu izmena, podaci iz forme se prosleđuju roditeljskoj komponenti, u ovom slučaju to je StandardFormsComponent. Slanje podataka roditeljskoj komponenti u Angularu je implementirano emitovanjem prilagođenog događaja u koji se smeštaju podaci koje je potrebno poslati.

Roditeljska komponenta osluškuje na prilagođeni događaj i kada ga dete komponenta emituje, roditeljska komponenta reaguje na događaj pozivom metode za obrađivanje tog događaja.

Komponenta StandardFormsComponent potom, na osnovu režima rada u kom se nalazi, emituje odgovarajući događaj svojoj roditeljskoj komponenti, odnosno aplikaciji koja integriše ovu biblioteku. Na ovaj način, implementacija dodavanja, izmene i brisanja prepuštena je sistemu koji integriše biblioteku.

Biblioteka reaguje na svaku promenu podataka od strane aplikacije koja je integriše i u skladu sa novim podacima osvežava komponente.

Korisnik može da odbaci izmene koje je uneo klikom na dugme za odbacivanje izmena. U zavisnosti od režima rada u kom se nalazi standardna forma, događaj odbacivanja izmena izazvaće različito ponašanje. Ako se forma nalazi u režimu unosa podataka, odbacivanje izmena će isprazniti sva polja za unos. Ako se forma nalazi u režimu izmene ili brisanja podataka, odbacivanje izmena će vrednosti u poljima za unos vratiti u stanje u kom su bili pre izmena. Ako se forma nalazi u režimu pretrage podataka, odbacivanje izmena će isprazniti sva polja za unos i poništiti filtere koji su primenjeni nad podacima. Osim izmene stanja polja za unos, nakon što se desi događaj odbacivanja izmena, komponenta Form će emitovati prilagođeni događaj roditeljskoj komponenti StandardFormsComponent. Roditeljska komponenta osluškuje na ovaj događaj i kada se on desi poziva metodu za obradu ovog događaja koja će poništiti primenjene filtere ukoliko je forma bila u režimu rada pretrage podataka.

4.3. Klijentska aplikacija

Sistem u koji je integrisana biblioteka je jednostavan i služi za manipulisanje podacima o korisnicima. Klijentska aplikacija napisana je u Angular-u i omogućava prikaz, dodavanje, izmenu, brisanje i pretragu korisnika.

Klijentska aplikacija komunikacijom sa serverom dobija podatke i meta podatke koje treba da prosledi biblioteci za generisanje standardnih formi. Da bi se ostvarila komunikacija sa serverom, Angular pruža HttpClient servis klasu u sklopu @angular/common/http paketa.

Kako bi klijentska aplikacija integrisala biblioteku za generisanje standardnih formi, neophodno je ubaciti taj modul kao zavisnost u klijentsku aplikaciju. Ubacivanjem StandardFormsModule-a u glavni modul klijentske aplikacije omogućava korišćenje komponenti koje StandardFormsModule izvozi.

Klijentska aplikacija potom vrši vezivanje podataka dobijenih sa servera na podatke koji su potrebni standardnoj formi. Takođe, vezuju se i obrađivači događaja za događaje koje emituje standardna forma.

4.4. Serverska aplikacija

Serverska aplikacija je jednostavna NodeJS aplikacija napisana koristeći ExpressJS. Aplikacija omogućava čitanje, čuvanje, izmenu i brisanje podataka o korisnicima. Sadrži jedan endpoint i to /users kojem je moguće slati zahteve GET, OPTIONS, POST, PUT i DELETE HTTP metodama. Aplikacija očekuje podatke u JSON formatu i za prebacivanje JSON string-a u JavaScript objekat koristi bodyParser. Podaci se mogu čuvati i lako dobavljati iz baza podataka, ali se za potrebe demonstriranja funkcionalnosti čuvaju u memoriji procesa, odnosno ne čuvaju se trajno. Podaci su podeljeni u dva segmenta, jedan koji predstavlja meta podatke i drugi koji predstavlja podatke koji su rezultat upita.

Uloga serverske aplikacije jeste da meta podatke, zajedno sa podacima prebaci u JSON format. Meta podaci se koriste na klijentskoj strani za generisanje standardne forme. Serverska aplikacija registruje 4 metode za obradu HTTP zahteva koji stignu na /users endpoint. Prilikom primanja zahteva na osnovu HTTP metode određuje se koja od tih metoda će obraditi pristigli zahtev. Nakon obrade zahteva, serverska aplikacija šalje klijentskoj aplikaciji HTTP odgovor.

5. ZAKLJUČAK

Zadatak ovog rada bio je da se razvije biblioteka za generisanje standardnih formi. Standardne forme su generične, odnosno ponovno iskoristve za različite entitete u informacionom sistemu, čime je razvoj informacionog sistema znatno ubrzan jer se ne razvija forma za svaki entitet posebno.

Zadatak je implementiran upotrebom Angular radnog okvira za razvoj klijentskih aplikacija, baziran na TypeScript jeziku. Sistem u koji je biblioteka integrisana za potrebe demonstriranja funkcionalnosti razvijen je upotrebom Angular-a i NodeJS-a. Angular je stabilan i moderan radni okvir, koji pruža mnogo funkcionalnosti out-of-the-box, što smanjuje broj 3rd party biblioteka o kojima zavisi aplikacija.

Biblioteka za generisanje standardnih formi dodatno je ubrzala razvoj informacionog sistema iz razloga što se čitav grafički korisnički interfejs i funkcionalnosti mogu dobiti jednostavnom integracijom u sistem, bez potrebe da se forma razvija ispočetka.

Razvojem biblioteke za generisanje standardnih formi omogućeno je da bilo ko može da je integriše u svoj informacioni sistem ukoliko to želi. Prednost ove biblioteke u odnosu na druge biblioteke sličnog tipa, jeste ta da se struktura podataka ne mora definisati eksplicitno, već se može pročitati iz meta podataka dobijenih sa servera, što čini integraciju u sistem jednostavnijom.

Biblioteku za generisanje standardnih formi moguće je dodatno unaprediti dodavanjem prilagođene validacije, čija pravila bi bila definisana u meta podacima. Još jedan vid poboljšanja bi mogao biti da se u meta podacima definišu i relacije između entiteta, kako bi mogla da se implementira Zoom funkcionalnost, odnosno da kada se zumira entitet, da se prikaže detaljno entitet kojeg on referencira.

6. LITERATURA

[1] https://angular.io/guide/architecture (pristupljeno u oktobru 2020.)

[2] https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html (pristupljeno u oktobru 2020.)

Kratka biografija:

Kim Novak rođena je 26.03.1993. godine u Novom Sadu. 2017. godine završila završila je osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu, smer Računarstvo i automatika i stekla zvanje diplomiranog inženjera elektrotehnike i računarstva. Iste godine na istom fakultetu upisuje master akademske studije smer Primenjene računarske nauke i informatika, modul Softversko inženjerstvo.

Milan Vidaković je rođen u Novom Sadu 1971. godine. Na Fakultetu tehničkih nauka u Novom Sadu završio je doktorske studije 2003. godine. Na istom fakultetu je 2014. godine izabran za redovnog profesora iz oblasti Primenjene računarske nauke i informatika.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4:621.395
DOI: https://doi.org/10.24867/12BE03Kostic

RAZVOJ PROGRESIVNIH VEB APLIKACIJA NA PRIMERU IONIC I POLYMER OKRUŽENJA

DEVELOPMENT OF PROGRESSIVE WEB APPLICATIONS USING IONIC AND POLYMER FRAMEWORK

Jelena Kostić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad predstavlja poređenje razvoja progresivnih veb aplikacija pomoću Ionic i Polymer okruženja*

Ključne reči: *Progresivne veb aplikacije, Ionic, Polymer, Javascript, HTML, WebRTC, Firebase*

Abstract – *This paper deals with comparisson of progressive web applications development using Ionic and Polymer frameworks.*

Keywords: *Progressive web applications, Ionic, Polymer, Javascript, HTML, WebRTC, Firebase*

1. UVOD

Zadatak rada predstavlja poređenje razvoja progresivnih veb aplikacija putem namenskih okruženja. Kao tema rada, odabrana su dva popularna okruženja za razvoj progresivnih veb aplikacija, Ionic [1] i Polymer [2]. Za demonstraciju rada su implementirane dve aplikacije iz domena *chat* aplikacija, napisane koristeći navedena okruženja.

Ionic i Polymer okruženja su odabrana kao okruženja popularna u oblasti razvoja progresivnih veb aplikacija, ali ne i klasičnih veb aplikacija. Prilikom implementacije, za implementaciju *back-end* dela aplikacije je korišćena *Firebase* [3] platforma za mobilne aplikacije, kao i WebRTC [4] tehnologija za pristup multimedijalnim periferijama uređaja.

2. TEHNOLOGIJE PROGRESIVNIH VEB APLIKACIJA

Progresivne veb aplikacije predstavljaju posebnu vrstu klasičnih veb aplikacija koje korisniku pružaju isto iskustvo u korišćenju aplikacije nezavisno od uređaja na kome je koristi. Pravljene su na temelju istih tehnologija na kojima se prave i klasične veb aplikacije, ali se od istih razlikuju u tome što korisniku pružaju osećaj i performanse korišćenja aplikacije pravljene namenski za korisnikov uređaj.

Da bi se stvorio navedeni utisak, kod dizajniranja i implementacije progresivnih veb aplikacija akcenat se stavlja na dva koncepta – *responsive* dizajn, koji omogućava prilagođavanje sadržaja stranice uređaju na kome se ona gleda, i *native* funkcionalnosti, poput brzine, funkcionisanja bez zavisnosti mreže i keširanja resursa.

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

Prema definiciji postavljenoj od strane Aleksa Rasela [5], osnovni koncepti progresivnih veb aplikacija su: *responsive* dizajn, nezavisnost od povezanosti na Internet, osećaj korišćenja *native* aplikacije, ažurnost, sigurnost, mogućnost pronalaženja, mogućnost interakcije sa korisnikom nezavisno od njegove interakcije sa aplikacijom, mogućnost instalacije i mogućnost deljenja.

Minimalni uslovi koje progresivna aplikacija treba da ispuni da bi se mogla tako i nazvati jesu da: bude dostupna putem TLS protokola, bude dostupna *offline* i sadrži *web manifest*.

2.1. Service worker i web manifest

Service workeri predstavljaju posebnu vrstu *web workera*, Javascript koda koji se izvršava u pozadini aplikacije bez blokiranja UI niti. Glavna uloga *service workera* je da presreću HTTP zahteve koji dolaze iz aplikacije za koju je registrovan. Presretanjem HTTP zahteva se omogućava keširanje resursa i podrška za *offline* funkcionisanje aplikacije.

Manifest veb aplikacije predstavlja datoteku koja sadrži podatke o aplikaciji neophodne prilikom preuzimanja i instalacije. Manifest veb aplikacije je napisan u vidu JSON datoteke i sadrži opšte informacije o aplikaciji, informacije o instalaciji, navigaciji i izgledu aplikacije, kao i putanje do resursa korišćenih u aplikaciji. Manifest se dodaje u aplikaciju putem *link* HTML elementa u kome se navodi putanja do JSON datoteke.

Manifest predstavlja jedan od kriterijuma za instalaciju progresivne veb aplikacije na uređaj. Pošto svaki pretraživač različito definiše kriterijume za instalaciju aplikacije, sadržaj manifesta koji je dovoljan za instalaciju na neki uređaj se razlikuje između njih.

2.2. Ionic okruženje

Ionic predstavlja open source okruženje za razvoj aplikacija za veb i mobilne platforme. Može se koristiti sa nekim od popularnih UI okruženja, poput Angular-a, React-a ili Vue.js-a, ili se može koristiti sa Javascriptom, bez posrednog okruženja. Ionic aplikacije se mogu postaviti na Internet kao progresivne veb aplikacije, ili se mogu objaviti na Google Play Store-u ili App Store-u kao hibridne aplikacije.

Ionic sadrži bogatu biblioteku UI komponenti koje se mogu koristiti i u mobilnim i u veb aplikacijama. Navedene komponente su responsivne i dizajnirane poštujući principe Material Design-a.

Ionic Native predstavlja set Ionic biblioteki koje se mogu iskoristiti da bi se korisniku omogućile funkcije dostupne na mobilnim uređajima. Navedene biblioteke se koriste zajedno sa okruženjima za razvoj hibridnih mobilnih aplikacija. Ionic okruženje takođe sadrži i set CLI instrukcija za jednostavniji razvoj aplikacija, pokrivajući ceo životni ciklus istih.

2.2. Polymer okruženje

Polymer predstavlja open source biblioteku razvijenu od strane Google-a. Prednost Polymera jeste to što, sem što pruža gotov skup komponenti koje se mogu koristiti, takođe pruža mogućnost za definisanje i publikovanje sopstvenih komponenti na javni repozitorijum komponenti.

Akcenat Polymer komponenti jeste na jednostavnosti i brzini iscrtavanja komponenti, čime se povećava efikasnost aplikacije. Najnovija vezija biblioteke, nazvana *LitElement*, uvodi inovaciju u odbacivanju dvosmernog vezivanja za model (*two-way binding*) i koristi samo jednosmerno vezivanje za model (*one-way binding*).

Polymer ne sadrži skup biblioteka za omogućavanje funkcionalnosti vezanih za mobilne aplikacije. Umesto toga, za omogućavanje navedenih funkcija se mogu koristiti druge Javascript biblioteke koje pružaju te mogućnosti. Polymer, poput Ionic-a, takođe sadrži set CLI instrukcija za jednostavniji razvoj aplikacija.

2.3. WebRTC

WebRTC (Web Real-time Communication) projekat predstavlja tehnologiju koja veb aplikacijama omogućava razmenu zvuka i videa koristeći *real-time* komunikacione protokole. Tehnologija je standardizovana od strane W3C-a, dostupna u vidu Javascript API-ja i podržana od većine modernih pretraživača.

Osnovu WebRTC-a predstavljaju protokoli za pristup multimedijalnim uređajima i *peer-to-peer* protokoli. Multimedijalni uređaji koriste standardizovani Javascript Media Device API i tako omogućavaju pristup multimedijalnim periferijama uređaja. *Peer-to-peer* tehnologije omogućavaju razmenu informacija između uređaja, čineći time implementaciju protokola poput VoIP mogućom.

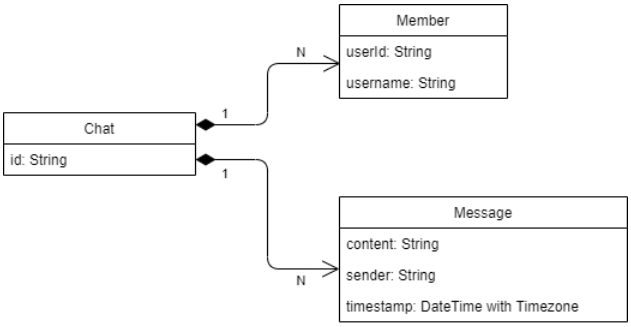
2.4. Firebase

Firebase predstavlja Google platformu za kreiranje mobilnih aplikacija. Platforma pruža niz usluga koje pokrivaju kreiranje, unapređivanje i razvoj mobilnih aplikacija. Kao neke od Firebase usluga, platform pruža mogućnost integracije analitike aplikacije, sistema za autorizaciju, sistema za čuvanje datoteka, NoSQL baze, hostinga aplikacije na cloud serveru itd. Tehnologija se razvija od strane Google-a, i moguća je integracija iste unutar mobilnih i veb platformi.

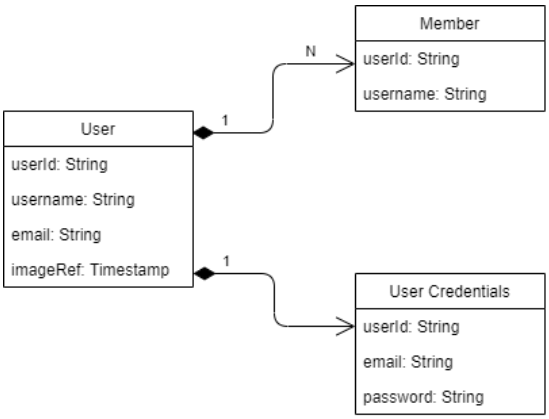
2.5. Specifikacija

Za domen aplikacije je odabrana *chat* aplikacija, pomoću koje korisnici mogu da šalju poruke drugim korisnicima, dodaju ih u kontakte i kreiraju svoje profilne fotografije. Aplikacija je kreirana u dva okruženja koji su tema rada. Aplikacije su koncipirane da izgledom podsećaju jedna na drugu, radi jednostavnijeg poređenja.

Na slikama 1 i 2 je dat model baze podataka. Firebase platforma nudi NoSQL bazu kao rešenje za perzistenciju podataka, tako da je baza sačinjena od dve kolekcije podataka – kolekcija konverzacija i kolekcija korisnika.



Slika 1. *Model dokumenta iz kolekcije konverzacija*



Slika 2. *Model dokumenta iz kolekcije korisnika*

Pristizanje poruka se dešava u realnom vremenu, bez ručnog osvežavanja interfejsa. Rad aplikacije je moguć i bez konekcije na Internet pomoću mehanizama keširanja resursa.

Promena profilne fotografije se obavlja koristeći prednju kameru uređaja. Pritiskom na odgovarajuće dugme se otvara interfejs kamere pomoću kog je moguće napraviti fotografiju. Kada korisnik odluči da je zadovoljan napravljenom fotografijom, kamera vraća nazad rezultujuću sliku koja se čuva u bazi i postavlja kao nova slika.

2.6. Implementacija

Prilikom implementacije aplikacija, naglasak je bio na mogućnosti rada aplikacije na glavnim mobilnim platformama- Android i iOS. Pre započinjanja rada su definisani grafički interfejs i glavni elementi interakcije, radi bolje preciznosti prilikom implementacije interfejsa. Oba okruženja sadrže komponente kreirane pomoću *WebComponents* standarda. Komponente su dostupne na javnim repozitorijumima putem *Node Package Manager*-a.

Kao jedan od glavnih izazova prilikom izrade rada, izdvojene su dve glavne funkcionalnosti aplikacije. Prva i osnovna funkcionalnost *chat* aplikacije jeste slanje i primanje poruka u realnom vremenu. Prilikom implementacije funkcionalnosti je korišćen *Firebase Firestore API*, koji pruža mogućnost podešavanja *observera* na određeni

set podataka i definisanja funkcionalnosti koja će se izvršiti svaki put kada se posmatrani podaci promene. Navedena osobina je iskorišćena za osvežavanje podataka prilikom primanja nove poruke. Time je uklonjena potreba za ručnom implementacijom *WebSocket* protokola koji bi omogućavao da server kontaktira korisničku aplikaciju prilikom primanja poruke.

Naredni izazov je predstavljala implementacija funkcionalnosti kamere koja korisniku pruža mogućnost da promeni svoju profilnu sliku. Prilikom implementacije funkcionalnosti je bilo potrebno kreirati poseban interfejs koji će sadržati funkcionalnosti kamere.

Za pristup kameri je iskorišćen navedeni WebRTC protokol, koji omogućava da se, koristeći Media Device API, pristupi kameri, prikažu podaci koji kontinualno dolaze sa kamere i da se ti podaci na kraju preuzmu kao profilna slika korisnika. Na listingu 1 je data implementacija inicijalizacije kamere, dok je na listingu 2 data implementacija preuzimanja fotografije i oslobađanja resursa.

```
openCamera() {
 const constraints = {
  video: true
 };

 navigator.mediaDevices.getUserMedia(constraints)
  .then(stream => {
  this.mediaStream = stream;
  var video = document.querySelector('video');
  video.setAttribute('style', 'border: 1px solid
black; position: fixed; left: 0px; top: 0px');
  document.getElementById('camera-capture')
.setAttribute('style', 'position: fixed; left:0px;
top: 0px');

  if ("srcObject" in video) {
   video.srcObject = this.mediaStream;
  } else {
   video.src = window.URL
     .createObjectURL(this.mediaStream);
  }
  video.onloadedmetadata = function(e) {
   video.play();
  };
 }
);
```

Listing 1. *Inicijalizacija kamere*

```
const takePhoto = (() => {
 const canvas = document.getElementById('canvas');
 const context = canvas.getContext('2d');

 const vid = document.querySelector('video');
 canvas.width = vid.videoWidth;
 canvas.height = vid.videoHeight;
 context.drawImage(vid, 0, 0, vid.videoWidth,
vid.videoHeight);

 const imgData = canvas.toDataURL('image/png');
 const imageName = new Date().getTime();
 this.userService.updateImage(this.userId, imageName,
imgData).then(() => {
  this.userService.updateProfile(this.userId,
imageName).then(() => {});
  this.mediaStream.getTracks()[0].stop();
 });

 const video = document.querySelector('video');
 video.setAttribute('style', 'border: 1px solid
black; position: fixed; left: 0px; top: 0px; display:
none');
 const button = document.getElementById('camera-
capture');
 button.setAttribute('style', 'position: fixed;
   right:0px; top: 0px; display: none');
}).bind(this);

document.getElementById('camera-capture')
    .addEventListener("click", takePhoto);
```

Listing 2. *Prikaz preuzimanja profilne fotografije sa kamere, čuvanja fotografije i oslobađanja resursa*

Nakon završetka, obe aplikacije su postavljene na javne domene putem TLS protokola. Funkcionalnost instalacije progresivne aplikacije je proverena na Samsung A50 fizičkom uređaju, dok je za testiranje iOS uređaja korišćen BrowserStack [6] online emulator.

3. ZAKLJUČAK

U današnje vreme je primetan porast u broju progresivnih veb aplikacija i preduzeća koja se prebacuju na ovakav vid aplikacija. Nakon prelaska, preduzeća navode primetan porast u broju korisnika, njihovom zadovoljstvu i samim time porastom profita koji aplikacija donosi. Danas se progresivne veb aplikacije smatraju budućnošću *e-commerce* modela poslovanja.

Ionic i Polymer okruženja se i danas aktivno razvijaju, i samim time, postoji mogućnost u unapređivanju aplikacije koja je korišćena za demonstraciju rada. Ionic okruženje prednjači u broju korisnika i načinu dokumentacije sistema, dok je prednost Polymera u njegovoj prilagodljivosti i mogućnosti podešavanja.

4. LITERATURA

[1] Ionic okruženje, https://ionicframework.com
[2] Polymer okruženje, https://www.polymer-project.org/
[3] Firebase sistem, https://firebase.google.com/
[4] WebRTC protokol, https://webrtc.org/
[5] Progressive apps: Escaping Tabs Without Losing Our Soul, https://medium.com/@slightlylate/progressive-apps-escaping-tabs-without-losing-our-soul-3b93a8561955
[6] Browser Stack, https://www.browserstack.com/

Kratka biografija:

Jelena Kostić rođena je u Rumi 1995. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Primenjene računarske nauke i informatika odbranila je 2018.god.
kontakt: kosticka.jelena@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4:621.395
DOI: https://doi.org/10.24867/12BE04Jovic

REACT KOMPONENTA ZA KONTROLU PRISTUPA BAZIRANU NA KORISNIČKIM ULOGAMA

REACT COMPONENT FOR ROLE-BASED ACCESS CONTROL

Vukašin Jović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu će biti opisan sistem za dinamičko generisanje komponenti na korisničkoj strani veb aplikacije implementiranoj korišćenjem React radnog okvira i RBAC kontrole pristupa.*

Ključne reči: *Kontrola pristupa, Veb aplikacija, React, RBAC*

Abstract – *This paper will present a system for dynamic component generation on the client side of a web application using React framework and RBAC.*

Keywords: *Access control, Web application, React, RBAC*

1. UVOD

Veb aplikacije često sadrže stranice i podatke koji se prikazuju samo određenim korisnicima. Svim podacima koji ne zahtevaju određeno pravo pristupa može se pristupiti direktno prilikom posećivanja određene stranice, kao što su na primer početna stranica, „o nama“ stranica i slično. Sa druge strane, stranice koje su specifične za pojedinačnog korisnika zahtevaju da se korisnik na neki način identifikuje sistemu, kako bi mu bile dostupne.

Ovaj rad bavi se konceptom kontrole pristupa u veb aplikacijama na osnovu uloga u sistemu. Prikazuje specifikaciju i implementaciju sistema za dinamičko generisanje odnosno aktivaciju/deaktivaciju komponenti na klijentskoj strani veb aplikacije koristeći korisničke role tj. uloge. Korisniku se u zavisnosti od uloge prikazuju određene stranice kao i akcije na stranicama koje može izvršiti.

2. OSNOVNI POJMOVI I DEFINICIJE

Za bolje razumevanje rada potrebno je prvo razumeti koncepte veb aplikacija, korisnikovom identifikovanju sistemu – autentifikaciji, kao i kontroli pristupa baziranoj na korisničkim ulogama.

2.1. Veb aplikacija

Veb aplikacije predstavljaju aplikacije kojima se pristupa putem interneta koristeći internet pretraživač (eng. web browser) i koje su razvijene koristeći jezike prihvatljive od strane samog pretraživača poput *HTML*-a (Hyper Text Markup Language), *JavaScript*-a i slično.

Da bi mogle da funkcionišu, oslanjaju se na internet pretraživače i mnoštvo poznatih aplikacija poput veb prodavnica, aukcija i mejl klijenata [1]. Pružaju mogućnost pokretanja aplikacije na različitim platformama i operativnim sistemima, pružaju istu verziju aplikacije za sve korisnike, ne zahtevaju prethodnu instalaciju što isključuje ograničenja memorijskog prostora i slično. Pojedine aplikacije su dinamičke, što znači da zahtevaju procesiranje na serverskoj strani, dok su druge statičke i ne zahtevaju nikakvo procesiranje već samo prikazuju statički sadržaj.

Veliki broj današnjih veb sajtova sadrži značajne količine koda na klijentskoj strani s obzirom da postoje kompleksni korisnički interfejsi i dosta podataka mora da se premešta i menja u toku vremena [2]. Sa druge strane, aplikaciju je potrebno napraviti brzom kako korisnik ne bi morao mnogo da čeka prilikom njenog učitavanja. Iz tog razloga potrebno je koristiti određenu arhitekturu koja pruža sve te mogućnosti kao i mogućnost jednostavnog dodavanja novih opcija, otklanjanja bagova i skaliranja cele aplikacije.

Za razvoj ovakve aplikacije mogu se koristiti razne javascript biblioteke i radni okviri, kao što je *React*. Radni okviri veb aplikacija su dizajnirani da pojednostave proces programiranja i promovišu kod postavljanjem biblioteka, različitih struktura, dokumentacije i smernica.

Za React se može reći da predstavlja *View* iz *MVC* šablona, međutim nije striktno određeno kako se mora koristiti. Kreira apstraktnu reprezentaciju pogleda i razdvaja delove pogleda na celine koje se nazivaju komponentama. Ove komponente sadrže i logiku za rukovanje pogledom ali i sam pogled i mogu biti nadograđenje i ponovo iskorišćene.

Komponenta može sadržati podatke koji se koriste kako bi se generisalo određeno stanje aplikacije. Stanje predstavlja objekat koji sadrži informacije koje pripadaju samo toj komponenti. Prilikom promene stanja, izvršavaju se algoritmi koji odlučuju da li je neophodno opet učitati komponentu i na koji način je to najefikasnije uraditi. S obzirom da je React JavaScript biblioteka, ona manipuliše *DOM*-om (*Document Object Model*).

Kako je ta operacija dosta skupa po pitanju resursa, *React* umesto pravim *DOM*-om manipuliše virtualnim. Promene i provera unutar virtuelnog *DOM*-a je brža i efikasnija što i sam *React* čini brzim i efikasnim [3, 4].

2.2. Autentifikacija

Autentifikacija je proces u kom se utvrđuje da je korisnik, odnosno entitet, ono što tvrdi da jeste. Autentifikacija kao takva predstavlja osnovni oblik pristupa veb aplikaciji. Da

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

ovaj mehanizam ne postoji, aplikacija bi sve korisnike tretirara kao goste (eng. guest) odnosno anonimne korisnike [5]. Postoje dva tipa autentifikacije: korisnička autentifikacija i entitetska autentifikacija. Korisnička identifikacija je najčešće predstavljena korsničkim nalogom, gde se korisnik sistemu identifikuje putem jedinstvenog identifikatora (uglavnom korisničkog imena ili elektronske pošte - *e-mail*) i lozinke.

Nakon unetih podataka, aplikacija proverava njihovu validnost te ukoliko je ona uspešno izvršena, sesijski token (eng. session token) se smešta u korisnikov pretraživač.

Na taj način korisnički pretraživač svaki put prilikom slanja novog zahteva šalje i sesijski token što predstavlja entitetsku identifikaciju. Imajući to u vidu, entitentska autentifikacija vrši se prilikom slanja svakog zahteva, dok se korisnička autentifikacija vrši samo jednom u toku sesije [6].

2.3. Kontrola pristupa

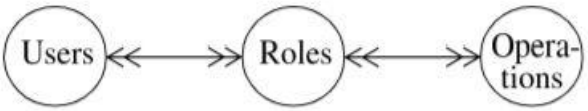
Kontrola pristupa i njeni mezanizmi čine ključni element pri projektovanju bezbednosti svake veb aplikacije. Podaci sa klijentske (eng. front-end) i sa serverske (eng. back-end) strane kao i sistemski resursi bi trebalo da budu zaštićeni od strane aplikacije. Kontrola pristupa bi trebalo da zaštiti podatke od neautorizovanog gledanja, menjanja i kopiranja ograničavanjem korisnikovih radnji, pristupa resursima kao i funkcijama koje manipulišu tim resursima [5].

Pojam koji se često zamenjuje kontrolom pristupa jeste pojam autorizacije. Ona predstavlja proveru odrovarajuće dozvole korisnika da pristupi određenom podatku ili izvrši određenu akciju, pod pretpostavkom da se korisnik prethodno uspešno autentifikovao. Autorizacija se zasniva na specifičnim pravilima liste kontrole pristupa koje unapred određuju administratori aplikacije ili vlasnici podataka.

Neke od čestih provera autorizacije jesu: ispitivanje članstva u određenoj grupi korisnika, pristustvo korisnika na listi odobrenih korisnika resursa ili posedovanje odgovarajućeg odobrenja. Svaki mezanizam kontrole pristupa koji se korisni za autorizaciju neposredno zavisi od efikasnih zaštićenih kontrola autentifikacije [6].

Kontrola pristupa zasnovana na ulogama (*Role-based access control – RBAC*) predstavlja kontrolu pristupa gde su odluke pristupa zasnovane na korisnikvim individualnim ulogama i odgovornostima unutar organizacije [7].

Uloga se pravilno posmatra kao semantička konstrukcija oko koje se formuliše politika kontrole pristupa. Određeni skup korisnika i dozvola koje okuplja jedna uloga je privremeni. Sa druge strane, uloga je stabilnija jer se aktivnosti ili funkcije organizacije ređe menjaju. Uloga može predstavljati kompetenciju za obavljanje određenih zadataka, a može predstavljati autoritet i odgovornost. Ima mogućnost da održava određene zadatke dužnosti koji se rotiraju kroz više korisnika. Uloge se razlikuju od grupa jer se tretiraju kao skup dozvola, a ne kao skup korisnika. One u stvari predstavljaju skup korisnika s jedne strane i skup dozvola sa druge, odnosno služe kao posrednik za spajanje ova dva skupa što je prikazano na slici 2 [7].



Slika 2. *Korisnici, uloge i dozvole [8]*

Dvostruke strelice prikazane na slici 2 ukazuju na vezu više prema više. To znači da jedan korisnik može biti u asocijaciji sa jednom ili više uloga, i isto tako jedna uloga može biti u asocijaciji sa više jednim ili više korisnika. Isti scenario se odvija i između operacija i uloga [8].

Kako bi se definisala uloga, potrebno je izvršiti analizu ciljeva i strukture organizacije što je obično povezano sa bezbednosnim normama. Koristeći *RBAC* metod kontrole pristupa administratori veb aplikacije imaju mogućnost određivanja operacija nad podacima, odnosno ko može da vrši koju akciju, kada, kojim redosledom i pod kojim relacionim okolnostima. Dakle operacije koje su povezane s ulogama ograničavaju članove uloge na određeni skup akcija. Na taj način korisnici sistema imaju mogućnost da pristupe samo onim informacijama koje su im potrebne kako bi izvršili svoj posao. Iako *RBAC* model ne promoviše ni jednu politiku zaštite, pokazalo se da podržava nekolicinu poznatih principa i politika koji su važni za bezbednost komercijalnih i državnih preduzeća koji rade sa osetljivim informacijama [8].

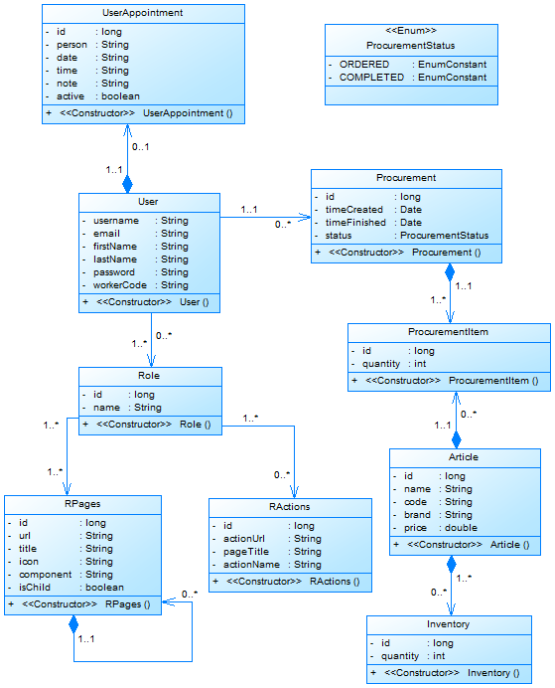
3. MODEL SISTEMA KONTROLE PRISTUPA

Prilikom kreiranja uloga sistema statički su im dodeljene odgovarajuće dozvole odnosno akcije. Kako se radi o front-end delu aplikacije, u pitanju su komponente (stranice) i različite akcije i pogledi na stranicama. Te stranice i akcije su sačuvane u bazi podataka kao liste objekata *RPages* i *RActions*, i u bazu se upisuju nakon što se takva komponenta implementira na klijentskoj strani. Dodavanjem korisnika u sistem dodeljuje mu se i odgovarajuća uloga, ili više njih. Korisniku se prikazuju samo komponente i akcije za koje mu je dozvoljen pristup, dok se ostale komponente ne generišu. To funkcioniše tako što se, nakon što se korisnik uspešno prijavi na sistem, sa back-end dela aplikacije na osnovu njegove uloge u vidu liste dobave sve stranice i akcije koje su mu dozvoljene. Postoji komponenta koja primi ovu listu te na osnovu nje generiše ostatak aplikacije. Na taj način korisnik ima pristup samo onim operacijama koje su dozvoljene u okviru njegove uloge te ne može vršiti neautorizovane operacije niti videti neautorizovani sadržaj.

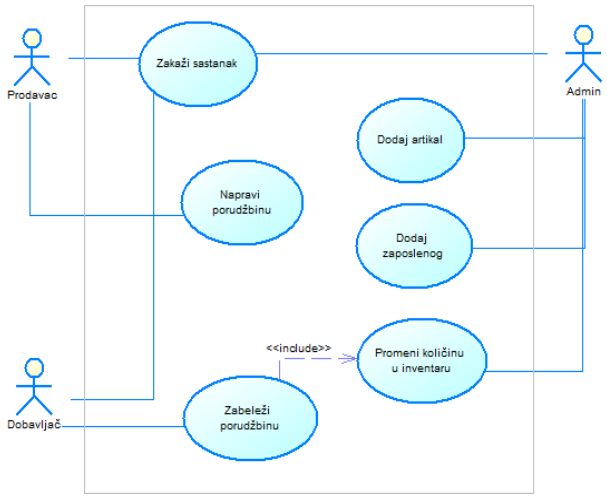
Sistem na kom je verifikovano korišćene kontrole pristupa posredstvom uloga, predstavlja veb aplikaciju magacina za skladištenje robe. Magacin sadrži skup artikala, tj. inventar, koji se trenutno nalazi u njemu i aplikacija služi kako bi se izvršila potražnja artikala koji nedostaju kao i izvršila evidencija artikala koji su dopremljeni. Na slici 3 prikazan je dijagram klasa sistema. Unutar organizacije magacina postoje 3 glavne vrste zaposlenih i to su:

- Prodavac – osoba koja potražuje robu
- Dobavljač – osoba koja evidentira robu koja je pristigla
- Admin – osoba koja dodaje nove zaposlene i ima kontrolu nad sistemom

Sistem podržava dodeljivanje više uloga jednom korisniku, s toga radnik koji ima ulogu menadžera, ima i ulogu prodavca i ulogu dobavljača. Na slici 4 prikazan je dijagram slučeva korišćenja za prethodno navedene tri uloge.



Slika 3. *Dijagram klasa sistema*



Slika 4. *Dijagram slučajeva korišćenja*

4. IMPLEMENTACIJA SISTEMA

U narednom odeljku biće predstavljena veb aplikacija za magacinsko poslovanje i način generisanja komponenti.

4.1. Implementacija aplikacije

Da bi korisnik mogao da pristupi stranicama i podacima, prvo je potrebno da se prijavi na sistem. Neprijavljeni korisnik ima pristup samo *Login* komponenti, i ukoliko pokuša da pristupi nekoj drugoj komponenti putem address bar-a prikazuje se *NotFound* komponenta.

Nakon što je korisnik uneo svoje kredencijalne u formu za prijavu, podaci se šalju back-end sistemu na proveru. Ukoliko su kredencijali tačni, server generiše *JSON Web Token* (*JWT*) koji se vraća klijentu kao odgovor i pomoću kog će se klijent identifikovati serveru ubuduće u toku sesije dok se korisnik ne odjavi sa sistema, ili dok token ne istekne. Ukoliko token istekne, korisnik će ponovo morati da se prijavi na sistem. Kada se korisnik uspešno autentifikuje, biva preusmeren na *PrivateRoutes* komponentu, prikazanu na listingu 1, koja generiše sve dozvoljene stranice i akcije u sistemu. Prilikom generisanja ove komponente, vrši se dodatna provera da li je korisnik prijavljen na sistem kako bi se još više osigurali pristup komponentama.

```
function PrivateRoutes() {
 const match = useRouteMatch('/app');
 const [allowedRoutes, setAllowedRoutes] = useState([]);
 const [hasRoutes, setHasRoutes] = useState(false);

 useEffect(() => {
  UserService.isUserLoggedIn().then(response => {
    let isit = response.data
    if(isit) {
     UserService.getUserRoutes().then(response => {
      if(response != null) {
       setAllowedRoutes(response.data)
       setHasRoutes(true);
      }
     })
    } else {
     return <Redirect to="/" />;
    }
   })
 }, [])

 const handleChild = (parent, route, url) => {
  const Component = Routes[route.component];
  return route.children.length > 0 ?
  route.children.map((child) => {
   const newUrl = url + child.url;
   return handleChild(route, child, newUrl)
  })
   :
  <Route
   exact
   key={route.url}
   path={`${match.path}${url}`}
  >
   <Component allowedActions={route.actions}/>
  </Route>
 }

 if({hasRoutes}) {
  return (<Fragment>
   <Navigation routes={allowedRoutes} path={match.path} />
   <Switch>
    {allowedRoutes.map((route) => {
     if(route.children.length > 0) {
      return (route.children.map( child => {
       var newUrl = route.url + child.url;
       return handleChild(route, child, newUrl)
      }))
     }
     const Component = Routes[route.component];
     return (
      <Route
       exact
       key={route.url}
       path={`${match.path}${route.url}`}
      >
       <Component allowedActions={route.actions}/>
      </Route>
     )
    })}
    <Route component={NotFound} />
   </Switch>
  </Fragment>
  )
 } else {
  return (
   <div style={{textAlign: 'center'}}>
    <h1 style={{marginTop: '15%'}}>Loading...</h1>
   </div>
  )
 }
}
```

Listing 1. *PrivateRoutes komponenta*

Pre samog prikaza *PrivateRoutes* komponente, šalje se zahtev *back-end* delu sistema kako bi se pribavile rute odnosno stranice kojima korisnik može pristupiti. Pored stranica, dobavljaju se i akcije koje korisnik može izvršiti. Akcije mogu predstavljati i delove stranice koje će se prikazati korisniku, kao i operacije koje kornisnik može izvršiti (npr. pravljenje nove porudžbine).

Kada na *front-end* kao odgovor stigne lista korisničkih ruta, prikazuju se komponente. Najpre se postavi navigaciona traka (eng. navbar) koja sadrži linkove pomoću kojih korisnik navigira između stranica. Prilikom generisanja komponente *Navigation*, prolazi se kroz svaku rutu i proverava da li je lista dece prazna. Ukoliko jeste, ruta se generiše kao link, dok ukoliko nije, ruta se generiše kao padajući meni (eng. dropdown menu). Nakon što se navigaciona traka generiše, generišu se komponente na koje linkovi navigacije vode.
One se generišu tako što se preuzme naziv komponente iz rute, nakon čega se preuzme komponenta sa istim nazivom koja se izvozi iz komponente *ComponentRoutes*. U ovoj komponenti nalaze se uvezene i potom izvezene sve komponente koje se prikazuju u sistemu kako bi se na osnovu naziva iz rute, mogla pronaći odgovarajuća komponenta sistema. Važan parametar prilikom kreiranja komponente jeste parametar *allowedActions* pomoću kojeg se komponenti koja se generiše prosleđuju akcije koje su dozvoljene korisniku.
Kada je generisanje navigacione trake i svih komponenti završeno, korisniku se prikazuje aplikacija i omogućeno mu je korišćenje. Ukoliko korisnik navigira na neku komponentu, tj. stranicu sistema, potrebno je prilikom prikazivanja proveriti za svaku akciju na toj stranici koja zahteva dozvolu da li je korisnik ima. To se radi tako što sto se uz deo koda koji zahteva proveru postavlja i uslov, koji ukoliko je tačan, prikazuje ackiju. Taj uslov se proverava pomoćnom funkcijom *isActionAllowed* sa listinga 2.

```
isActionAllowed = (actionName = "") => {


let numbb = this.props.allowedActions.map(function(a) { retu
rn a.actionUrl; }).indexOf(actionName);
  return (numbb >= 0) ? true : false;
 }
```

Listing 2. *Pomoćna funkcija isActionAllowed*

4.2. Dodavanje nove komponente ili dozvole u sistem

Kako bi se dodala nova komponenta u sistem, potrebno je nakon kreiranja same komponente uvesti je u komponentu *ComponentRoutes* a potom i dodati u listu izvezenih komponenti. Pri tome treba voditi računa da se komponenta izvozi sa istim nazivom kao što je ona nazvana u sistemu. Ovaj proces potrebno je uraditi na *front-end* strani sistema dok je na *back-end* strani potrebno dodati komponentu u bazu podataka. Na isti način se u bazu podataka dodaju i akcije koje se nalaze na novonastaloj komponenti.
Ukoliko se ulozi u sistemu dodaje dozvola za pristup određenoj komponenti ili akciji, potrebno je dodati željenu komponentu/akciju u listu već postojećih komponenti/akcija koje su u vezi sa ulogom.

5. ZAKLJUČAK

Tema ovog rada jeste kreiranje sistema za dinamičko generisanje komponenti na klijentskoj strani veb aplikacije, implementirane korišćenjem React radnog okvira. Predstavljen je sistem koji na osnovu kontrole pristupa zasnovanoj na ulogama (*RBAC*) vrši aktivaciju i deaktivaciju odgovarajućih komponenti sistema.
Objašnjeni su neki od osnovnih koncepata veb aplikacija, autentifikacije, kontrole pristupa, kao i kontrole pristupa zasnovanoj na ulogama. U trećem poglavlju prikazan je model celokupnog sistema, zajedno sa objašnjenjem rada sistema propraćenim dijagramom klasa. Sama implementacija prikazana je u poglavlju četiri. Sistem na kom je verifikovana kontrola pristupa zasnovama na ulogama predstavlja veb aplikaciju magacina gde korisnici sa odgovarajućom ulogom imaju dozvolu da pristupe različitim stranicama i obavljaju različite zadatke. Pored implementacije, prizakane su fotografije sistema za vreme korišćenja.

Dalji pravci razvoja i unapređenja veb aplikacije može biti implementacija *HTTPS*-a (*Hypertext Transfer Protocol Secure*) kako bi se osigurala sigurna komunikacija između klijenta i servera. Pored toga, moguće je sistem implementirati korišćenjem React Redux-a koji predstavlja mesto za čuvanje stanja unutar cele aplikacije, te bi se korisničke rute i dozvole mogle tu sačuvati, a ne prosleđivati među komponentama. Na kraju, moguće je implementirati funkcionalnost i grafički korisnički interfejs za promenu dozvola među ulogama, kojoj bi imao pristup admin sistema. Na taj način nakon kodiranja klijentskog i serverskog dela, odnosno dodavanje nove komponente u sistem, moguće je prepustiti adminu sistema da odredi koja uloga će imati dozvolu da pristupi novonastaloj komponenti, kao i da promeni prethodno postavljene veze uloga, komponenta i akcija.

6. LITERATURA

[1] Sabah Al-Fedaghi, *Developing Web Applications*, Computer Engineering Department - Kuwait University
[2] Marin Šili, Jakov Krolo, Goran Dela, *Security Vulnerabilities in Modern Web Browser Architecture*, Faculty of Electrical Engineering and Computing, University of Zagreb, 2010.
[3] Vipul A M, Prathamesh Sonpatki, *ReactJS by Example – Building Modern Web Applications with React*, Packt Publishing, 2016.
[4] Mark Piispanen, *Modern architecture for large web applications*, Bachelor's Thesis in Information Technology, 2017. [Online]. Доступно: https://jyx.jyu.fi/bitstream/handle/123456789/54129/URN%3aNBN%3afi%3ajyu-201705272524.pdf?sequence=1&isAllowed=y
[5] Muzafer Saračević, Muhedin Hadžić, *Upravljanje bezbednošću u cyber prostoru i mehanizmi zaštite veb apliakcija*, Septembar 2019.
[6] Stevan Džigurski, Bezbednost veb aplikacija, Novi Sad, 2003.
[7] Ravi S. Sandhu, *Role-based Access Control. Advances in Computers*, Laboratory for Information Security Technology, Academic Press 1998.
[8] David F. Ferraiolo, Janet A. Cugini, Richard Kuhn, *Role-Based Access Control (RBAC): Features and Motivations*, National Institute of Standards and Technology, 2003.

Kratka biografija:

Vukašin Jović rođen je u Novom Sadu 06.09.1996. godine. Osnovne akademske studije upisao je na Fakultetu Tehničkih nauka Univerziteta u Novom Sadu 2015. godine. Diplomirao je 2019. godine nakon čega upisuje master akademske studije na istom fakultetu.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.1:766
DOI: https://doi.org/10.24867/12BE05Stanarevic

ANGULAR КОМПОНЕНТА ЗА КОНТРОЛУ ПРИСТУПА БАЗИРАНУ НА КОРИСНИЧКИМ УЛОГАМА

ANGULAR COMPONENT FOR ROLE-BASED ACCESS CONTROL

Јелена Станаревић, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *У раду је описана Angular компонента која омогућује динамичку ауторизацију на клијентској страни апликације. Ауторизација, односно, контрола приступа базирана је на RBAC (Role-based Access Control) моделу. Angular компонента представља проширење постојећих могућности Angular програмског оквира и омогућава уклањање/приказивање компоненти и делова компоненти, као и дозволу/забрану измене или уноса одређених поља у оквиру компоненти. Верификација креиране Angular компоненте приказана је кроз информациони систем за физикалну терапију.*

Кључне речи: *Kontrola pristupa, RBAC, Angular framework, dinamička autorizacija*

Abstract – *This paper describes the Angular component that is responsible for dynamic authorization on the client side of the application. Authorization (access control) is based on the RBAC (Role-based Access Control) model. The Angular component is an extension of the existing capabilities of the Angular framework and allows the removal / display of components and parts of components, as well as the permission / prohibition of modification or entry of certain fields within the components. Verification of the created Angular component is presented through the information system for physical therapy.*

Keywords: *Access control, RBAC, Angular framework, dynamic authorization*

1. УВОД

Често се *Role-based Access Control* модел контроле приступа имплементира на серверској страни апликације. Међутим, овакав приступ има великуману, а то је брзина којом се проверава да ли корисник има или нема право приступа делу система. Приступ имплементиран у оквиру система за физикалну терапију, јесте динамичка ауторизација на клијентској страни апликације. Претходно споменута брзина провере права приступа корисника се знатно повећава са овим приступом.

У оквиру овог рада биће описана и презентована посебно креирана Angular компонента која је одговорна за примену контроле приступа базиране на корисничким улогама (*Role-based Access Control*, *RBAC*), односно, за динамичко генерисање различитих компоненти и делова компоненти у складу са правима приступа.

2. КОНТРОЛА ПРИСТУПА У ИНФОРМАЦИОНИМ СИСТЕМИМА

У области информационе безбедности (енгл. Information Security) контрола приступа подразумева праксу спречавања неовлашћеног приступа, коришћења, откривања, модификације, снимања или уништавања информација [1].

2.1. Role-based access control

Концепт контроле приступа заснован на улогама (енгл. Role-based access control, RBAC) јавио се са увођењем вишекорисничких *online* система и апликација око 1970. године. Централно место у оквиру RBAC модела заузимају појмови и односи улога, дозвола и корисника. Дозволе су везане за улоге, док се корисницима те улоге додељују.

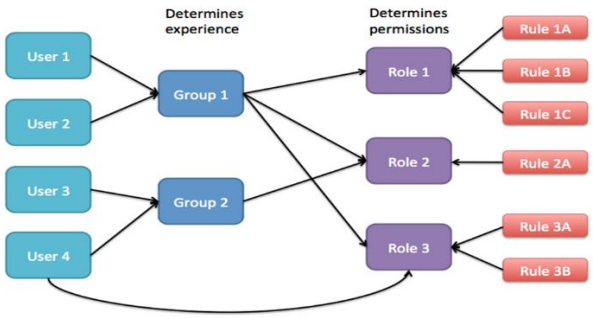
Улоге се креирају у складу са различитим корисничким функцијама у организацији или компанији. Дакле, корисницима се додељују улоге у складу са одговорностима и надлежностима које имају у оквиру компаније. С обзиром да су промене у оквиру било ког информационог система неизбежне, корисницима је могуће додељивати друге улоге у току времена.

Такође, улогама могу да се уклањају претходне и додају нове дозволе [1]. У оквиру организације улоге се ретко мењају, док се дозволе и корисници чешће могу мењати. Однос између улога, корисника и дозвола приказан је на слици 2.1.

Основна предност RBAC модела јесте управо централизована администрација улога, дозвола и корисника.

Централизована администрација обезбеђује једноставније управљање у смислу да се све своди на додељивање одговарајућих улога корисницима, за разлику од ACL листи (енгл. Access Control Lists). У оквиру ових листи неопходно је да се за сваког новог корисника прође кроз сваки ресурс система и његовој листи придружи корисник са одговарајућим правом које има над тим ресурсом [1].

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био проф. др Горан Сладић.



Слика 2.1. *Однос корисника, група, улога и дозвола* [1]

3. СПЕЦИФИКАЦИЈА КОМПОНЕНТЕ ЗА КОНТРОЛУ ПРИСТУПА

У оквиру овог поглавља биће презентована улога компоненте за контролу приступа, али без детаља око саме имплементације.

3.1. Постојеће могућности Angular framework-а

Angular представља платформу и програмски оквир (енгл. framework) за израду клијентске стране интернет апликација. Програмски оквир је направљен у JavaScript-у и омогућава израду тзв. *single-page* апликација користећи TypeScript и HTML. Angular нуди још једну могућност динамичке измене структуре HTML фајла одређене компоненте уз помоћ тзв. структуралних директива. Структуралне директиве, попут **ngIf* или **ngFor* омогућавају да се мења изглед *Document Object Model* стабла дефинисаног у оквиру HTML странице у зависности од вредности појединих података доступних конкретној компоненти [2]. **ngIf* директива представља уграђену директиву програмског оквира, која уклања или креира део *Document Object Model* стабла. Претходно наведено зависности од резултата валидације израза на који се односи (слика 3.1) [2].

```
<div *ngIf="condition">Content to render when condition is true.
</div>
```

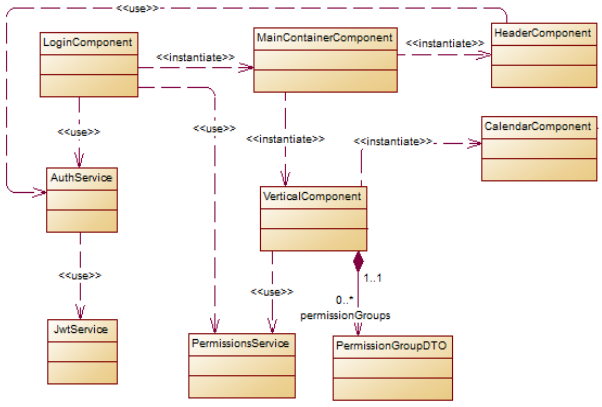
Слика 3.1 – *Пример примене *ngIf директиве*

3.2. Модел компоненте за контролу приступа

На сликама у наставку биће приказан део UML дијаграма класа за клијентску страну апликације. Конкретно, приказане су постојеће компоненте у оквиру Angular програмског овкира, сервисне и модел класе, као и везе између њих. Комуникација ових компоненти је кључна у реализацији динамичке ауторизације.

На слици 3.2. приказан је део UML дијаграма класа који се односи на компоненте клијентске стране које се инстанцирају приликом пријаве на систем, везе између њих, као и везе компоненти са сервисним класама. *LoginComponent* компонента, путем *dependency injection* концепта, садржи везе ка уграђеним компонентама Angular програмског оквира [2].

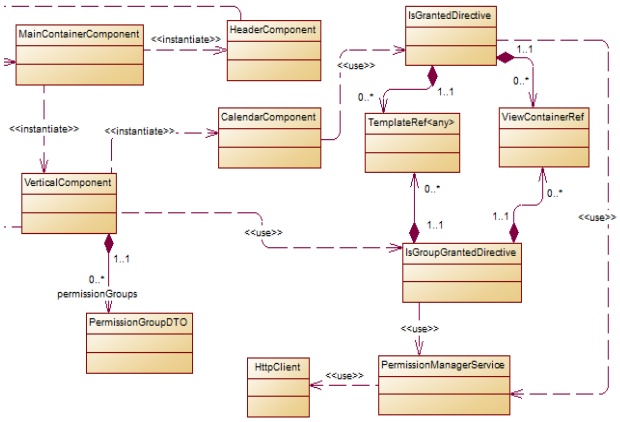
Такође, садржи везе према сервисима задуженим првенствено за управљање процесом пријаве корисника на систем (*AuthService* и *PermissionsService*). *AuthService* у себи садржи везу ка *JwtService* сервис класи. Везе које садржи *LoginComponent* компонента омогућавају јој да користи све методе претходно набројаних сервиса. *JwtService* сервис класа омогућава управљање JWT токенима. У оквиру система за физикалну терапију имплементирана је аутентификација учесника система уз помоћ JSON Web Токена или, скраћено, JWT [3].



Слика 3.2 – *Део UML дијаграма класа који се односи на компоненте одговорне за пријаву корисника на систем*

На слици 3.3 приказан је део UML дијаграма класа који се односи на везе између компоненти приказа и директива које омогућавају приказ/уклањање компоненти и делова компоненти. Директиве задужене за динамичку ауторизацију у систему су *IsGrantedDirective* и *IsGroupGrantedDirective*.

Обе су путем *dependency injection* концепта повезане са *PermissionManagerService* сервис класом, која, преко комуникације са *PermissionsService* сервис класом, омогућава преузимање и проверавање дозвола које улоговани корисник поседује. *CalendarComponent* компонента такође користи *IsGrantedDirective* директиву и има везу ка њој.



Слика 3.3 – *Део UML дијаграма класа који се односи на везу компоненти са директивом*

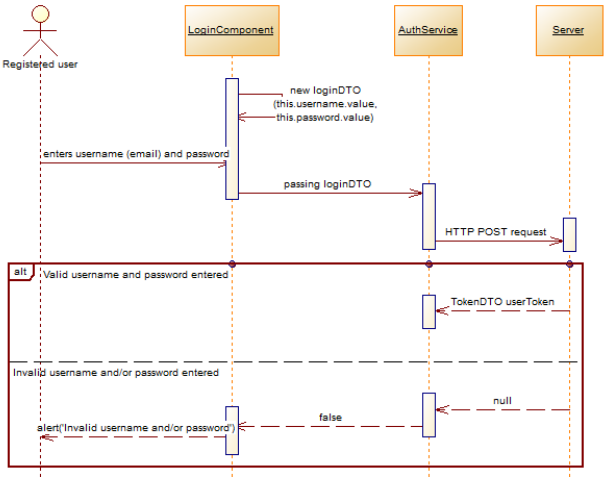
IsGrantedDirective и *IsGroupGrantedDirective*, преко инстанци *TemplateRef<any>* и *ViewContainerRef*, омогућавају да се део HTML садржаја креира, обрише, или да се забрани, односно, омогући измена тог садржаја.

3.3. Интеракција компоненте за контролу приступа са компонентама система

На сликама у наставку биће презентован целокупан процес и логика иза креирања/уклањања компоненти и делова компоненти, као и омогућавања/забране уноса и измене појединих поља у складу са скупом дозвола пријављеног корисника. На слици 3.4 приказана је комуникација компоненти и сервиса приликом пријаве корисника на систем. Регистровани корисник уноси емаил и лозинку у одговарајућа поља *LoginComponent* компоненте, која од датих информација формира објекат типа *LoginDTO* и прослеђује формирани објекат *AuthService* сервис класи [4].

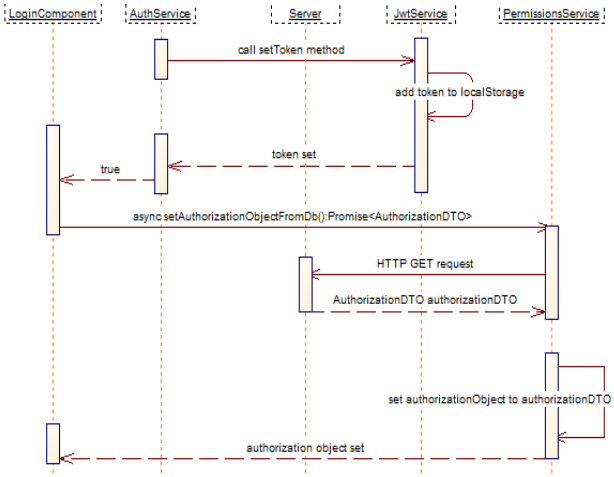
Ова класа комуницира са серверском страном апликације, ради верификовања исправности унетих креденцијала корисника. Серверска страна је у дијаграмима секвенци приказана по тзв. *Black-Box* принципу. Овај принцип подразумева да се одређени систем или део система представи само преко улазних и излазних параметара, без самих детаља имплементације и начина функционисања истог [5]. Комуникација са серверској страном обавља се путем HTTP POST захтева.

Након провере послатих креденцијала, серверска страна има два алтернативе. Једна подразумева да унети емаил и/или лозинка нису исправни и у том случају сервер враћа null вредност и клијентска страна обавештава корисника о неисправности креденцијала. Друга алтернатива подразумева да сервер одговори са формираним JWT токеном, као потврдом успешне пријаве корисника на систем [3].



Слика 3.4 – *пријава корисника на систем*

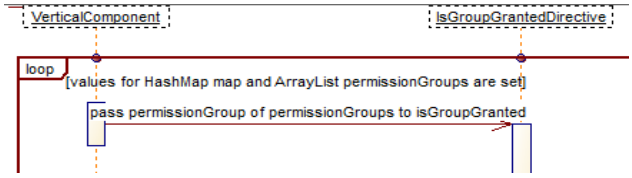
Преузети JWT токен се шаље од *AuthService* сервис класе до *JwtService* сервис класе која смешта тај токен у *locale storage* интернет претраживача. Информација о успешно ускладиштеном JWT токену враћа се до *LoginComponent* компоненте. Даље следи иницијализација сервисних класа и компоненти почетне странице апликације [3]. *PermissionsService* сервис класа добавља објекат ауторизације.
Наведени објекат садржи информације о свим улогама, дозволама и групама дозвола корисника путем комуникације са серверском страном (слика 3.5).



Слика 3.5 *Сценарио успешне пријаве корисника на систем*

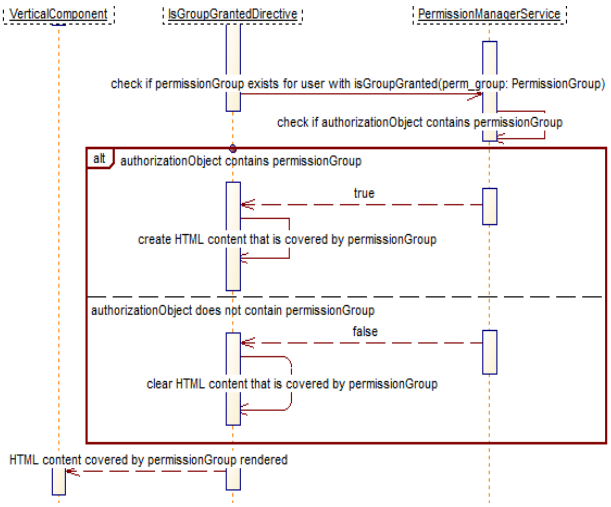
Иницијализација *VerticalComponent* компоненте, која репрезентује вертикални мени почетне странице и попуњава се линковима за извршавање појединих операција (додавање новог пацијента, измена одређеног термина, итд.) врши се путем комуникације са *IsGratedDirective* и *IsGroupGrantedDirective* директивама. Попуњавање компоненте линковима зависи од скупа дозвола пријављеног корисника. Најпре се кроз комуникацију са *PermissionsService* сервис класом (последично и са серверском страном апликације), добављају одвојено листа свих група дозвола и мапа груписаних свих дозвола у систему.

С обзиром да су дозволе распоређене по групама у облику мапе у компоненти *VerticalComponent*, приликом њене иницијализације најпре се проверава да ли корисник има право да приступа одређеној групи дозвола. Та провера врши се преузимањем по једне из листе свих група и комуникацијом са *IsGroupGrantedDirective* директивом. Споменута директива се обраћа *PermissionManagerService* сервис класи која ће, на основу претходно преузетог *authorizationDTO* објекта од *PermissionsService* класе (види слику 3.5) вратити директиви информацију о томе да ли корисник поседује прослеђену групу дозвола [3]. У случају да корисник поседује прослеђену групу дозвола, HTML садржај се креира, док се у супротном брише у оквиру директиве (слике 3.6 и 3.7).



Слика 3.6 – *Слање по једне групе дозвола*

Пре него што се провери да ли корисник поседује наредну групу, постојећа група која је анализирана се користи као кључ у мапи у оквиру *VerticalComponent* компоненте. На основу кључа преузимају се и проверавају дозволе везане за исти.



Слика 3.7 – *Proveravanja prisustva prosleđene grupe dozvola*

У овој ситуацији *VerticalComponent* компонента комуницира са *IsGrantedDirective* директивом, прослеђујући јој дозволу по дозволу. Директива у комуникацији са *PermissionManagerService* сервис класом проверава присуство сваке дозволе у скупу дозвола пријављеног корисника. У складу с тим *PermissionManagerService* сервис класа враћа вредности *true* или *false* уколико корисник поседује, односно, не поседује дозволу у оквиру свог скупа дозвола. *IsGrantedDirective* директива ће у складу са тим креирати или уклонити одређени HTML садржај.

4. ИМПЛЕМЕНТАЦИЈА КОМПОНЕНТЕ ЗА КОНТРОЛУ ПРИСТУПА

4.1. Провера права приступа ресурсу система на клијентској страни

Константно пресретање захтева и провера скупа дозвола које корисник са одређеном улогом има над сваким ресурсом представља безбедан и поуздан начин имплементације ауторизације. Међутим, овај начин имплементације је знатно спорији од алтернативе која ће бити презентована у овом одељку и која постоји у систему. Алтернатива подразумева динамичку ауторизацију на клијентској страни апликације, односно, проверу скупа дозвола над ресурсима система без учешћа пресретача (енгл. Interceptor) на серверској страни.

4.1.1 Динамичка ауторизација у систему за физикалну терапију

У систему креирана је нова директива као проширење постојећих, која омогућава динамичко проверавање дозвола пријављеног корисника и приказ или уклањање компоненти или делова компоненти, као и забрану/дозволу измене појединих поља. Директива је означена са *@Decorater({selector: '[appIsGranted]'})*. Као и код структуралних директива, примена кода нове директиве подразумева да се HTML садржај обухвати са селектором директиве, односно, у овом случају – са **appIsGranted*. Кључне елементе у оквиру директиве представљају *ViewContainterRef* и *TemplateRef*, који су уједно и поља у оквиру исте. *ViewContainterRef* представља контејнер у који се може додати прикази дефинисани *TemplateRef* шаблоном.

Тих приказа може бити један или више. Овај шаблон садржи део *Document Object Model* стабла у оквиру HTML фајла конкретне компоненте. Тај део ће бити генерисан, елиминисан или ће се омогућити/ онемогућити унос и измена истог у зависности од скупа дозвола пријављеног корисника.

6. ЗАКЉУЧАК

У раду су описани основни појмови везани за контролу приступа, са посебним нагласком на *Role-based Access Control* или *RBAC* модел контроле приступа. Овај модел описан је у посебном одељку. Спецификација Angular компоненте, односно, директиве, одговорне за динамичку ауторизацију, налази се у оквиру одвојеног поглавља. У оквиру истог поглавља презентоване су постојеће могућности Angular програмског оквира у погледу динамичког генерисања компоненти и делова компоненти, затим презентовање компоненте путем дела UML дијаграма класа и одређених UML дијаграма секвенци.

Кроз пример генерисања вертикалног менија објашњена је примена креиране директиве и начин на који компоненте, сервиси и директиве комуницирају у циљу реализације динамичке провере дозвола корисника у систему. Потенцијална проширења представљеног решења би могла бити могућност креирања нових дозвола у систему и њихово додељивање постојећим или потпуно новим улогама. Затим могућност да се привремено модификује постојећи скуп дозвола (нпр. да се поједине дозволе ипак онемогуће уколико се у оквиру система дода потпуно нови корисник, без искуства у раду са системом, са улогом администратора).

7. ЛИТЕРАТУРА

[1] Role-based Access Control, Ravi S. Sandhu, http://www.profsandhu.com/articles/advcom/adv_comp_rbac.pdf
[2] Izrada web aplikacije putem Angulara 6 razvojnog okvira, F Tuđan, https://repozitorij.unin.hr/islandora/object/unin:2023/datastream/PDF/download
[3] Security of JSON Web Tokens (JWT), https://cyberpolygon.com/materials/security-of-json-web-tokens-jwt/
[4] Create Data Transfer Objects (DTOs), https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5
[5] Black-Box Model, https://www.sciencedirect.com/topics/engineering/black-box-model

Кратка биографија:

Јелена Станаревић рођена је 28.07.1995. године у Новом Саду. Завршила је Основну школу ”Васа Стајић”, а потом и Гимназију ”Јован Јовановић Змај” (природно-математички смер), у Новом Саду. Гимназију је завршила 2014., и исте године је уписала Факултет техничких наука у Новом Саду, одсек Рачунарство и аутоматика. Завршила је основне академске студије 27.09.2018.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.39(043.2)
DOI: https://doi.org/10.24867/12BE06Cvijic

GOOSE KOMUNIKACIJA

GOOSE COMMUNICATION

Dajana Cvijić, Zoran Stojanović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu su prikazani osnovi principi modelovanja u okviru IEC 61850 protokola i GOOSE komunikacija kao poseban dio samog protokola. Osim samog modelovanja podataka, prikazana je i GOOSE komunikacija između zaštitnih uređaja - releja. Opisana je i ispitana jedna primjena GOOSE komunikacije, koja se odnosi na blokadu zaštitnih funkcija releja, čime je ujedno dokazana njena funkcionalnost i prednosti ovakvog vida komunikacije.*

Ključne reči: *IEC 61850, GOOSE komunikacija*

Abstract – *In this paper modelling within IEC 61850 protocol and GOOSE communication as special part of this protocol is presented. Except modelling, GOOSE communication between protective devices – relays is shown. One application, related to blocking of protection functions, of GOOSE communication is desribed and tested. In this way, functionallity of GOOSE communication and its advantages are proven.*

Keywords: *IEC 61850, GOOSE communication*

1. UVOD

Uvođenjem mikroprocesorskih i multifunkcionalnih zaštitnih uređaja – releja počelo je digitalno doba elektroenergetskih sistema. Na samom početku, prikupljanje informacija bilo je "offline", odnosno nije bilo moguće uživo pratiti status rasklopne opreme i mjerenja. Međutim, unaprijeđenjem inteligentnih elektronskih uređaja IED-a (IED – Intelligent Electronic Device) i njihovih komunikacionih komponenti, omogućeno je uživo ("real-time") praćenje i upravljanje elektroenergetskih postrojenja. Ranih devedesetih godina, razni proizvođači su predstavili IED-e sa integrisanim komunikacionim protokolima.

Velike razlike u načinu realizacije komunikacionih protokola među proizvođačima izazvale su pojavu novog problema, gdje je bilo teško ostvariti međusobnu komunikaciju ovih uređaja. Da bi se ovaj problem riješio, 2004. godine je nastao međunarodni standard danas poznat kao IEC 61850. Osim što je postao standardni komunikacioni protokol u elektroenergetici, IEC 61850 je postao i standard za dizajn i implementaciju upravljačkog sistema (*SAS – Substation Automation System*) u elektroenergetskim objektima.

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

U okviru IEC 61850 protokola postoji:

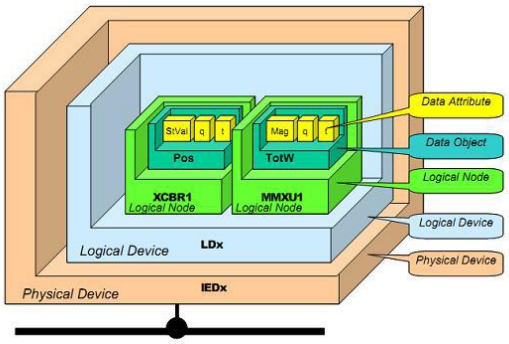
- vertikalna ili klijent-server komunikacija za koju se koristi protokol proizvođačkih specifikacionih poruka – MMS (*Manufacturing Message Specification*);
- horizontalna komunikacija za koju se koriste generički objektno orijentisani događaji GOOSE (*Generic Object Oriented Substation Event*), koja je ujedno i tema ovog rada.

U nastavku rada će biti opisana osnovna struktura IEC 61850 protokola, osnovni principi rada GOOSE komunikacije i njena primjena, kao i rezultati praktičnog rada. Praktični dio rada zasniva se na podešenju i ispitivanju GOOSE komunikacije između zaštitnih uređaja – releja.

2. IEC 61850 PROTOKOL

Protokoli, korišćeni prije nastanka IEC 61850 protokola, definisali su kako se bajtovi podataka prenose žicom. Međutim, ti protokoli nisu definisali organizaciju podataka u smislu primjene. Ovakav pristup zahtjevao je od inženjera da ručno podešavaju objekte i mapiraju ih na promjenljive vezane za elektroenergetski sistem, ulazno izlazne module, itd. IEC 61850 protokol je jedinstven. Osim što definiše način prenosa podataka žicom, on definiše i način (model) na koji se organizuju podaci vezani za uređaje u elektroenergetskim sistemima.

Na ovaj način, postignuta je velika ušteda vremena u smislu organizacije i pripreme podataka potrebnih za podešenje objekata sistema. Organizacija, odnosno modelovanje podataka, u okviru ovog protokola prikazano je na slici 1.



Slika 1. *Modelovanje podataka u okviru IEC 61850 protokola*

Komponente jednog podatka u okviru IEC 61850 protokola su:

- fizički uređaj,
- logički uređaj LD (*Logical Device*),
- logički čvor LN (*Logical Node*),
- podatak koji opisuje objekat DO (*Data Object*),
- atribut podatka DA (*Data Atributte*).

Fizički uređaj (PD – *Physical Device*) je uređaj koji se povezuje na mrežu. Fizički uređaj se opisuje sa njegovom IP adresom. U okviru jednog fizičkog uređaja može da postoji jedan ili više logičkih uređaja. Logički uređaj (LD – Logical Device) dozvoljava fizičkom uređaju da se ponaša kao "gateway" za više uređaja, pri čemu on postaje koncentrator podataka. IEC 61850 protokol ne definiše kako fizički uređaj treba da bude podijeljen u logičke uređaje. Logički uređaj grupiše logičke čvorove. Logički čvorovi (LN – *Logical Node*) su "folderi" koji sadrže podatke koji opisuju objekte DO (*Data Objects*). Obavezni logički čvorovi u okviru jednog logičkog uređaja su:

- LLN0, koji sadrži uobičajena pitanja (podatke) vezane za logički uređaj,
- LPHD, koji sadrži uobičajena pitanja (podatke) vezane za fizički uređaj.

Osim ova dva, mora postojati najmanje još jedan logički čvor. Logički čvorovi se sastoje od definisanog seta podataka koji opisuju objekte (DO-*Data Objects*). Podatak koji opisuje objekte je instanca (promjenljiva) vezana za klasu podataka (CDC-*Common Data Classes*). Atribut podatka (DA-Data Attribute) u sebi sadrži stvarne vrijednosti podatka. Svaki objekat koji opisuje podatak u sebi sadrži najmanje 3 atributa:

- vrijednost (npr. *stVal*),
- kvalitet (q),
- vrijeme (t).

Atributi su opisani sa funkcionalnim ograničenjima (*Functional Constraint*).

Jezik za opis podešenja transformatorske stanice SCL (SCL=Substation Configuration description Language) koristi se da opiše mogućnosti IED-a i omogućava razmjenu informacija između alata za podešenje IED-a različitih proizvođača. SCL je jezik zasnovan na XML-u. Postoje 4 različita tipa SCL:

- SSD (Substation Specification Descripition),
- SCD (Substation Configuration Description),
- ICD (IED Capability Description),
- CID (Configured IED Description).

Set podataka *(data set*) sadrži kolekciju referenci na objekte. On ne sadrži objekte koji opisuju podatke već reference na funkcionalno ograničene objekte ili funkcionalno ograničene atribute. Ove reference su članovi seta podataka i definišu podatke, koji treba da se prenesu kroz izvještaj. Članovi seta podataka, kao i njihov redoslijed, moraju biti poznati i klijentu i serveru. Set podataka može sadržati reference iz različitih logičkih čvorova.

Proizvodnja izvještaja i njihov prenos su kontrolisani pomoću kontrolnih blokova za izvještaje RCB (*Report Control Block*). RCB sadrži sve informacije neophodne za proizvodnju izvještaja i definisan je svojim atributima (dodijeljeni set podataka, opcije za aktivaciju, opciona polja). Dodijeljeni set podataka definiše koji podaci će se prenositi, a opcije za aktivaciju definišu kada će se podaci prenositi.

U GOOSE komunikaciji, slanje podataka zasniva se na setovima podataka i GOOSE upravljačkim blokovima. Set podataka definiše koji podaci se koriste i šalju kao GOOSE poruke. GOOSE upravljački blok GCB povezuje set podataka i njegove atribute sa stvarnim podacima.

3. GOOSE KOMUNIKACIJA

GOOSE predstavlja generičke objektno orijentisane događaje. Cilj GOOSE-a je da zamijeni konvencionalnu žičanu logiku za razmjenu informacija između releja sa razmjenom informacija pomoću komunikacionih protokola. GOOSE se koristi u automatizaciji transformatorskih stanica za brzu horizontalnu komunikaciju između zaštitnih releja. GOOSE se može koristiti za direktnu razmjenu podataka, npr. razmjena informacija o blokadnim uslovima između zaštitnih releja. Prema IEC 61850-8-1 standardu, GOOSE koristi „izdavač/pretplatnik" profil, gdje se informacije između uređaja razmjenjuju korišćenjem eternet „multicast" poruka. Poruka je „slika" poslatog IEC 61850 seta podataka, koji je definisan u konfiguraciji [1]. Podešenje horizontalne komunikacije sastoji se iz podešenja GOOSE upravljačkih blokova (GCB) i podešenja GOOSE ulaza. Kao rezultat dobija se standardni fajl u formatu SCL fajla, koji opisuje konfiguraciju sistema i koristi se za zaštitne releje.

Razmjena informacija pomoću GOOSE poruka uspostavlja se koristeći specifičan GSE (Generic Substation Event = Generički događaj u transformatorskoj stanici) model. Ovaj model omogućava brzu i pouzdanu razmjenu informacija (ulaznih i izlaznih vrijednosti podataka). GSE model predstavlja efikasan metod za simultan prenos podataka (istih generičkih informacija) prema više različitih fizičkih uređaja koristeći „multicast" servis. Prema standardu, GSE model odnosi se na razmjenu vrijednosti kolekcije atributa podataka. Standard definiše dva različita tipa poruka koja koriste GSE model. GSSE (Generic Substation State Event = Generički događaj u transformatorskoj stanici vezan za promjenu stanja) tip poruke ima mogućnost prenosa informacije o promjeni stanja, što znači parove bitova. GOOSE tip poruke može da prenese širok opseg atributa podataka organizovanih u set podataka. Osnovna razlika između između ova 2 tipa poruka je da GSSE obezbjeđuje jednostavnu listu statusa, dok GOOSE obezbjeđuje fleksibilnu kombinaciju informacija organizovanih u set podataka. Prema tome, za GOOSE komunikaciju moraju biti definisane informacije koje treba da se prenesu. Kao što je već spomenuto u prethodnom poglavlju, razmjena podataka zasniva se na „izdavač/pretplatnik" mehanizmu [2].

Praktično gledano, ovo znači da određen GOOSE upravljački blok (GCB) mora biti podešen za svaku GOOSE poruku. Ovaj upravljački blok sadrži informacije koje su potrebne za prenos seta podataka. GOOSE upravljački blok određuje MAC adresu (Media Access Control) za izvor i destinaciju GOOSE poruke. Odredišna adresa GOOSE poruke sadrži „multicast" MAC adresu, dok izvorišna adresa GOOSE poruke sadrži „unicast"

MAC adresu. Prva tri okteta su definisana IEEE standardom i to su 01-0C-CD za GOOSE i GSSE poruke. Četvrti oktet MAC adrese definiše tip poruke. Vrijednost 01 koristi se za GOOSE poruke. Konačno, poslednja 2 okteta koriste se za pojedinačno adresiranje različitih tipova poruka. Specifični trocifreni heksadecimalni VLAN (Virtual Local Area Network) identifikacioni broj (broj lokalne virtualne mreže) takođe je dio GCB-a u opsegu od 000 do FFF. Ovo se koristi za identifikaciju VLAN-a u kom GOOSE poruka treba da se prenese. Prioritet GOOSE poruke može biti određen specifičnim VLAN prioritetnim brojem, koji je decimalna vrijednost u opsegu od 1 do 7. Poruke sa prioritetom od 1 do 3 smatraju se porukama niskog prioriteta, dok se poruke sa prioritetom od 4 do 7 smatraju porukama visokog prioriteta. Još jedan dio GOOSE upravljačkog bloka je i identifikacioni broj aplikacije APPID (Application identity number). Ovo je jedinstveni heksadecimalni broj za slanje GCB-a u okviru mreže. On određuje set podataka i GOOSE poruku. APPID ima heksadecimalni opseg od 0000 do 3FFF. Da bi se osiguralo primanje GOOSE poruke, ona se šalje više puta u brzim intervalima [2].

Neke od prednosti GOOSE komunikacije su sledeće:

- značajna ušteda u prostoru zbog smanjenja signala koji treba da se prenose žičano,
- smanjenje troškova za instalaciju postrojenja,
- konstantan nadzor GOOSE komunikacije omogućava brzo pronalaženje i rješavanje problema u slučaju nestanka komunikacije,
- proširenje sistema za automatizaciju, npr. dodavanjem novih IED-a, ne zahtjeva dodatno žičenje između releja ili izmjene postojećih ožičenja,
- itd.

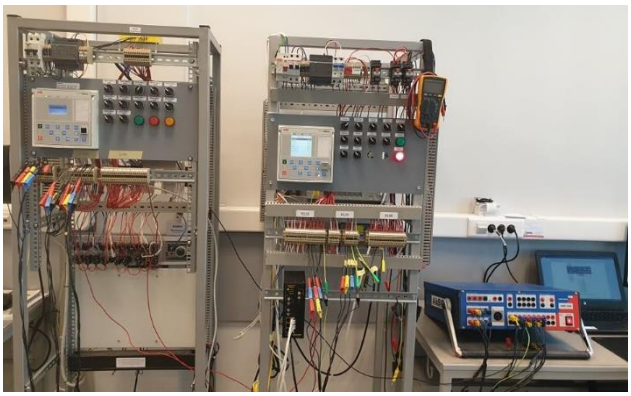
U zavisnosti od izvedbe postrojenja, postoje različite primjene GOOSE komunikacije:

- blokada manipulacije rasklopnom opremom,
- blokada zaštitnih funkcija,
- zaštita od otkaza prekidača,
- zaštita od električnog luka,
- aktiviranje zapisa o snimljenom poremećaju.

4. PODEŠENJE I ISPITIVANJE GOOSE KOMUNIKACIJE IZMEĐU RELEJA

Za ispitivanje GOOSE komunikacije u laboratorijskim uslovima korišćeni su ABB releji serije 615: RED615 i REF615, kao i OMICRON CMC356 test set za ispitivanje zaštitnih uređaja - releja. Izgled testnog okruženja prikazan je na slici 2.

Prvi korak u podešenju releja je podešenje trenutnih neusmjerenih prekostrujnih zaštita koje su u ovim relejima označene kao PHIPTOC1 zaštite. Obije zaštite podešene su tako da imaju jednake vrijednosti struje prorade i vremenska zatezanja. Kako se relej REF615 koristi kao relej koji služi za zaštitu izvodnih polja, a RED615 kao relej za zaštitu dovodnog polja, važno je da se spriječi djelovanje releja RED615 prije releja REF615 u slučaju kada oba releja vide jednake struje kvara, a kvar se nalazi na izvodnom polju releja REF615. U slučaju kada bi relej RED615 djelovao prije releja REF615, bez napajanja bi ostala sva izvodna polja iako je kvar detektovan na samo jednom polju.



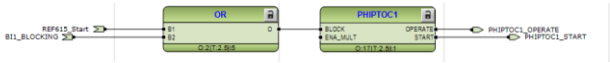
Slika 2. *Izgled testnog okruženja za ispitivanje GOOSE komunikacije*

Da bi se neželjeno djelovanje zaštite spriječilo, koristi se blokada trenutne neusmjerene prekostrujne zaštite, realizovana kao GOOSE poruka. Osim GOOSE poruke, blokada će biti realizovana i žičano preko jednog digitalnog ulaza, da bi se izvršila uporedna analiza žičanih signala i signala realizovanih kao GOOSE poruka.

Žičana blokada realizovana je direktnim povezivanjem digitalnog izlaza releja REF615, koji je prethodno softverski povezan sa signalom pobude trenutne neusmjerene prekostrujne zaštite, na digitalni ulaz releja RED615. Ovaj digitalni ulaz kasnije se vezuje na *BLOCK* ulaz funkcionalnog bloka zaštite, čime mu je dodjeljena funkcionalnost blokade.

Da bi se uvela i GOOSE blokada u funckionalni blok zaštite, potrebno je definisati GOOSE signale. Sva podešenja GOOSE funkcionalnosti rade se u PCM600 softveru. Prvi korak u definisanju GOOSE signala je kreiranje seta podataka, koji treba da sadrži informaciju o pobudi trenutne neusmjerene prekostrujne zaštite. Ovaj atribut se u okviru datog fizičkog uređaja može pronaći kao *LD0.PHIPTOC1.Str.general*. Osim ovog, setu podataka može se dodijeliti i signal prorade trenutne neusmjerene prekostrujne zaštite označen kao *LD0.PHIPTOC1.Op.general*, u cilju praćenja signala. Nakon što je set podataka kreiran, kreira se GOOSE upravljački blok sa parametrima definisanim u prethodnom poglavlju, a zatim se taj GCB dodjeljuje releju RED615. Nakon što su podaci dodjeljeni releju RED615, oni se mogu na odgovarajući način i iskoristiti. Da bi se GOOSE podaci mogli koristiti potrebno ih je "primiti" *GOOSERCV* blokom. U ovom slučaju koristi se *GOOSERCV_BIN* blok, koji služi za primanje digitalnih podataka, odnosno statusa. Kada su blokovi kreirani za svaki signal, oni se u okviru *Singal Matrix* alata povezuju sa prethodno definisanim signalima iz seta podataka. Na ovaj način je u potpunosti dodjeljena funkcionalnost GOOSE signalima.

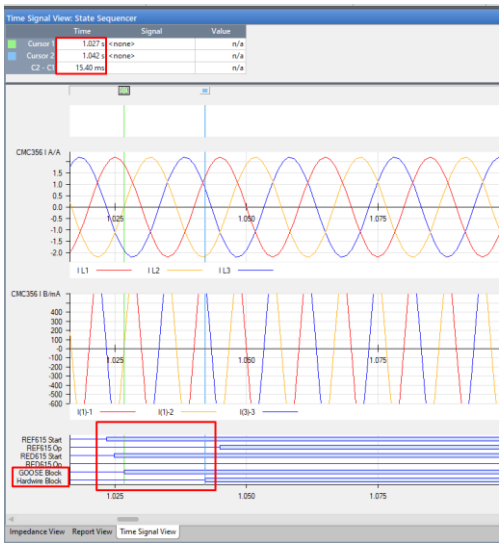
Ovako definisani signali sada se koriste dalje za blokadu zaštite (slika 3), povezivanje sa LED lampicama i na digitalne ulaze bloka *RDRE*, koji se odnosi na zapise o snimljenim poremećajima.



Slika 3. *Funkcionalni blok trenutne neusmjerene prekostrujne zaštite sa blokadama*

Za ispitivnje funkcionalnosti podešene GOOSE blokade korišćen je CMC356 test set sa odgovarajućim *State Sequencer* alatom. U okviru ovog alata podešena su tri stanja. U prvom stanju injektuju se nominalne struje za oba releja, u drugom stanju injektuju se struje kvara i poslednje stanje je ponovo stanje sa nominalnim strujama. Tokom stanja kvara posmatraju se vremenski signali koji se odnose na digitalne ulaze test seta CMC356. Na digitalne ulaze test seta povezani su signali pobude i prorade neusmjerene prekostrujne zaštite sa oba releja, signal GOOSE blokade i signal žičane blokade.

Na slici 4 prikazani su odzivi vremenskih signala tokom stanja kvara.



Slika 4. *Vremenski odzivi signala tokom stanja kvara*

Sa slike 4 uočava se da su oba releja dobila signal pobude zaštite približno u isto vrijeme, a 20 ms nakon pojave struje kvara. 2 ms nakon pojave signala pobude zaštite releja REF615 aktiviran je signal GOOSE blokade, što je rezultovalo time da nema prorade zaštite releja RED615. Prorada zaštite za relej REF615 desila se 40 ms nakon pojave struje kvara, što je u skladu sa podešenjem releja. Osim GOOSE blokade, prisutna je i žičana blokada, koja se pojavila 18ms nakon pojave signala pobude zaštite na releju REF615.

Da bi se izvršila uporedna analiza, kursorima su obilježeni trenuci pojave blokada i izračunata je vremenska razlika od 15.4 ms.

Ovim testom je ne samo dokazana funkcionalnost podešene GOOSE zaštite već i njena prednost u odnosu na žičanu blokadu.

5. ZAKLJUČAK

U ovom radu ukratko je objašnjen princip modelovanja podataka u okviru IEC61850 protokola,a zatim i GOOSE komunikacija kao poseban dio samog protokola.

Da bi se opisana funkcionalnost GOOSE komunikacije dokazala, u praktičnom dijelu rada podešena je i ispitana GOOSE komunikacija između dva zaštitna uređaja – releja, a na primjeru blokade zaštitnih funkcija.

Kako cilj rada nije samo dokazivanje funkcionalnosti već i prednosti GOOSE komunikacije, izvršena je uporedna analiza žičanog i GOOSE signala. Kako je za realizovanje žičane blokade neophodno korišćenje dodatnih digitalnih ulaza i izlaza, njihovo žičenje, ispitivanje i prostor koji zauzimaju predstavljaju značajnu manu u odnosu na GOOSE signal realizovan softverski. Osim toga, relej je GOOSE signal „primio“ 15.4 ms prije žičanog signala, u čemu se ogleda još jedna prednost ovog načina razmjene signala.

Uvođenjem IEC 61850 protokola, kao i GOOSE komunikacije, pojavljuju se značajne mogućnosti za unaprijeđenje IED-a i samih sistema automatskog upravljanja.

6. LITERATURA

[1] ABB, „615 Series IEC 61850 Engineering Guide“, https://library.e.abb.com/public/63c98d28e525f279c1257b2f0054c237/RE_615_IEC61850eng_756475_ENg.pdf (pristupljeno u septembru 2020.)

[2] Jukka Piirainen, „Applications of horizontal communication in industrial power networks“, Tampere University of Technology, 2010.

Kratka biografija:

Dajana Cvijić rođena je u Tesliću 1995. godine. 2013. godine je upisala Fakultet tehničih nauka, studijski program Energetika, elektronika i telekomunikacije. Na studijama se opredijelila za smjer Elektroenergetika – Elektroenergetski sistemi i diplomirala 2017. godine. Master studije upisala je 2017. godine.

Zoran N. Stojanović rođen je 22.07.1979. godine u Požarevcu. Doktorsku disertaciju pod nazivom "Usmereni releji bazirani na digitalnom faznom komparatoru" odbranio je 11.06.2012. godine na Elektrotehničkom fakultetu Univerziteta u Beogradu. Oblasti naučnoistraživačkog rada kojima se do sada bavio su relejna zaštita, razvodna postrojenja i monitoring i dijagnostika visokonaponskih postrojenja.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.316.925
DOI: https://doi.org/10.24867/12BE07Djurdjevic

KONFIGURISANJE I TESTIRANJE MIKROPROCESORSKOG RELEJA ZA ZAŠTITU ELEKTROENERGETSKIH VODOVA

CONFIGURATION AND TESTING OF MICROPROCESSOR BASED RELAY FOR POWER LINES PROTECTION

Milan Đurđević, Zoran Stojanović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu su dati osvnovni principi rada relejne zaštite. Navedene su i opisane osnovne zaštite koje se koriste u štićenju električnih mreža. Kao primjer dat je jedan tip mikroprocesorsog releja i objašnjene su njegove zaštitne funkcije. Nakon detaljnog objašnjenja, praktičnim radom funkcije su ispitane odgovarajućim testnim alatom i uporednom analizom povratnih informacija kako u releju tako i u testnom alatu izvedeni su određeni zaključci.*

Ključne reči: *Relejna zaštita, Mikroprocesorski relej, Elektroenergetski vod*

Abstract – *This document presents the basic principles of relay protection. The basic protection functions used in the protection of elecrtical networks are listed and described. One type of microprocessor relay is used as an example and its protective functions are explained. These functions are tested with an appropriate testing tool and a comparative analysis of the feedback in both the relay and the test tool, certain conclusions were obtained.*

Keywords: *Relay protection, Microprocessor relay, Power line*

1. UVOD

Oblast relejne zaštite predstavlja jednu od najvažnijih oblasti elektroenergetike koja treba da pruži adekvatnu zaštitu elemenata sistema, a posredno i zaštitu rukovaoca tim elementima. Iz tih razloga je neophodno da uređaji relejne zaštite budu ispravni i pouzdani, da bi u svakom trenutku mogli da odreaguju na unaprijed definisan način. Osnovni zadatak elektroenergetskog sistema je da obezbijedi pouzdanu proizvodnju, prenos i distribuciju eleltrične energije krajnjim potrošačima. Postoji više faktora koji utiču na pouzdanost, a najvažniji od njih su:

- topologija mreže;
- način uzemljenja neutralne tačke;
- tipovi kvarova kojima je mreža izložena;
- korišteni sistem relejne zaštite.

Relej predstavlja uređaj koji je napravljen tako da kada se na njegove ulaze dovodi električna, mehanička ili neka druga veličina odgovarajućeg intenziteta, djeluje na unaprijed određen način.

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

Takođe može se reći i da je to uređaj koji služi za detekciju nenormalnih pogonskih stanja djelova elektroenergetskog sistema i inicijalizaciju odgovarajućih upravljačkih akcija za obezbjeđenje normalnog pogona [1]. Pod nenormalnim pogonskim stanjima podrazumijevamo stanja sa kvarom i opasna pogonska stanja [1].

2. ZAŠTITA ELEKTRIČNIH MREŽA

Električne mreže kao dio elektroenergetskog sistema služe za prenos električne energije od izvora do potrošačkih područja, za međusobno povezivanje dijelova elektroenergetskog sistema ili velikih sistema u jedinstven sistem, te za raspodjelu električne energije unutar potrošačkih područja, sve do svakog pojedinačnog potrošača [2].
Prema naponskim nivoima mogu se podijeliti na:

- Niskonaponske mreže – NN napona 400/230 V, u industriji i 500 V;
- Srednjenaponske mreže – SN napona 6 kV u industriji i 10,20 i 35 kV za distribuciju električne energije;
- Visokonaponske mreže – VN napona 110, 220 i 400 kV;
- Mreže vrlo visokih napona – VVN 500, 750 i 1000 kV.

U zavisnosti od izvedbe, mogu biti nadzemne (dalekovodi) i podzemne (kablovi) [2].
Osnovna karakteristika električnih mreža jeste njihova rasprostranjenost na velikim područijima. Vodovi prolaze manje ili više nepogodnim terenima i u pogonu su izloženi mnogobrojnim vanjskim uticajima. Zbog toga su električne mreže dio elektroenergetskog sistema na kojem se pojavljuje relativno veliki broj kvarova [2].

Prema broju faza obuhvaćenih kvarom oni mogu biti:

- Tropolni i dvopolni kratki spojevi bez spoja sa zemljom;
- Tropolni, dvopolni i jednopolni kratki spojevi sa zemljom u mrežama sa efikasno uzemljenim zvjezdištem mreže;
- Zemljospojevi jedne faze u mrežama sa izolovanim ili kompenozovanim zvjezdištem.

Za zaštitu elekričnih mreža koriste se sledeće vrste zaštita [1]:

1. Prekostrujna zaštita;
2. Distantna zaštita;
3. Diferencijalna zaštita;
4. Zaštita osiguračima.

Prekostrujna zaštita predstavlja najjednostavniju i najčešće primjenjivanu zaštitu u srednjenaponskim mrežama (10-35 kV). Primjenjuje se uspješno u mrežama sa radijalnim napajanjem. U zavisnosti od usmjerenja može biti neusmjerena i usmjerena.

Razlikujemo dva načina korištenja ove zaštite i to: sa strujno nezavisnom vremenskom karakteristikom i strujno zavisnom vremenskom karakteristikom. Prednosti prekostrujne zaštite su u njenoj jednostavnosti, sigurnosti i rezervnom djelovanju. Nedostatak je relativno sporo djelovanje kod kvarova u blizini izvora napajanja, naročito ako na radijalnom vodu ima više podstanica [2].

Mreže složenijih oblika sa višestranim napajanjem, kakve su po pravilu visokonaponske mreže za prenos električne energije, ne mogu se selektivno štititi običnim ni usmjerenim prekostrujnim relejima, ali se mogu veoma uspješno štititi pomoću distantnih releja. Pošto vrijeme djelovanja ovih releja zavisi od udaljenosti mjesta kvara, može se postići selektivno djelovanje bez obzira na oblik i uklopno stanje mreže [2].

Ova zaštita ima dva parametra za podešavanje:

1) impedansu pomoću koje se određuje domet odnosno doseg zaštite i

2) vrijeme djelovanja koje u stvari predstavlja vrijeme kašnjenja pri djelovanju zaštite u nekom stepenu.

Kombinacijom ova dva parametra postiže se traženi kvalitet zaštite – što je kvar bliže releju, to se on brže eliminiše [1]. Prednost štićenja vodova pomoću distantne zaštite je u postizanju brzog i selektivnog isključenja kratkih spojeva, kao i u velikoj osjetljivosti zaštite. Prednost je i rezervno djelovanje u slučaju zatajenja prethodne zaštite [2]. Nedostatak distantne zaštite je postojanje zone štićene drugim stepenom sa relativno dugim vremenskim zatezanjem koje je neugodno kod primjene automatskog ponovno uključenja.

Razlikujemo dvije vrste diferencijalne zaštite: podužna i poprečna. Podužna se koristi uglavnom na relativno kratkim vodovima, dok se poprečna primjenjuje na paralelnim vodovima.

Podužna diferencijalna zaštita koristi principe diferencijalne zaštite i osim što se koristi za zaštitu generatora i transformatora, može se koristiti i za zaštitu vodova, poredeći amplitudu i fazni stav struja na početku i na kraju voda. Podužna diferencijalna zaštita se obično primjenjuje za vodove dužina do 15 km i ovaj tip zaštite vodova predstavlja najselektivniji i praktično najbrži tip zaštite koji se može koristiti za eliminaciju kvara na vodu. Sa ovim tipom zaštite kvar se detektuje za oko 30 ms [2].

Prednosti podužne diferencijalne zaštite u odnosu na ostale su to što postiže selektivnost djelovanja zaštite kod kvara na štićenom vodu bez potrebe prilagođavanja i stepenovanja sa susjednim zaštitama, veoma je brza, sigurna je u pogonu, bez dodatnih mjera postiže se istovremeno isključenje prekidača na oba kraja voda, što omogućava efikasnu primjenu brzog automatskog ponovnog uključenja. Glavni nedostatak ove zaštite jeste primjena pilot provodnika koji se mora stalno kontrolisati da eventualno nije prekinut.

Poprečna diferencijalna zaštita se primjenjuje na paralelnim vodovima. Prednost ove zaštite je u njenoj jednostavnosti, brzini djelovanja i u tome što nije potreban spojni vod između početka i kraja štićene dionice. Takođe, prednost je selektivno isključenje kvarova bez potrebe pilotskih veza. Osnovni nedostatak je mala osjetljivost. Kod kratkog spoja na kraju štićenog voda ili u blizini sabirnica susjednog postrojenja razlika struja je suviše mala za djelovanje releja, tako da postoji određena mrtva zona. Zbog toga je potrebna dopunska zaštita i obično je to prekostrujna [2].

3. ABB RED 615 – MIKROPROCESORSKI RELEJ

ABB RED 615 je relej koji se koristi za zaštitu, upravljanje, mjerenje i nadzor u distributivnim mrežama radijalne, prstenaste i mješovite strukture sa generatorima ili bez njih. Dizajniran je za rad u paru u svrhu korištenja podužne diferencijalne zaštite, s tim da ima sposobnost da radi mjerenja i proračune odvojeno za svaku fazu. RED 615 je je član ABB-ove porodice uređaja „Relion“ i dio njihove 615 serije za zaštitu i upravljanje [3]. Izgled releja dat je na slici 1.

Slika 1. *Izgled mikroprocesorskog releja RED 615*

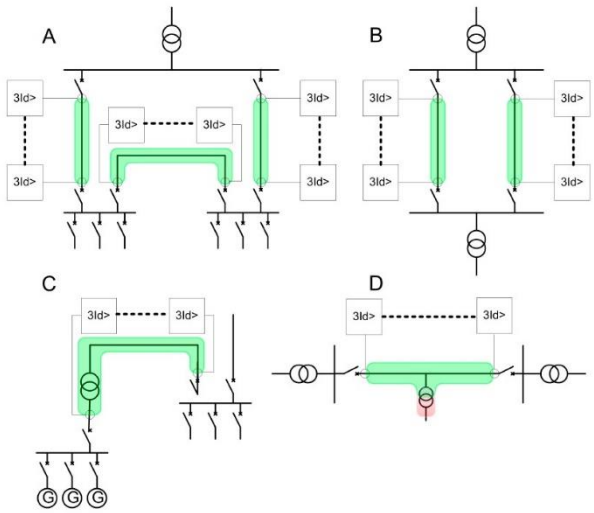
Releji sa dva kraja štićene zone komuniciraju između sebe preko optičkog kabla ili preko galvanizovanih pilot kablova. Ova 615 serija je karakteristična po svom kompaktnom i tzv. izvlačivom dizajnu. Rekonstruisana je ispočetka i vođena standardom IEC 61850 za komunikaciju tako da može da komunicira sa svom opremom u transformatorskoj stanici koja je dizajnirana po ovom IEC standardu [3]. Relej pruža glavnu zaštitu prenosnih vodova i visokonaponskih kablova u prenosnim i distributivnim mrežama. Takođe, posjeduje i prekostrujne zaštite koje se mogu kristiti kao rezervne zaštitne funkcije za lokalnu podužnu diferencijalnu zaštitu ili kao rezerva zaštitama koje su hijerarhijski ispod ovog releja u mreži.
Konfigurisanje releja, podešavanje zaštitnih funkcija, nadzor i upravljanje uređaja omogućava softver PCM600. Ovaj softver pruža veoma širok spektar podešenja vezanih za zaštitne i funkcije upravljanja uređaja odnosno releja u prenosnim i distributivnim mrežama.
U okviru softvera korisnik može da odradi veliki dio zadataka u oblastima planiranja, inženjeringa, testiranja i

analize poremećaja koje uređaj zabilježi. Pomoću različitih alata moguće je obaviti određene zadatke u svrhu upravljanja cijelim objektom kao što je transformatorska stanica.

Relej ABB RED615 posjeduje sledeće zaštitne funkcije:

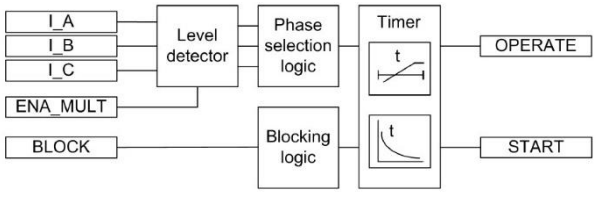
- *LNPLDF* – podužna diferencijalna zaštita;
- *PHxPTOC* – prekostrujna zaštita;
- *NSPTOC* – zaštita od nesimetričnog opterećenja;
- INRPHAR – detekcija udarne struje;
- CCBRBRF – zaštita od otkaza prekidača.

Funkcija podužne diferencijalne zaštite LNPLDF koristi se kao diferencijalna zaštita izvoda u distributivnim mrežama, bilo da su to vodovi ili kablovi [3]. Ova zaštitna funkcija sadrži dva stepena: niži stabilizovani i viši nestabilizovani. Podužna diferencijalna zaštita se takođe može koristiti i kada se u zoni štićena izvoda nalazi i transformator (slika 2).



Slika 2. *Primjene podužne diferencijalne zaštite*

Trofazna prekostrujna zaštita pod nazivom PHxPTOC može se koristiti kao jednofazna, dvofazna ili trofazna neusmjerena prekostrujna zaštita i kao kratkospojna zaštita. Prorada funkcije se dešava u trenutku prekoračenja vrijednosti struje iznad unaprijed podešene vrijednosti u okviru podešenja funkcije. Zaštita može biti aktivirana ili deaktivirana podešavanjem parametara u ovkiru kartice *Operation* u softveru PCM600. Princip rada neusmjerene prekostrujne zaštite ovog releja može se prikazati preko blok dijagrama na slici 3.



Slika 3. *Blok dijagram principa rada neusmjerene prekostrujne zaštite PHxPTOC*

Zaštita od nesimetričnog opterećenja NSPTOC koristi se za povećanje osjetljivosti pri detekciji jednofaznih ili međufaznih kvarova ili za detektovanje neuravnoteženog opterećenja usljed prekinutih provodnika i pojave nesimetričnih napona. Rad funkcije se zasniva na mjerenju negativne komponente struje [3].

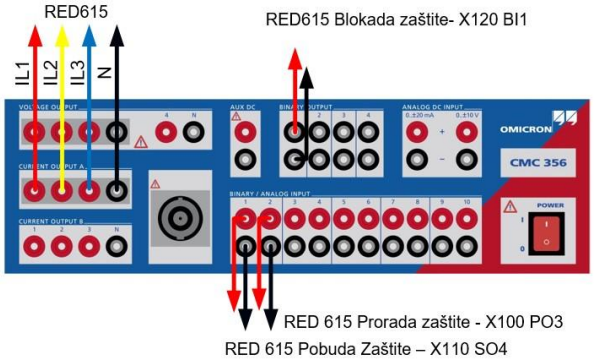
Detekcija tzv. udarne struje transformatora („INRPHAR“) koristi se za podešavanje prorade zaštite u situacijama pojave takvih struja u distributivnim mrežama. Prilikom uključivanja transformatora iz stanja mirovanja dolazi do javljanja udarne struje koja se manifestuje pojavom drugog harmonika polazne struje [3].

Funkcija CCBRBRF, odnosno zaštita od otkaza prekidača, aktivira se komandom za isključenje dobijenom od zaštitnih funkcija. Komanda može biti interna komanda prema terminalima ili komanda koja je stigla na binarne ulaze. Komanda početka rada je uvijek standardna za trofazan rad. Zaštita uključuje trofaznu uslovnu ili bezuslovnu funkciju kojom se šalje ponovljeni signal za isključenje (*retrip*), a takođe i trofaznu uslovnu rezervnu funkciju za isključenje (*back-up-trip*) [3].

4. TESTIRANJE ZAŠTITNIH FUNKCIJA

Ispitivanje i testiranje je izvršeno u laboratoriji firme „Albo Energy“ u Novom Sadu. Kako je na raspolaganju bio samo jedan uređaj RED615, a s obzirom na to da je za podužnu diferencijalnu zaštitu neophodno postojanje oba releja, jer rade u paru, nije bilo moguće odraditi testiranje ove zaštite. Međutim, ispitane su sve ostale zaštitne funkcije: PHIPTOC1, PHLPTOC1, PHHPTOC1, PHHPTOC2, NSPTOC1, NSPTOC2, a takođe ispitana je i funkcionalnost blokiranja rada funkcija.

Za testiranje je korišten ispitni alat proizvođača Omicron i to model CMC356. Ovaj uređaj je korišten za injekciju trofaznih struja, za praćenje izlaza releja koji signaliziraju pobudu i proradu zaštite i za slanje blokade funkcija na relej. CMC356 posjeduje određen broj analognih i digitalnih izlaza i ulaza, međutim za testiranje releja RED615 nisu bili neophodni svi ovi izlazi i ulazi,nego samo dio njih koji je prikazan na slici 4.



Slika 4. *Povezivanje CMC356 sa relejom ABB RED 615*

Da bi se moglo obaviti odgovarajuće testiranje sa ovim uređajem neophodno je i posjedovanje odgovarajućih licenci i softvera za testiranje. Programski paket koji se koristi je Omicron Test Universe.

U svrhu testiranja, bilo je neophodno ožičiti relej na odgovarajući test poligon, a potom ga povezati sa uređajem za testiranje. Na test poligonu prisutno je AC napajanje od 230V, sa zaštitnim automatskim prekidačem, koje obezbjeđuje izvor napajanja releju. S obzirom na to

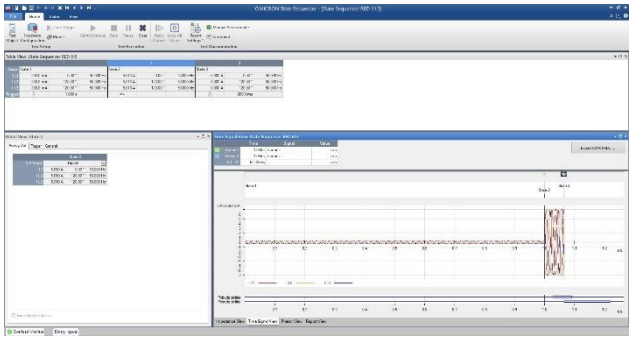
da su digitalni ulazi releja konstruisani za rad na 24 V prisutan je i AC/DC pretvarač od 24 V.
Prilikom testiranja svih zaštitnih funkcija korištena su dva modula u softveru Test Universe i to: *State Sequencer* i *Omicron test document – OCC*. Takođe, u okviru *OCC* modula korišten je i modul Ramping prilikom određivanja vrijednosti pobuda zaštite. Razlog korištenja ova dva modula je da se pokaže da se testiranje može odraditi „ručno“ pomoću State Sequencer, gdje korisnik sve podešava samostalno, dok se kod *OCC* koriste odgovarajući test moduli tipični baš za vrstu zaštite koja se koristi. Npr. *Overcurrent* modul koristi se za sve prekostrujne zaštite.

Prilikom testiranja određene zaštitne funkcije neophodno je prvo podesiti odgovarajuće parametre. Na slici 5 prikazani su podešeni parametri zaštitne funkcije PHIPTOC.

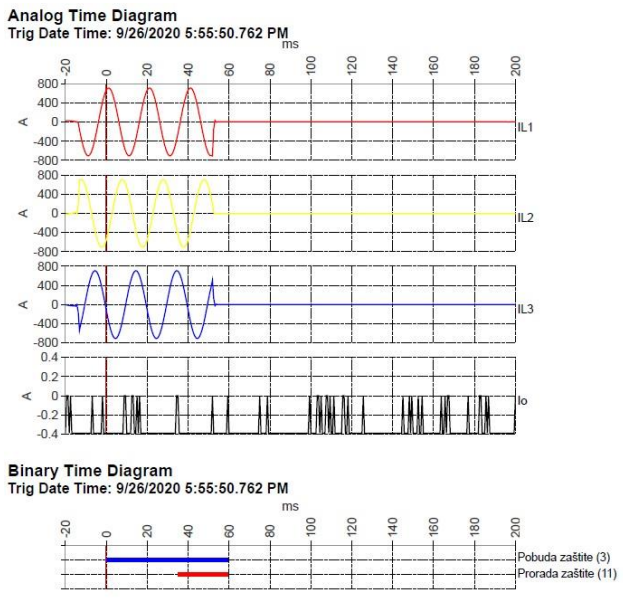
Group / Parameter Name	IED Value	PC Value	Unit	Min	Max
PHIPTOC1: 1					
3I>>>(1)					
Operation	on	on			
Num of start phases		1 out of 3			
Reset delay time		20	ms	0	60000
Setting Group 1					
Start value	5.00	5.00	xIn	1.00	40.00
Start value Mult		1.0		0.8	10.0
Operate delay time		50	ms	20	200000

Slika 5. *Podešeni parametri zaštitne funkcije PHIPTOC*

Za dobijanje režima kvara simuliran je tropolni kratak spoj i iščitani su zapisi o poremećajima iz releja (slike 6 i 7).



Slika 6. *Simulacija tropolnog kratkog spoja*



Slika 7. *Zapis o poremećaju za 3pks u softveru PCM600*

Sa dijagrama se može uočiti da se relej pobudio i proradio na pristutvo odgovarajuće struje kvara.

5. ZAKLJUČAK

Cilj ovog rada bio je da se odradi konfigurisanje mikroprocesorskog releja, a potom sprovedu određena testiranja. Da bi se to moglo realizovati, bilo je neophodno pružiti detaljan opis principa rada i funkcionisanja uređaja, odnosno njegovih zaštitnih funkcija. Nakon toga pristupilo se testiranju.

Da bi testovi bili što kvalitetniji, odnosno da bi se uređaj što bolje ispitao, rađeni su testovi na dva različita režima rada testnog alata i simulirane su situacije sa kvarom i to tropolni kratak spoj i dvopolni kratak spoj u različitim fazama.

Nakon svake simulacije kvara rađena je uporedna analiza podataka dobijena iz zapisa o poremećaju u releju sa povratnim informacijama koje su stizale na testni alat. Priloženim podacima o testiranju u dijelu rada „Testiranje zaštitnih funkcija“, može se zaključiti da su mjerene vrijednosti, kako struje kvara, tako i vremena pobude i prorade zaštite na oba mjesta skoro identične.

S obzirom na to da su sve dobijene vrijednosti u referentnom opsegu i da su svi testovi zadovoljili odgovarajuće kriterijume, izvodi se zaključak da ovaj relej u potpunosti radi ispravno i da bi se u realnom pogonu ponašao baš onako kako se to od njega zahtijeva i na taj način bi zaštitio element na koji je postavljen.

6. LITERATURA

[1] Duško Bekut, „*Relejna zaštita*“, Fakultet Tehničkih nauka, Novi Sad, 2009.
[2] Franjo Božuta, „*Relejna zaštita*“, Elektrotehnički fakultet, Sarajevo, 1991.
[3] ABB, „*615 Series Technical Manual*“, Product version: 5.0 FP1,2018.

Kratka biografija:

Milan Đurđević rođen je u Brčkom 1991. godine. 2013. godine je upisao Fakultet tehničih nauka, studijski program Energetika, elektronika i telekomunikacije. Na studijama se opredijelio za smjer Elektroenergetika – Elektroenergetski sistemi i diplomirao 2017. godine. Master studije upisao je 2017. godine.

Zoran N. Stojanović rođen je 22.07.1979. godine u Požarevcu. Doktorsku disertaciju pod nazivom "Usmereni releji bazirani na digitalnom faznom komparatoru" odbranio je 11.06.2012. godine na Elektrotehničkom fakultetu Univerziteta u Beogradu. Oblasti naučnoistraživačkog rada kojima se do sada bavio su relejna zaštita, razvodna postrojenja i monitoring i dijagnostika visokonaponskih postrojenja.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38:620.9
DOI: https://doi.org/10.24867/12BE08Vasic

PRIMENA NEURALNIH MREŽA ZA DETEKCIJU PROPADA NAPONA NA PRIMERU RADA TEST MREŽA

APPLICATION OF NEURAL NETWORKS FOR VOLTAGE DIPS DETECTION ON THE EXAMPLE OF TEST GRIDS OPERATION

Ivan Vasić, Vladimir A. Katić, Aleksandar M. Stanisavljević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je obrađena primena pametnih tehnologija, odnosno neuronskih mreža u energetici, pri čemu se njihova primena vrši u softverskom paketu MATLAB. Data je studija slučaja detekcije kvarova u IEEE13 test mreži sa dodatim obnovljivim izvorom energije. Studija slučaja pokazala je da primena neuronskih mreža može biti od velike koristi u ovoj grani, što pokazuju rezultati, kojima je dokazano da se obučavanje neuronske mreže pokazalo ispravnim u cilju detekcije kvarova.*

Ključne reči: *IEEE 13-bus test mreža, neuronske mreže, propadi napona.*

Abstract – *In the thesis applications of neural networks in power engineering is described, especially in MATLAB software package. A case study of fault detection in the IEEE13 test grid with inserted renewable energy resource is given. It is shown that neural network gives great result in this branch, following impressive results for fault detection in IEEE13 test grid.*

Keywords: *IEEE 13-bus test grid, neural networks, voltage dips.*

1. UVOD

Razvoj tehnologije doprineo je mnogo većim mogućnostima i sposobnostima današnjih računara, odnosno značajnom poboljšanju njihovih performansi. To je dovelo do njihove intenzivnije u različitim oblastima, od svakodnevnog života, medicine, telekomunikacija, pa i do energetike. Značaj računara u energetici doprineo je digitalizaciji upravljanja, čime se povećava sigurnost rada sistema, process održavanja (čuvaju se ljudski životi) i planiranja. Sve ovo čini osnovu formiranja tkzv. Pametnih mreža, koje se odlikuju velikim udelom IT tehnologija.

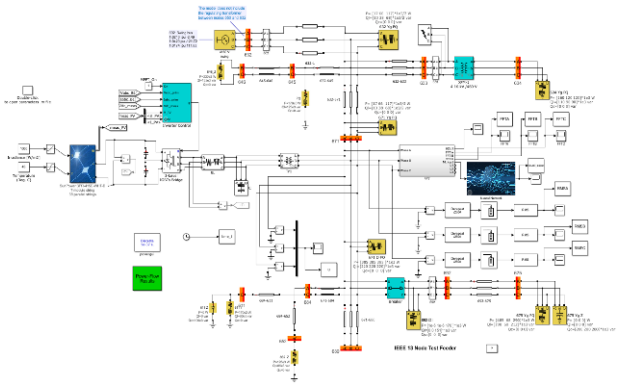
Za operativni rad mreža, posebno je važna njihova primena za detekciju i lokalizaciju kvarova, njihovu izolaciju, pravilnu raspodelu opterećenja, brzu rekonfiguraciju mreže, kontrolu rada sistema i dr. Analiza velikog broja podataka zahteva primenu „veštačke inteligencije“, odnosno neuronskih mreža, koje rešavaju probleme prepoznavanja šablona, predikcije, klasifikacije, upravljanja sistemom [1].

Tradicionalni postupak rešavanja, gde se algoritmi pišu za neki specifični problem, uslovljava dobro poznaje domena, kao i sam problem, kao i postojanje adekvatnog rešenja. Kod metode mašinskog učenja, postupak obuhvata postavljanje početnog stanja, odnosno učitavanje adekvatnih ulaznih podataka, a zatim se dobijeni rezultati ocenjuju po unapred definisanim kriterijumima. Ukoliko odgovori zadovoljavaju kriterijume, daje se pozitivna ocena i obrnuto, na bazi čega se ažurira stanje metode. Time ona u narednoj iteraciji daje bolje, tj. tačnije rešenje, odnosno dobijeni rezultat u većoj meri odgovara stvarnom stanju. Ovaj proces se ponavlja iterativno dok se ne postigne željena tačnost [2].

U ovom radu razmatrana je i razvijena primena neuronskih mreža za detekciju propada napona u mreži, čime se omogućuje efikasniji i kvalitetniji rad, pogotovu u slučaju većeg prisustva obnovljivih izvora. Radi univerzalnosti predloženog rešenja, za testiranje je uzeta standardna IEEE test mreža sa 13 sabirnica (*IEEE 13-bus test grid*) modifikovana dodatkom obnovljivih izvora.

2. MODIFIKOVANA IEEE 13-BUS TEST MREŽA

Na slici 1 dat je prikaz korišćenog referentnog modela distributivne mreže u okviru ovog rada. Radi se IEEE 13 – bus test mreži, koja je modifikovana priključenjem fotonaponske (FN) elektrane nominalne snage 250 kW preko transformatora 0.25/4.16 kV/kV kod sabirnice 671. Za potrebe ovog rada, pretpostavljena je standardna vrednost solarne iradijacija od 1000 W/m^2, kao i njena nepromenljivost u opsegu razmatranja. Kvar je generisan na sabirnici 633, a detekcija sa predloženom primenom neuronske mreže je urađena na mestu priključka FN elektrane.



Slika 1. *Modifikovana IEEE 13 – bus test mreža*

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

Prema IEEE standardu 1159-2009 propadi napona definisani su kao redukcija napona u rasponu od 10% do 90% od nominalne vrednosti napona, kada je frekvencija sistema nominalna i kada je trajanje poremećaja u rasponu od pola periode do jednog minuta [2]. Naponski signal ne može biti direktno iskorišćen za detekciju ili klasifikaciju događaja i poremećaja u EES. Zbog toga jednostavna i najčešće korišćena metoda je bazirana na direktnom proračunu efektivne vrednosti napona iz sinusnog oblika napona (RMS metoda):

$$V_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} v_i^2} \quad (1)$$

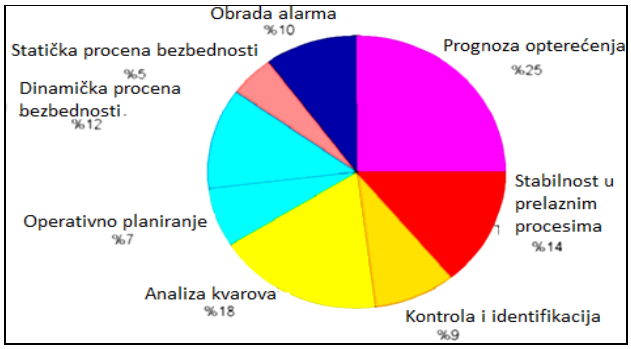
Efikasnije i vremenski brže rešenje je umesto RMS vrednosti koristiti metodu harmonijskog otiska (engl. *Harmonic Footprint*), u kojoj se detekcija bazira na redukovanom setu harmonika [2].

3. NEURONSKE MREŽE

Neuronske mreže (engl. *Neural Networks - NN*) predstavljaju sisteme za procesiranje informacija, poseduju mogućnost učenja, memorisanja i generalizacije na osnovu dobijenih podataka za obučavanje. Sastoje se od velikog broja gusto povezanih procesorskih elemenata, tzv. čvorova, koji uobičajeno rade u paraleli i koji su organizovani po nekim regularnim arhitekturama. Razvoj neuronskih mreža nastao je motivacijom da se modeliuje ponašanje (bioloških) neurona u mozgu.

Neuroni se grupišu u slojeve. U okviru jednog sloja, svaki neuron tog sloja dobija isti ulaz. Slanjem istog ulaza u različite neurone postiže se otkrivanje različitih informacija i donošenje novih zaključaka o ulaznim podacima. To se postiže tzv. težinama, odnosno težinskim signalima, koji se definišu za svaki ulazni signal neurona i svaki od njih ima ulogu da direktno promeni svoj ulazni signal pre nego što isti stigne u aktivacionu funkciju.

Mogućnosti primene neuronskih mreža u EES predstavljene su na slici 2. Kao što se može videti, postoji čitav niz veoma korisnih aplikacija, od kojih su najčešće prognoza opterećenja (25%) i analiza kvarova (18%).



Slika 2. *Primena neuronskih mreža u EES [3]*

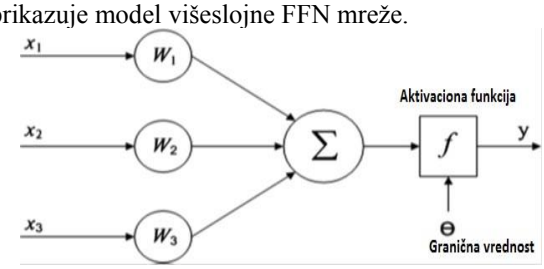
Veštačke neuronske mreže se u najopštijem smislu mogu podeliti na neuronske mreže sa prosleđivanjem unapred, rekurentne mreže i na njihove hibride.

3.1. Neuronske mreže sa prosleđivanjem unapred

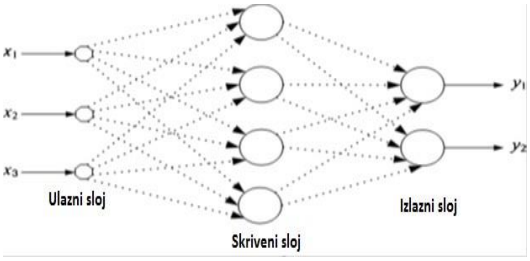
Neuronske mreže sa prosleđivanjem unapred (eng. *Feed Forward Neural Network - FFN*) predstavljaju najjednostavniji tip neuronskih mreža. Kod njih je tok informacija jednosmeran i ide od ulaznih jedinica, prolazi kroz skrivene slojeve (ukoliko ih ima), do izlaznih jedinica, odnosno perceptrona (slika 3) [4]. Mogu biti jednoslojne i višeslojne.

Jednoslojna mreža perceptrona sastoji se od samo jednog izlaznog sloja, a ulazi se prosleđuju izlaznima preko niza težina. Suma proizvoda težina i stanja ulaza izračunava se u svakoj jedinici, pa ukoliko je vrednost iznad neke granične vrednosti (obično 0), neuron daje signal i preuzima aktiviranu vrednost (obično 1), dok u suprotnom, uzima deaktiviranu vrednost (obično -1) [5].

Višeslojni perceptroni sastoje se od više međusobno povezanih slojeva, tako da se informacije idu isključivo unapred. Svaki neuron u određenom sloju poseduje usmerene veze ka neuronima u narednom. Najčešće se u ovakvim mrežama primenjuje sigmoidna funkcija, kao funkcija aktivacije. Višeslojne mreže koriste veliki broj tehnika za obučavanje, od kojh najpoznatiju predstavlja metod propagacije unazad (eng. *Back Propagation – BP*). Izlazne vrednosti porede se sa tačnim vrednostima u cilju izračunavanja vrednosti neke predefinisane funkcije greške. S obzirom da se greška propagira unazad kroz mrežu, težinski faktori se ažuriraju, odnosno popravljaju kako bi se smanjila greška za neku malu vrednost. Slika 4 prikazuje model višeslojne FFN mreže.



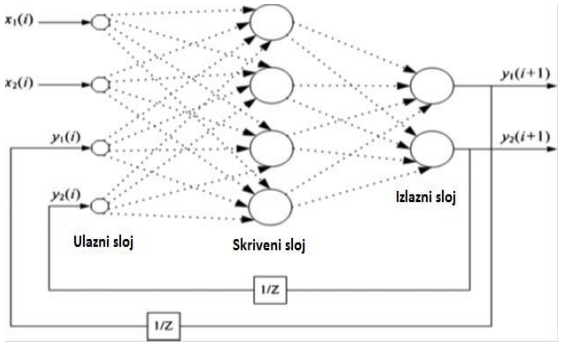
Slika 3. *Jednoslojna FFN mreža [4]*



Slika 4. *Višeslojna FFN mreža [4]*

3.2. Rekurentna neuronska mreža

Nasuprot FFN mreži, rekurentne neuronske mreže (engl. *Recurrent neural networks*, RNN) predstavljaju modele sa dvosmernim protokom podataka, odnosno podaci se propagiraju od daljih stadijuma procesiranja ka ranijim. Slika 5 prikazuje izgled jedne RNN.



Slika 5. *Rekurentna neuronska mreža [4]*

3.3. Obučavanje neuronskih mreža

Procedura koja služi za obučavanje neuronske mreže predstavlja algoritam obučavanja. Kroz ovu proceduru se na algoritamski (sistematski) način menjaju sinaptičke težine u cilju dostizanja željenih performansi. Ima više metoda učenja, od kojih je ovde primenjeno nadgledano učenje. Ono se karakteriše činjenicom da su uz ulazne vrednosti date i izlazne vrednosti koje im odgovaraju. Ulazne i izlazne vrednosti najčešće se predstavljaju u vektorskom obliku i obično se označavaju sa x i y, pri čemu se ulazne promenljive često nazivaju atributima, a izlazne ciljnim. U opštem slučaju, pretpostavka je da je odnos između ulaznih i izlaznih promenljivih dat zajedničkom raspodelom verovatnoće (x, y). U najvećem broju slučajeva, pod ovom verovatnoćom podrazumeva se gustina raspodele. Obično ova raspodela nije dostupna, pa se iz tog razloga pristupa određivanju modela $f(x)$ koji ulaznim vrednostima pridružuje izlazne promenljive. Ovakvih modela može biti veliki broj, pa je od značaja najbolji takav model. Zbog ovoga, definiše se funkcija greške, koja se koristi za merenje odstupanja predviđenih i stvarnih vrednosti izlazne promenljive koja se označava sa $(y, (x))$. Greška koja se dešava na kombinacijama koje su verovatnije, ima veći značaj, pa se gustina raspodele kombinacija (x, y) koristi za usrednjavanje grešaka, čime se definiše stvarni rizik [6].

$$R(f) = E[L(y; f(x))] = \int L(y, f(x))\, p(x; y)dxdy \qquad (2)$$

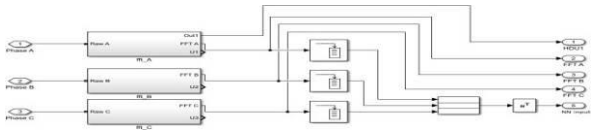
S obzirom da je pojmom rizika definisan kvalitet modela, neophodno je naći funkciju koja minimizuje rizik, odnosno rešava se problem:

$$\min_f (f) \qquad (3)$$

4. PRIMENA NEURONSKIH MREŽA ZA DETEKCIJU PROPADA U IEEE13 TEST MREŽI

4.1. Priprema podataka

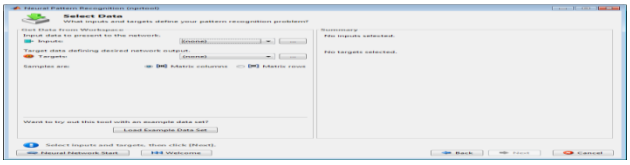
Nakon nekoliko pokretanja simulacija, sa ciljem da se prikupi dovoljna količina podataka, izvršeno je snimanje signala kvarova u korišćenom modelu (slika 1). Detekcija je izvršena analizom harmonijskog spektra naponskih signal (FFT). Blok je podešen tako da se traže niži harmonici, odnosno oni koji čine Harmonijski otisak i to u sve tri faze, na osnovu kojih se dalje vrši detekcija propada napona pomoću neuronske mreže. Taj signal je ubačen u bazu podataka i dalje se priprema, u vidu smeštanja podataka u matricu nad kojom je izvršeno transponovanje kako bi se prilagodila za ulaz u neuronsku mrežu. FFT blok prikazan je na slici 6.



Sl. 6. *FFT blok*

4.2. Obučavanje

Nakon što je obezbeđena baza podataka, u vidu signala napona za različite tipove propada, prelazi se na odabir tipa neuronske mreže i obučavanje iste. Za potrebe ovog rada, korišćena je jednostavna FFN neuronska mreža. Nalazi se u MATLAB-u pod nazivom *Neural Net Pattern Recognition* i koristi za kreiranje i obučavanje jednostavne dvoslojne FFN, sa ciljem da se izvrši detekcija propada napona u korišćenoj mreži. Baza podataka snimljenih propada unosi se u polje „Inputs“ u aplikaciji, kao na slici 7.



Slika 7. *Unos podataka za neuronsku mrežu*

Posle pripremljenih ulaznih podataka za neuronsku mrežu i podešavanja arhitekture, počinje njeno obučavanje. Obučavanje se vrši u 1000 epoha, što je podrazumevana vrednost ovog parametra. Početak obučavanja neuronske mreže prikazan je na slici 8.

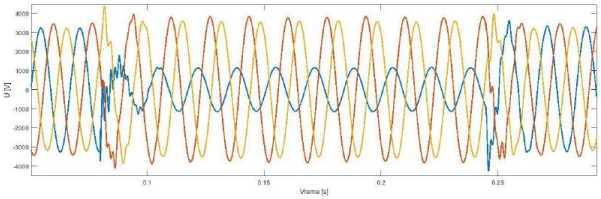
Slika 8. *Obučavanje neuronske mreže*

4.3. Rezultati

Izvršene su simulacije za dve vrste propada napona kao posledice jednopolnog kratkog spoja faze A sa zemljom (slučaj 1) i međufaznog kratkog spoja faze A i faze B (slučaj 2). Snimljeni su odzivi signala za detekciju primenom neuronskih mreža i pomoću standardne RMS metode. Celokupna simulacija traje 1 s, pri čemu su kvarovi simulirani u periodu od t=0,08 s do t=0,24 s.

4.3.1. Jednopolni kratak spoj faze A sa zemljom

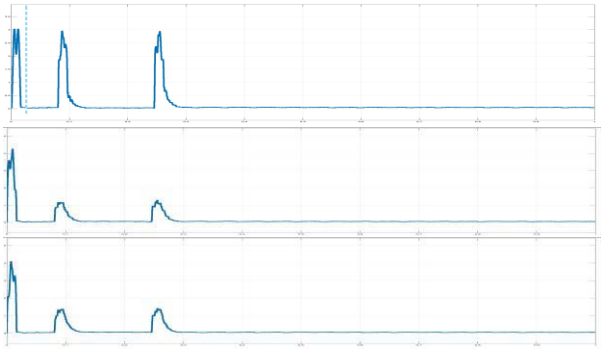
Nakon što se dogodio kvar u t=0,08 s, dolazi do izobličenja talasnog spektra napona sve tri faze, pri čemu se značajnija izobličenja javljaju upravo u fazama u kojima se dogodio kvar, što je u ovom slučaju faza A (slika 9).



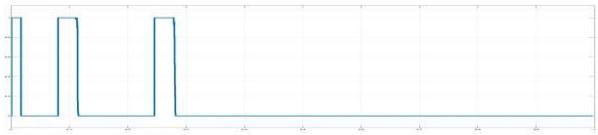
Slika 9. *Talasni oblici napona za slučaj 1*

Može se jasno uočiti da se u trenutku nastanka propada napona (t=0,08 s), pojavljuje izobličenje (značajno učešće viših harmonika u talasnom obliku napona), koje se ponavlja u trenutku prestanka kvara, odnosno početka oporavljanja naponskih prilika (t=0,24 s). Na slici 10 prikazani su ovi signali za svaku fazu, s tim da prvi impuls, odnosno izobličenje za t=0 s treba zanemariti, jer je posledica početnog oscilovanja modela, a ne kvara u mreži.

Ova izobličenja (po fazama) predstaljvaju signal neuronskoj mreži da se dogodio propad napona i ona generiše signal 1. Na slici 11 vidi se da neuronska mreža (FNN) generiše signal 1 kada je detektovala propad (t=0,08067 s) i kada je prestao (impuls u t=0 s je vezan za početne oscilacije modela i zanemaruje se). Može se zaključiti da je neuronska mreža uspešno detektovala pojavu propada i to za svega Δt=0,67 ms.

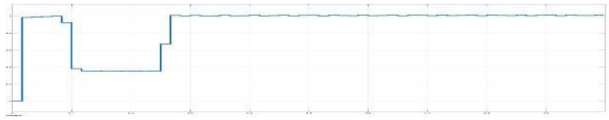


Slika 10. *Harmonijski otisak - izobličenje napona faze A (gore), faze B (u sredini) i faze C (dole) za slučaj 1*



Slika 11. *Izlazni signal bloka neuronske mreže (slučaj 1)*

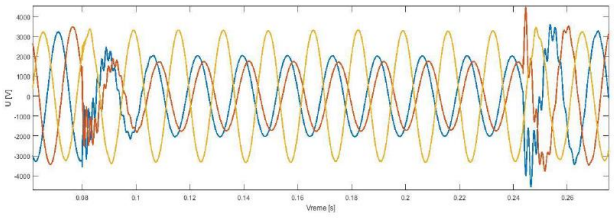
Radi poređenja, urađena je simulacija detekcije istog propada napona pomoću standardne RMS metode. Rezultat je prikazan na slici 12. Vidi se RMS napona faze A opada na vrednost manju od 90% nominalne u t=0,1 s, što kada se uzme u obzir da je trenutak nastanka propada definisan u t=0,08 s, pokazuje da je vreme detekcije Δt=20 ms.



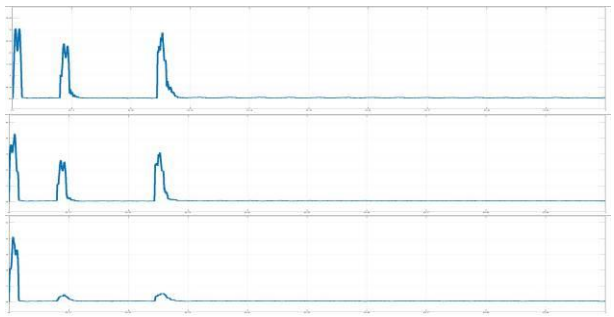
Slika 12. *RMS metoda detekcije propada – slučaj 1*

4.3.2. Međufazni kratak spoj faze A i faze B

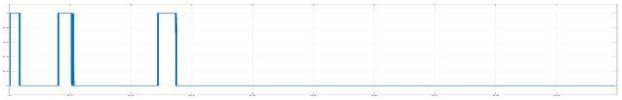
U slučaju 2, nakon što se dogodio kvar u fazama A i B u trenutku t=0,08 s, dolazi do izobličenja talasnih oblika napona (slika 13). Odgovarajući Harmonijski otisak prikazan je na slici 14, za sve tri faze. Izlazni signal neuronske mreže, kao i odziv dobijen RMS metodom prikazani su na slikama 15 i 16. Vidi da je FNN detektovala propad za svega Δt=0,65 ms, dok je primenom RMS metode bilo potrebno opet Δt=20 ms.



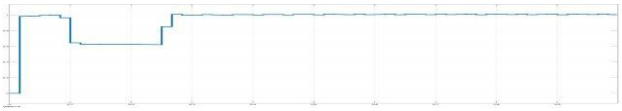
Slika 13. *Talasni oblici napona za slučaj 2*



Slika 14. *Harmonijski otisak - izobličenje napona faze A (gore), faze B (u sredini) i faze C (dole) – slučaj 2*



Slika 15. *Izlazni signal bloka neuronske mreže – slučaj 2*



Slika 16. *RMS metoda detekcije propada – slučaj 2*

5. ZAKLJUČAK

Pametne tehnologije imaju sve veću primenu u svim sferama života, pa tako i u elektroenergetici. U budućnosti, može se očekivati dodatna ekspanzija primene pametnih tehnologija, primena neuronskih mreža.

Rezultat rada pokazao je da neuronske mreže mogu uspešno detektovati svaku vrstu propada napona značajno brže od standardne RMS metode.

6. LITERATURA

[1] D. J. Đozić: „Upotreba veštačkih neuronskih mreža za predviđanje ponašanja i upravljanje složenim elektroenergetskim sistemima“, Fakultet tehničkih nauka, Doktorska disertacija, Novi Sad, 2020.

[2] V. A. Katić, A.M. Stanisavljević, “Smart Detection of Voltage Dips Using Voltage Harmonics Footprint”, *IEEE Transaction on Industry Application*, Vol.54, No.5, Sep./Oct. 2018, pp.5331-5342,

[3] M. T. Hagh: “Application of Neural Networks in Power Systems”, WAS, Engineering and Technology, No.6 2005, pp.53-57.

[4] L. H. Hassan, M. Moghavvemi, Haider A.F. Almurib, Otto Steinmayer: “Current state of neural networks applications in power system monitoring and control“

[5] K.-L. Du, M.N.S. Swamy: “Neural Networks and Statistical Learning”, Springer-Verlag London, 2014.

[6] M. M. Milosavljević: „Veštačka inteligencija“, Univerzitet Singidunum, Beograd, 2015.

Kratka biografija:

Ivan Vasić rođen je 1996. god. u Smederevu. Osnovne studije završio je na Fakultetu tehničkih nauka 2019. god., a master 2020. god. iz oblasti Elektrotehnike i računarstva.

Vladimir A. Katić rođen je 1954. god. u Novom Sadu. Doktorirao je na Univerzitetu u Beogradu 1991. god. Od 2002. god. je redovni profesor Univerziteta u Novom Sadu. Oblasti interesovanja su mu energetska elektronika, kvalitet električne energije, obnovljivi izvori električne energije i električna vozila.

Aleksandar M. Stanisavljević, rođen je u Beogradu 1988. god. Doktorirao ne na Univerzitetu u Novom Sadu 2019. god. gde je trenutno u zvanju docenta. Oblast interesovanja su mu integracija obnovljivih izvora energije na mrežu i kvalitet električne energije.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.39
DOI: https://doi.org/10.24867/12BE11Slijepcevic

INTELIGENTNA EKSTRAKCIJA KVANTITETA IZ KONTINUALNOG MULTIMEDIJALNOG STRIMA

INTELLIGENT QUANTITY EXTRACTION FROM A CONTINUOUS MULTIMEDIA STREAM

Nikola Slijepčević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisan je problem inteligentne ekstrakcije kvantiteta i prepoznavanja ručno pisanih cifara iz multimedijalnog strima, kao i jedno rješenje opisanog problema. Problem je rješavan uz pomoć vještačke inteligencija, mašinskog učenja, neuronskih mreža i raznih metoda za obradu slike. Programsko rješenje ovog problema pisano je u programskom jeziku Python uz korišćenje pripadajućih biblioteka.*

Ključne reči: *Ekstrakcija kvantiteta, Prepoznavanje cifara, Digitalna slika, Mašinsko učenje, Neuronske mreže*

Abstract – *This paper describes the problem of intelligent quantity extraction and recognition of handwritten digits from a multimedia stream, as well as a solution to the described problem. The problem was solved with the help of artificial intelligence, machine learning, neural networks and various image processing methods. The software solution to this problem is written in the Python programming language using the associated libraries.*

Keywords: *Quantity extraction, Digit recognition, Digital image, Machine learning, Neural networks*

1. UVOD

Od najranijih početaka ljudske civilizacije čovjekovo djelovanje bilo je usmjereno tehnološkom napretku. Danas se taj napredak posebno ogleda u razvoju računarskih tehnologija, koje ujedinjuju naučnike različitih usmjerenja te inženjere raznih struka čiji je posao otkrivanje, razvoj i oblikovanje novih pristupa raznim problemima. Skupu takvih oblasti pripada i razvoj neuronskih mreža i mašinskog učenja, odnosno naučno-tehnološka disciplina koja se bavi teorijom izrade vještačke inteligencije i programa koji se bave dobavljanjem objekata iz videa ili slika te njihovom ekstrakcijom. Temelj razvoja čini mogućnost računarske percepcije ljudskog vida i digitalizacije slike, a djelovanje je usmjereno automatizaciji i integraciji širokog spektra procesa te prezentaciji vizuelne percepcije.

Klasični problemi ovog područja su analiza pokreta, restauracija slike, rekonstrukcija događaja, prepoznavanje i ekstrakcija objekata.

U ovom radu predstavljeni su problemi i rješenje inteligentne ekstrakcije objekata iz multimedijalnog strima tj. preuzimanjem slika, kao frejma, iz video zapisa, zatim izdvajanjem regiona od značaja, potom prepoznavanje objekata, gdje su objekti Arapski ručno ili kompjuterski (mašinski) pisani brojevi.

Danas se većina dostupnih informacija nalazi u elektronskom ili u obliku fotografija i video zapisa. Pa i iz tog razloga razvijaju se sve više algoritmi digitalne obrade podataka. Algoritmi digitalne obrade i analize slike koji danas postoje uglavnom su orijentisani na automatsku detekciju i prepoznavanje jednog ili ograničenog skupa objekata ili kategorija, dok algoritam koji bi bio u stanju detektovati i prepoznavati proizvoljne objekte nisu još u potpunosti razvijeni.

Zavidni rezultati neuronskih mreža u brojnim zadacima vještačke inteligencije, čiju ilustraciju vidimo na slici 1, pokazuju kako je to područje s izrazito velikim potencijalom. Nezavisno o tome radi li se o obradi i analizi slika, zvuka, video zapisa, teksta ili bilo kog drugog ulaznog podatka, neuronske mreže sve značajnije prednjače kao najbolji odabir algoritma za učenje.

Slika 1. *Vještačka inteligencija ilustracija*

2. NEURONSKE MREŽE

Neuronska mreža je skup međusobno povezanih jedno-
,stavnih elemenata obrade, jedinica ili čvorova. Sposob-
,nost obrade mreže je posljedica jačine veza između tih elemenata, a postiže se kroz proces adaptacije ili učenjem iz već postojećeg skupa primjera [1].

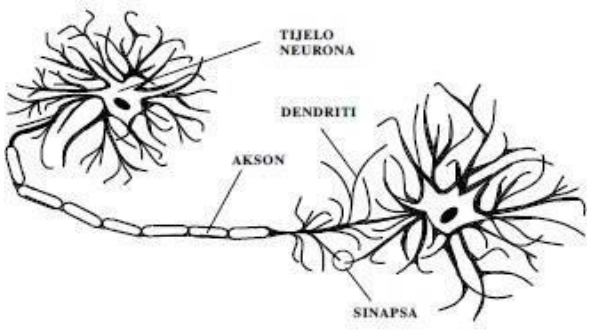
Neuronska mreža je masovno paralelizovan distribuirani procesor sa prirodnom sposobnošću memorisanja iskustvenog znanja i obezbjeđivanja njegovog korišćenja.

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

2.1. Neuroni, biološki i vještački

Sastavni dijelovi vještačkih neuronskih mreža su vještački neuroni. Model vještačkog neurona nastao je pokušajem imitacije načina preuzimanja i obrade informacija biološkog neurona.

Biološki neuron, čiji je pojednostavljeni oblik prikazan na slici 2, sastoji se od tijela, aksona i velikog broja dendrita koji okružuju tijelo neurona.



Slika 2. *Pojednostavljena slika biološkog neurona*

Neuron je modul koji radi nelinearnu transformaciju ulaznog podatka. To je matematička funkcija koja ulazni vektor realnih brojeva transformiše parametrizovanom linearnom transformacijom, primjenjuje nelinearnu aktivacijsku funkciju i na izlazu daje jednu realnu vrijednost. To se može predstaviti jednačinom (1).

$$y = f(\vec{x} \cdot \vec{w} + b) = f(\sum_{i=1}^{n} x_i \cdot w_i + b) \qquad (1)$$

Gdje je:

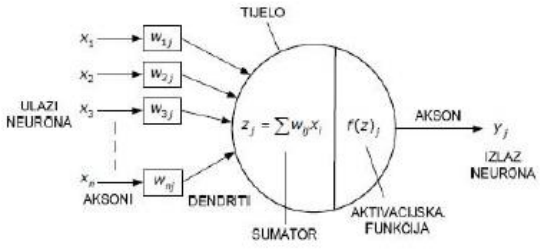
X_i – ulazni vektor;

W_i – vektor težina;

b – prag neurona.

Analogija između biološkog i vještačkpg neurona je sledeća: tijelo se zamijenjuje sa sumatorom, ulogu dendrita preuzimaju ulazi u sumator, izlaz iz sumatora je akson vještačkog neurona, a ulogu praga osjetljivosti biološkog neurona preuzimaju aktivacijske funkcije. Signali postaju neumeričke vrijednosti, dok se vrijednosti sinapse opisuju težinskim faktorima.

Težinski faktori povezuju izlaze drugih neurona sa ulazima sumatora. Izlaz iz sumatora povezuje se na ulaz aktivacijske funkcije na čijem se izlazu reprodukuje izlaz vještačkog neurona. Na slici 3 vidimo prikaz vještačkog neurona.



Slika 3. *Prikaz strukture vještačkog neurona*

2.2. Aktivacijske funkcije

Na izlazima slojeva neuronskih mreža koriste se aktivacijske funkcije da bi se u model unijela nelinearnost. Bez njih bi mreža mogla naučiti samo linearnu granicu između podataka različitih klasa.

Funkcije aktivacije su matematičke jednačine koje određuju izlaz neuronske mreže. Funkcija je vezana za svaki neuron u mreži i određuje da li je treba aktivirati ili ne, na osnovu toga da li je ulaz svakog neurona relevantan za predviđanje modela

2.3. Vrste vještačkih neuronskih mreža

Kada bi jedan neuron radio samostalno imao bi velika ograničenja u pogledu realizacije nekog konkretnog zadatka. Međutim, povezivanjem više neurona na smislen i odgovarajući način, dobija se mreža koja može riješiti kompleksnije probleme uz zadovoljavajuće rezultate.

Vještačke neuronske mreže mogu biti jednoslojne i višeslojne [2]. Višeslojne mreže imaju ulazni i izlazni sloj, a između njih se nalazi jedan ili više skrivenih slojeva neurona. U odnosu na vrijeme razlikuju se vremenski-neprekidne i vremenski-diskretne neuronske mreže. Zavisno od područja primjene imamo perceptronske, asocijativne, dvostruko asocijativne, adaptivne neuronske mreže. Takođe, može se izvršiti podjela mreža prema smjeru u kojem signali putuju. Ukoliko signali putuju samo u jednom smjeru, od ulaznog do izlaznog sloja, riječ je o jednosmjernim neuronskim mrežama. U slučaju da postoji sloj ili dio mreže u kojem signal putuje u suprotnom smjeru tj. mreža ima povratnu petlju govorimo o povratnim neuronskim mrežama.

2.4. Učenje vještačkih neuronskih mreža

Za mrežu, učenje predstavlja proces mijenjanja težina, koji se odvija u odnosu na unesene ulazne i izlazne promjenljive.

Ukoliko su poznate željene vrijednosti izlaznih promjenljivih, govori se o nadgledanom učenju. Nadgledano učenje je iteracijski postupak koji podrazumijeva postojanje spoljašnjeg faktora koji koriguje rezultate mreže, sve dok se ne dobije željeno ponašanje mreže.

Kod nekih mreža nisu predstavljeni željeni izlazi mreže. Tada se govori o nenadgledanom učenju. Na ulaz mreže dovedu se ulazne promjenljive nakon čega mreža sama podešava težine po dobro definisanom algoritmu.

2.5. Primjena vještačkih neuronskih mreža

Možda najveća prednost vještačkih neuronskih mreža je u njihovoj mogućnosti da se koriste kao proizvoljan mehanizam aproksimacije funkcija koje uče na osnovu podataka

Danas se neuronske mreže primjenjuju u mnogim segmentima života poput računarskih nauka, medicine, bankarstva, fizike itd., najčešće za sledeće zadatke: prepoznavanje šablona i uzoraka, obrada slike, praćenje objekata, obrada govora, problemi optimizacije, nelinearno upravljanje, obrada nepreciznih i nekompletnih podataka, simulacije i sl. Iako je već dosta postignuto na području neuronskih mreža, daljim razvojem tehnologije i one će se dalje razvijati i širiti svoje područje primjene.

3. DIGITALNA SLIKA I NJENA OBRADA

Digitalna slika je numerička (najčešće cjelobrojna) reprezentacija dvodimenzionalne slike.

Vektorske slike, odnosno njihov sadržaj, opisan je prostim geometrijskim objektima, kao što su tačke, prave, krive i poligoni. Svaki od ovih objekata je predstavljen relativno malom količinom podataka, kao što su pozicija u koordinatnom sistemu, dimenzije, stil i boja linije i stil i boja objekta.

Kod rasterskih slika podaci su predstavljeni u vidu konačne matrice, u kojoj se svaki element naziva piksel. Svaki piksel, kao najmanja jedinica slike, sadrži podatke o boji i intenzitetu, dok je njegov položaj na slici određen položajem unutar matrice.

3.1. Tipovi digitalnih slika

Digitalna slika je prikaz dvodimenzionalne slike sa konačnim skupom digitalnih vrijednosti koje se nazivaju pikseli. Pikseli se čuvaju u računarskoj memoriji ili hard disku kao rasterka slika ili rasterska mapa, odnosno dvodimenzionalni niz malih cjelina. Neke od najpoznatijih predstava slike su binarne slike, slike u nijansama sive, slike u boji, indeksirane slike.

Kod binarnih slika se koristi jedan bit za predstavljanje jednog piksela. Kako bit ima svega dva stanja, 0 i 1, to znači da je svaki piksel u jednoj od dvije predefinisane boje, a najčešće su to crna i bijela. Nemogućnost prikazivanja međunijansi sive je veliko ograničenje za binarne slike u prikazivanju fotografija.

Slike u nijansama sive, koje se često nazivaju i crno-bijelim slikama, sačinjene su od piksela čije boje su različiti intenziteti sive, u rasponu od bijele do crne boje. Najčešće se radi o 256 nivoa intenziteta sive, pa je za prikaz jednog piksela potreban jedan bajt. Imajući u vidu da je eksperimentima pokazano da ljudsko oko može da razlikuje najviše 200 nijansi sive, 256 nivoa je i više nego dovoljno da prikaže pun spektar između bijele i crne bez uočljivih prelaza, pa je ovaj tip digitalnih slika pogodan za prikaz fotografija.

Slike u boji čine pikseli koji pored informacija o intenzitetu sadrže i informacije o boji. Za vizuelno prihvatljive rezultate, neophodne su tri vrijednosti, odnosno kanala, koji opisuju boju svakog piksela u određenom prostoru boja. Najčešće korišćeni prostor boja je RGB, ali u upotrebi su i brojni drugi, kao što su HSV, CMYK, YCbCr itd. Kod RGB prostora se koristi 256 nivoa za svaku komponentu, tako da su potrebna tri bajta za prikaz jednog piksela. Ukupan broj boja koji se može prikazati u ovom prostoru je 16 777 216.

Indeksirane slike u boji imaju ograničenu paletu boja. Nazivaju se indeksirane zato što je boja piksela predstavljena indeksom, odnosno pozicijom te boje u predefinisanoj paleti. Kod prikazivanja fotografija pomoću indeksiranih slika dolazi do izvjesnog gubitka informacija, ali to u mnogome zavisi od broja i izbora boja u paleti, kao i od opsega boja fotografije. Zbog ograničenja starijih monitora koji su mogli da prikažu samo 256 boja, bilo je pokušaja da se formira paleta boja koja bi bila najbolje prilagođena tadašnjem hardveru. Izabrano je 216 boja, koje se dobijaju kombinacijom po 6 nijansi crvene, zelene i plave.

3.2. Obrada digitalnih slika

Metode obrade slika se mogu podijeliti u dvije osnovne grupe, zavisno od vrste ulaznih podataka i rezultata obrade [3]. Prvu grupu čine metode čiji su ulazni i izlazni podaci slike, dok drugu grupu čine metode čiji ulazni podatak može biti slika, ali izlazni podaci su neki atributi izvedeni sa slike. U prvu grupu spadaju formiranje slika, poboljšanje, restauracija, obrada slika u boji, obrada pomoću talasića, kompresija, itd. U drugu grupu spadaju segmentacija, reprezentacija, prepoznavanje objekata. Morfološka obrada za izlaz može imati sliku ali i neke atribute, pa ona kao takva pripada i prvoj i drugoj grupi.

3.3. Pretprocesiranje slike

Pretprocesiranje slike predstavlja pripremu slike za obradu. Sastoji se od različitih tehnika manipulacije nad ulaznom slikom kao što su skaliranje, poboljšanje kontrasta, detekcija ivica, segmentacija, otklanjanje šuma i sl. Ovaj korak predstavlja ključnu operaciju u mašinskom učenju, jer su ovdje izražene najvažnije osobine slike, dok su one manje bitne informacije i šum potisnute ili eliminisane.

Nova slika koja nastaje kao rezultat pretprocesiranja je slicna originalnoj slici, a razlikuju se po tome što nova slika ima npr. manje šuma, bolji kontrast.

3.4. Segmentacija slike

Jedan od glavnih elemenata u prepozavanju oblika na slici predstavlja pronalaženje ključnih elemenata slike koji reprezentativno predstavljaju njen sadržaj. Nakon što se ključni elementi pronađu stvaraju se uslovi da se ti elementi porede, klasifikuju i na drugi način koriste u daljoj obradi. Zadatak pronalaženja ovih elemenata ima segmenatacija koja dijeli sliku na regione, ili objekte do određenog nivoa i predstavlja nezaobilazan korak pri automastkoj detekciji oblika i analizi slike.

Thresholding ili segmentacija pragom predstavlja izdvajanje objekata od pozadine pomoću jednog ili više pragova. Segmentacija sa jednim pragom predstavlja najjednostavniju metodu i zasniva se na tome da ukoliko je vrijednost intenziteta piksela veća od praga, onda se on svrstava u pozadinu, a ukoliko je vrijednost manja od praga, onda se svrstava u objekat.

3.5. Histogram

Histogram koristimo kada je potrebno da dobijemo informaciju o distribuciji osvetljenosti piksela. Vrlo je koristan kada je potrebno odrediti prag za globalni threshold.
Histogram je grafik koji prikazuje broj piksela na slici pri svakoj različitoj vrijednosti intenziteta pronađenoj na toj slici.

3.6. Morfološke operacije

Kada se izoluje oblik kao binarna slika, u našem slučaju ručno ili mašinski pisan broj, oblici često imaju nesavršenosti i dolaze sa nepoželjnim šumovima i teksturama. Tada se koriste morfološke operacije i filteri kako bi se te nesavršenosti uklonile i dobio tačniji i precizniji oblik za dalju obradu. U morfološkim operacijama, vrijednost svakog piksela rezultujuće slike se zasniva na poređenju odgovarajućeg piksela na originalnoj slici sa svojom okolinom. Osnovne morfološke operacije su: erozija, dilacija, otvaranje, zatvaranje, istanjivanje i podebljavanje.

4. PROGRAMSKO RJEŠENJE

U ovom poglavlju će biti analizirani i izloženi rezultati prepoznavanja ručno pisanih brojeva koristeći vještačke neuronske mreže. Početni dio poglavlja posvećen je upoznavanju s programskim jezikom Python i pripadajućim programskim bibliotekama TensorFlow, NumPy, Keras i Matplotlib.

4.1. Python, TensorFlow, Keras, NumPy i Matplotlib

Python je interpretirani, interaktivni, objektno orijentisani programski jezik visokog nivoa i opšte namjene koji poseduje dinamičku sematiku. Python je „open source“, tako da je dopuštena distribucija i izmjena, što znači da ljudi širom svijeta mogu učestvovati u razvoju Python-a. Upravo takva fleksibilnost i otvorenost koda je doprinijela je rastućoj popularnosti Pythona u industrijskom i komercijalnom okruženju.

TensorFlow je biblioteka za numerička izračunavanja korišćenjem grafova toka podataka [4]. Među zapaženim primjenama su primjene u zadacima kao što su prepoznavanje govora, robotika, pronalaženje informacija, obrada jezika, pronalaženje geografskih informacija.

Osim TensorFlow-a, u izradi programskog koda korištena je i biblioteka NumPy. U njoj se nalaze esencijalne matematičke funkcije korištene u linearnoj algebri, razne transformacije, mogućnost stvaranja nasumičnih vrijednosti te naravno mogućnost rada s matricama.

Uz NumPy, važno je spomenuti i biblioteku Matplotlib, koja omogućava jednostavnu vizualizaciju podataka. Modul pyplot iz spomenute biblioteke je korišćen kod prikazivanja metrika učenja mreža korišćenih u programskoj podršci.

Baze podataka korištene u ovom radu za učenje i treniranje vještačkih neuronskih mreža su MNIST (*eng. Modified National Institute of Standards and Technology*) [5] i EMNIST (*eng. Extended Modified National Institute of Standards and Technology*).

4.2. Programska implementacija i rezultati

Prvo što je potrebno uraditi jeste video podijeliti na frejmove, da bi se potom izvršila transformacija u binarnu sliku, nakon čega će se izvršiti operacija erozije. Detekcija linije vrši se korišćenjem Hough transformacije. Iskorišćena je HoughLinesP transformacija iz OpenCV biblioteke.

Nakon toga potrebno je frejm po frejm izdvajati iz videa, pretvoriti svaki frejm u sliku u nijansama sive, a potom u binarnu sliku. Binarna slika se nakon toga kao parametar prosljeđuje u funkciju za selektovanje regiona tj. ekstrakciju cifara. Za svaki region je potrebno napraviti posebnu sliku dimenzija 28x28.

Za označavanje regiona korišćena je metoda cv2.boundingRect(contour). Da bi se označili samo oni regioni od interesa oko brojeva koji pređu preko linije, potrebno je uz pomoć koordinata dobijenih pomoću Hough transformacije iskoristi jednostavan algoritam za njihovo izdvajanje. Taj algoritam se ogleda u tome da se izdračunava udaljenost od gornje lijeve tačke konture broja do krajnjih tačaka detektovane linije.

Ukoliko je suma te dvije udaljenosti približna dužini linije (e=0.1) uzima se da je broj prešao liniju.

Na osnovu konture broja se vrši isijecanje regiona slike u kome se broj nalazi. Posle označavanja regiona i njihovog smještanja u niz slika koje predstavljaju regione sortirane po rastućoj vrijednosti x ose. Dobijeni region se proslijeđuje neuronskoj mreži, obučenoj pomoću MNIST data seta, na prepoznavanje.

Za obučavanje neuronske mreže, potrebno je obezbijediti ulaze, a potom željene izlaze za date ulaze. Nakon toga slijedi definisanje parametara algoritma za obučavanje i obučavanje neuronske mreže. Postignuta tačnost ovog programskog rješenja je 92%.

5. ZAKLJUČAK

U okviru ovog rada opisan je problem ekstrakcije, obrade i prepoznavanja objekata iz kontinualnog multimedijalnog strima. Posebna pažnja posvećena je manupulaciji nad slikama tj. frejmovima koji su izdvojeni iz videa i prepoznavanju ručno ili mašinski pisanih cifara uz pomoć vještačkih neuronskih mreža.

Prije izrade samog rješenja proučeni su mnogi načini kojima bi mogla da se implementira prethodno opisana problematika. Nakon detaljno proučenih mogućnosti pristupilo se izradi projektnog rješenja koje je pisano u programskom jeziku Python uz oslonac na pripadajuće bibliotke kao što su TensorFlow, NumPy, Keras i druge. Korišćene su neke od najpoznatijih metoda za obradu slike kao što su invertovanje, konvertovanje, globalni i lokalni threshold, histogram, morfološke operacije.

Nastavak razvijanja i poboljšavanja rješenja ovog problema mogao bi teći u smjeru ekstrakcije i prepoznavanja preklapajućih cifara iz multimedijalnog strima, kao i smanjenju uticaja faktora okoline na tačnost prepoznavanja kvantiteta.

6. LITERATURA

[1] Gurney K., „An introduction to neural networks“, University of Sheffield, London and New York, 1997.

[2] Matthew D Zeiler, Rob Fergus, „Visualizing and understanding convolutional networks“, 2013, New York.

[3] R. C. Gonzalez, R. E. Woods, „Digital Image Processing“, Prentice Hall, 2008.

[4] Yuan Yu, Xiaoqiang Zheng. „Large-scale machine learning on heterogeneous systems“, 2015.

[5] http://yann.lecun.com/exdb/mnist/ (pristupljeno u oktobru 2020.)

Kratka biografija:

Nikola Slijepčević rođen je u Trebinju 1996. god. Završio je gimnaziju u Gacku 2015. godine. Diplomski rad na Fakultetu Tehničkih Nauka u Novom Sadu odbranio je 2019. godine. Iste godine je upisao master studije u Novom Sadu na smjeru Računarstvo i automatika.

kontakt:
nikolaslijepcevi081296@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.39
DOI: https://doi.org/10.24867/12BE12Trifkovic

RAZVOJ APLIKACIJE ZA PRAĆENJE BILANSA MAKRONUTRIJENATA

DEVELOPMENT OF MACRONUTRIENT BALANCE MONITORING APPLICATION

Pavle Trifković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je opisan problem obračunavanja i praćenja bilansa kalorija i drugih makronutrijenata. Programsko rešenje ovog problema je implementirano u vidu dve aplikacije. Android aplikacija sadrži čitav skup funkcionalnosti, dok je serverska strana implementirana u vidu Spring aplikacije. Obračunavanje kalorijskog bilansa vrši se kroz evidentiranje obroka i aktivnosti. Bilans kalorija i drugih makronutrijenata računa se na dnevnom nivou.*

Ključne reči: *Kalorija, makronutrijent, aktivnost, obrok*

Abstract – *This paper describes the problem of calculating and monitoring the balance of calories and other macronutrients. The software solution to this problem is implemented in the form of two applications. The Android application contains a whole set of functionalities, while the server side is implemented in the form of a Spring application. Calculation of calorie balance is done through recording meals and activities. The balance of calories and other macronutrients is calculated on a daily basis.*

Keywords: *Calories, macronutrient, activity, meal*

1. UVOD

Postoji ogromna motivacija za kreiranje jedne ovakve aplikacije. Količina unesenih kalorija a takođe i pojedinačnih makronutrijenata poput masti, proteina, ugljenih hidrata, kao i vreme provedeno u nekoj fizičkoj aktivnosti, dominantno određuje kvalitet našeg života. Kalorije, u najkraćem predstavljaju meru za količinu energije u hrani. Podatak o tome koliko kalorija sadrži jedan obrok, može pomoći da se uravnoteži energija koja se unosi sa energijom koja se koristi, što predstavlja ključ za zdravu telesnu težinu.

S obzirom da ishrana predstavlja najbitniju stvar u kontrolisanju kvaliteta života, nešto više pažnje je posvećeno evidentiranju kalorijskih vrednosti, vrednosti makronutrijenata, kreiranju obroka radi preciznijeg računanja dnevnog bilansa, kreiranju pojedinačnih namirnica i mogućnošću njihovog dodavanja u obrok itd. Pošto bi kontrolisanje ovih vrednosti bez korišćenja jedne ovakve aplikacije bilo veoma naporno, ova aplikacija čitav posao završava u kratkom periodu, pa se čitav posao svodi na kreiranje obroka i evidentiranje aktivnosti. Čitav proračun i detaljni prikaz informacija, obavlja sama aplikacija.

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

2. RAZVOJNA OKRUŽENJA I TEHNOLOGIJE

U ovom poglavlju će biti pomenuta razvojna okruženja i tehnologije koje su korišćene pri izradi ove aplikacije.

2.1. Android

Android je mobilni operativni sistem kompanije Google zasnovan na Linux jezgru. Dizajniran je pre svega za mobilne uređaje sa ekranom osetljivim na dodir. Korisnički interfejs Androida je zasnovan na direktnoj manipulaciji objektima na ekranu, korišćenjem ulaza u vidu dodira. Aplikacije za Android se razvijaju u Java programskom jeziku, korišćenjem Android razvojnog paketa.

2.2. Android Studio

Android Studio je zvanično Android razvojno okruženje. Namenjeno je prvenstveno za razvoj Android mobilnih aplikacija. Predstavlja optimalno okruženje koje pruža sve mogućnosti trenutno dostupne u ovoj grani programiranja.Android Studio pruža i više nego dovoljno opcija za neometani razvoj Android aplikacija uključujući podešavanja različitih perspektiva za prikaz strukture projekta, integraciju sa različitim kontrolama koda itd [1].

2.3. Eclipse

Eclipse je integrisano razvojno okruženje koje sadrži osnovni radni prostor i proširivi plug-in sistem za prilagođavanje okruženja. Pisan je uglavnom na Javi i njegova primarna upotreba je za razvoj Java aplikacija, ali se može koristiti i za razvoj aplikacija na drugim programskim jezicima. Predstavlja programsko okruženje otvorenog koda, koje je nezavisno od operativnog sistema i koristi se kao pomoć pri izradi složenih aplikacija.

2.4 Java programski jezik

Java je objektno orijentisani programski jezik. U okviru Java programskog jezika uveden je koncept paketa koji se oslanjaju na fajl sistem, kao i koncept klasa iz objektno-orijentisane paradigme. Java je zvanično podržan jezik za izradu mobilnih aplikacija za Android uređaje. Prilikom izrade serverske aplikacije, korišćen je Java Spring framework, kao open-source Java platforma sa izuzetno kvalitetnom podrškom za izradu aplikacija. Spring sadrži veliki broj modula i projekata koji daju velike mogućnosti u razvoju enterprise aplikacija. Jedna od stvari koje Spring jako dobro obavlja jeste apstrahovanje programskih koncepata. Gotovo da ne postoji oblast koju ovaj framework ne obuhvata svojim bogatim skupom modula i projekata.

2.5 MySql baza podataka

MySQL je najpopularniji sistem otvorenog koda za upravljanje bazama podataka. Jedna od definicija MySQL-a je da predstavlja sistem za upravljanje relacionim bazama podataka. Kada se kaže da je MySQL softver otvorenog koda, to znači da svako može da ga koristi i

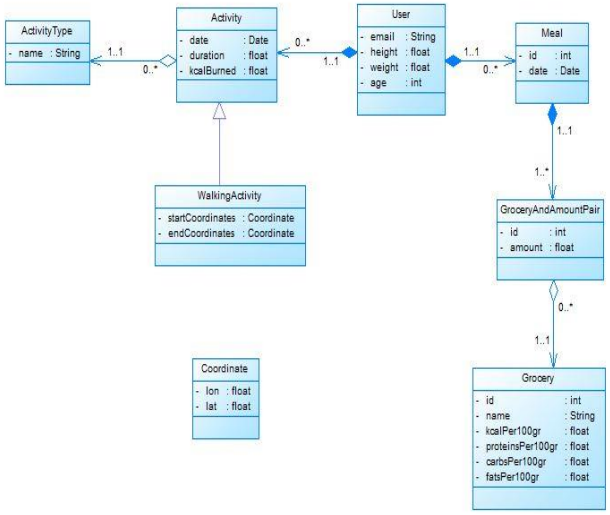
modifikuje.MySQL je veoma brz, pouzdan i jednostavan za korišćenje. Prilikom izrade aplikacije, podaci na serverskoj strani čuvani su u okviru MySQL baze podataka.

2.6 SQLite baza podataka

U okviru Android aplikacije, podaci su čuvani u SQLite bazi podataka. SQLite je open source biblioteka koja omogućava rad sa relacionim bazama podataka. Male je veličine i laka je za podešavanje, te joj je najčešća primena kao lokalno skladište za ugrađene uređaje i pametne telefone. Kao jedan od primera može se navesti upravo Google-ov Android operativni sistem za mobilne telefone koji sadrži upravo SQLite.

3. MODEL PODATAKA

Model podataka je predstavljen u vidu class dijagrama. Svaki bitan element aplikacije predstavljen je u vidu jednog entiteta. U okviru class dijagrama predstavljene su i veze između svih entiteta. Na slici 1 je prikazan model podataka aplikacije.



Slika 1. *Model podataka aplikacije*

Kao što je na slici 1 prikazano, svaki entitet sadrži polja koja ga opisuju kao i veze ka drugim entitetima. Na primeru entiteta User i Meal, ukratko će biti objašnjene veze. Prema modelu podataka sa slike 1, svaki entitet User može da bude u vezi sa 0 ili više entiteta Meal, dok svaki entitet Meal mora biti u vezi sa samo jednim entitetom User. U praksi to znači da svaki korisnik može da kreira više obroka a ne mora nijedan, dok jedan konkretan obrok mora biti kreiran od strane samo jednog korisnika. Sve ostale veze između ostalih entiteta, u okviru modela podataka sa slike 1, funkcionišu po istom principu.

4. FUNKCIONALNOSTI APLIKACIJE

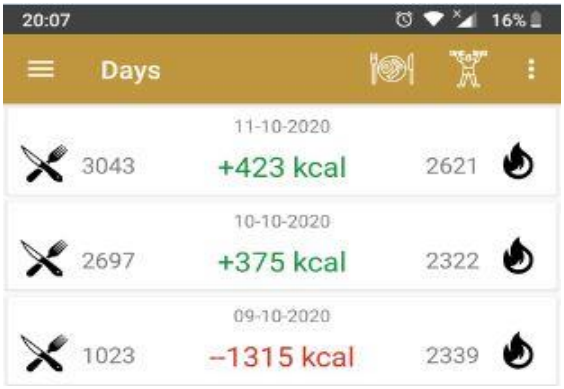
U ovom poglavlju će biti opisan skup funkcionalnosti koji ova aplikacija nudi.

4.1. Registracija i prijava korisnika

Pored osnovne funkcije registracije korisnika i unosa email-a i lozinke pomoću koje će se vršiti kasnija prijava na sistem i proces autentifikacije, u okviru forme za registraciju unose se još neki jako bitni parametri. Korisnik je obavezan da prilikom registracije unese informacije o svojoj visini i težini, a takođe je potrebno da unese i informaciju o svojim godinama i polu. Pomenuti podaci su jako bitni prilikom obračunavanja potrošnje kalorija. Kada se proces registracije uspešno završi, korisnik se vraća na početni ekran gde ima formu za prijavu na sistem.

4.2. Početni ekran aplikacije

Početni ekran aplikacije sadrži listu dana sa informacijama o ukupnom iznosu unetih i potrošenih kalorija u toku dana. Takođe, prikazan je ukupan kalorijski bilans za odgovarajući dan. Pored navedenih informacija, početni ekran sadrži i toolbar sa opcijom za dodavanje novog obroka, kao i opcijom za evidentiranje nove aktivnosti. Takođe, dostupne su i overflow opcija sa mogućnostima dodavanja nove namirnice i procesa sinhronizacije podataka, kao i opcija za navigaciju upotrebom Navigation Drawer-a. Na slici 2 je prikazan deo početnog ekrana.



Slika 2. *Deo početnog ekrana aplikacije*

4.3. Kreiranje nove namirnice

Nakon procesa registracije i prijave na sistem, korisnik ima pristup opciji za kreiranje novog obroka na osnovu namirnica koje već postoje u bazi podataka. Međutim, pošto taj skup namirnica nije konačan, korisniku je omogućena opcija da u slučaju da neka namirnica ne postoji, on može sam da je kreira. Na taj način, njegov obrok u aplikaciji bi u potpunosti odgovarao stvarnom obroku. Prilikom kreiranja namirnice, neophodno je uneti podatke o nazivu namirnice, broju kalorija na 100 grama, kao i količini proteina, masti i ugljenih hidrata na 100 grama. S obzirom da dodavanjem nove namirnice u bazu podataka ona postaje dostupna i u procesu kreiranja obroka, neophodno je da sadrži sva navedena polja kako bi aplikacija mogla da izvrši proračun svih nutritivnih vrednosti za svaki obrok.

4.4. Kreiranje novog obroka

Kreiranje novog obroka predstavlja jednu od glavnih i najvažnijih funkcionalnosti ove aplikacije. Kreiranjem novog obroka, a na osnovu informacija o namirnicama koje sačinjavaju dati obrok, aplikacija računa ukupan kalorijski bilans, kao i bilans makronutrijenata za kreirani obrok. Korisnik u toku jednog dana ima mogućnost kreiranja neograničenog broja obroka, gde se za svaki od obroka posebno obračunava kalorijski bilans. Prvi korak u procesu kreiranja novog obroka je odabir željene namirnice. Za svaku namirnicu postoji informacija o svim

nutritivnim vrednostima. Korisnik ima mogućnost da odabirom željene namirnice, unese iznos u kojem namirnica učestvuje u obroku. Tokom kreiranja obroka, korisnik u svakom trenutku ima mogućnost uvida u trenutno stanje kreiranog obroka.

4.5. Evidentiranje aktivnosti

Evidentiranje aktivnosti je takođe jedna od bitnijih funkcionalnosti. Evidentiranjem neke od aktivnosti, na osnovu podataka o njoj kao i drugih bitnih podataka, računa se potrošnja kalorija pri obavljanju date aktivnosti. Korisnik iz liste dostupnih aktivnosti bira aktivnost koju želi da evidentira i prilikom samog evidentiranja, unosi vreme trajanja aktivnosti. Aplikacija nudi veoma širok skup aktivnosti.

Jedan od načina za evidentiranje je onaj gde korisnik može da evidentira aktivnost iz liste aktivnosti unošenjem vremena trajanja.Svaka od aktivnosti iz liste sadrži svoju MET vrednost koja predstavlja odnos radne brzine metabolizma u odnosu na brzinu metabolizma u stanju mirovanja. MET vrednost se koristi prilikom računanja broja potrošenih kalorija.

Drugi način je evidentiranje aktivnosti od strane aplikacije. Aplikacija sama detektuje kada je korisnik započeo kretanje i kada je korisnik završio kretanje. Na osnovu vremena koje je proteklo između ta dva trenutka, aplikacija obračunava broj potrošenih kalorija.

Za detekciju kretanja, koristi se Activity Recognition API, koji automatski otkriva aktivnosti periodičnim čitanjem kratkih nizova podataka senzora i njihovom obradom pomoću modela mašinskog učenja [2].

Treći način svodi se na automatsko evidentiranje aktivnosti od strane aplikacije. Aktivnost koja se beleži naziva se bazalni metabolizam. Bazalni metabolizam je broj koji govori koliko organizam čoveka troši kalorija u toku dana dok miruje i računa se po posebnoj formuli koja je različita za žene i muškarce [3]. Aplikacija evidentira bazalni metabolizam za svaki novi dan.

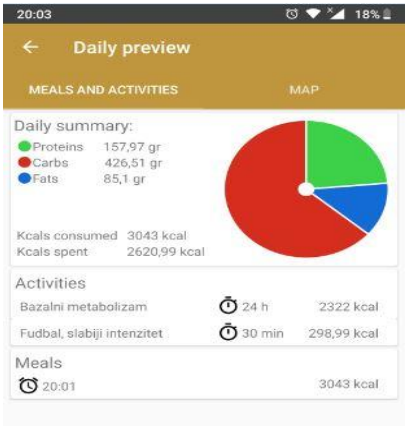
4.6. Informacije o dnevnom bilansu nutritivnih vrednosti

Osnovne informacije o dnevnom kalorijskom bilansu nudi početni ekran. Međutim, aplikacija omogućava detaljniji prikaz informacija o kalorijskom bilansu za svaki dan, kao i o bilansu svih ostalih makronutrijenata.

Ukoliko korisnik želi da vidi detaljnije informacije o kreiranim obrocima u toku dana i detaljnom obračunu kalorija za svaki od njih, kao i o evidentiranim aktivnostima i potrošenim kalorijama, potrebno je da odabere željeni dan iz liste dana. Klikom na željeni dan, korisniku se otvara ekran koji sadrži dva taba. Na slici 3, prikazan je ekran koji se otvara klikom na željeni dan iz liste dana.

Kao što se na slici 3 može videti, tab MEALS AND ACTIVITIES sadrži detaljne informacije za odabrani dan, i podeljen je na tri celine. U vrhu taba su prikazani detaljni podaci o ukupnom kalorijskom bilansu za odabrani dan. Daily summary daje informaciju o vrednostima makronutrijenata unesenih kroz obroke. Takođe, ovaj tab nudi i grafički prikaz raspodele makronutrijenata.

Druga celina obuhvata informacije o evidentiranim aktivnostima sa informacijama o tipu aktivnosti, trajanju aktivnosti i broju potrošenih kalorija.



Slika 3. *Ekran sa podacima o obroku i aktivnostima*

Treća celina sadrži informacije o svim kreiranim obrocima u okviru odabranog dana. Za svaki obrok prikazane su informacije o vremenu kreiranja i broju unesenih kalorija. Međutim, pored ovih informacija, korisnik ima mogućnost da odabirom nekog obroka iz liste dobije detaljan prikaz informacija o odabranom obroku. Na slici 4, prikazan je ekran koji se otvara klikom na neki obrok.

Slika 4. *Ekran sa detaljima o obroku*

Kao što se na slici 4 može videti, ekran je podeljen na dve celine, od kojih prva celina prikazuje raspodelu makronutrijenata u vidu grafika i liste, dok druga prikazuje detaljne informacije o svakoj namirnica koja sačinjava obrok. Na kraju je bitno pomenuti i tab MAP sa slike 3, koji korisniku omogućava prikaz njegovog kretanja na mapi. Za dobavljanje lokacije korisnika, korišćen je Google Fused Location Provider.

4.7. Grafički prikaz bilansa nutritivnih vrednosti

Koristeći opciju za grafički prikaz bilansa nutritivnih vrednosti, korisnik ima mogućnost prikaza količine određenog makronutrijenta u datumskom opsegu koji sam može da definiše. Korisniku se ostavlja mogućnost da definisanjem početnog i krajnjeg datuma vremenskog intervala za koji želi da prikaže informacije, dobije grafički prikaz za svaki dan iz odabranog intervala. Takođe, korisnik u svakom trenutku može da promeni makronutrijent koji želi da prikaže u odabranom datumskom opsegu.

4.8. Izmena profila korisnika

Korisniku aplikacija nudi mogućnost izmene ličnih podataka unesenih kroz proces registracije. Izmenu visine, težine i godina korisnika je veoma bitno menjati blago-

vremeno, kako bi korišćenje aplikacije bilo što kvalitetnije. Prethodno je pomenuto da količina potrošenih kalorija upražnjavanjem bilo koje fizičke aktivnosti između ostalog zavisi upravo i od visine, težine i godina koje je korisnik uneo ili ažurirao. Ukoliko se pomenuti podaci tokom korišćenja aplikacije promene, a korisnik to ne evidentira blagovremeno, rezultati kalorijskog bilansa neće biti sasvim precizni. Najveća odstupanja u tom pogledu nastaju automatskim evidentiranjem bazalnog metabolizma, jer u slučaju bazalnog metabolizma potrošnja kalorija dominantno zavisi upravo od pomenutih parametara.

4.9. Sinhronizacija podataka sa serverom

Glavni cilj implementiranja procesa sinhronizacije jeste mogućnost korišćenja aplikacije i u slučaju kada korisnik nije povezan na internet, a da se pri tome ti podaci nakon povezivanja na internet zabeleže na serverskoj strani. Kada se aplikacija koristi u offline režimu, odnosno kada korisnik nije povezan na internet, podaci se beleže u lokalnu SQLite bazu podataka, tako da korisnik prilikom svakog pokretanja aplikacije ima najnoviju verziju podataka. Nakon što se u centralnoj bazi naprave kopije podataka iz lokalne baze, poenta je da se izvrši sinhronizacija u smeru serverska aplikacija - android aplikacija. To obezbeđuje mogućnost da kada se korisnik uloguje na aplikaciju sa nekog drugog android uređaja a ukoliko je povezan na internet, može da u okviru aplikacije ima uvid u sva poslednja ažuriranja koja je izvršio na aplikaciji.

Dakle, prvo se vrši sinhronizacija u smeru android - server, gde se čuvaju podaci. Zatim u slučaju kada se korisnik prijavljuje na aplikaciju sa drugih uređaja može da inicira sinhronizaciju u kontra smeru kako bi mogao da koristi podatke koje je evidentirao koristeći aplikaciju na drugom android uređaju.

5. PRIMENA APLIKACIJE

S obzirom na sve češću upotrebu aplikacija ovog tipa, mogućnosti primene su veoma široke. Kao što je pomenuto, zdravstveno stanje čoveka a samim tim i kvalitet njegovog života zavisi od vremena provedenog u upražnjavanju neke fizičke aktivnosti, kao i od onoga čime i u kojoj se meri hrani. Iz tog razloga, ova aplikacija bi mogla da bude od pomoći svakome ko želi detaljno da prati pomenute parametre.

U određenoj meri ova aplikacija može biti i edukativnog karaktera u smislu upoznavanja sa nutritivnim vrednostima pojedinih namirnica, kako bi svaki čovek mogao da na osnovu stečenog znanja sebi osmisli obroke u toku dana. S obzirom da aplikacija nudi širok skup funkcionalnosti i mogućnosti za evidentiranje obroka i aktivnosti, može se reći da postiže veoma precizno obračunavanje bilansa makronutrijenata.

6. ZAKLJUČAK

U radu je opisan problem obračunavanja kalorijskog bilansa i bilansa ostalih makronutrijenata na dnevnom nivou, kao i aplikacija koja rešava taj problem.

Motivacija za kreiranje jedne ovakve aplikacije leži u činjenici da njena primena u mnogome može uticati na kvalitet života svakog čoveka.

Aplikacija je razvijana za Android operativni sistem u Android Studio-u, kao zvaničnom okruženju za razvijanje Android aplikacija. Celokupan sistem je podeljen u dve aplikacije, jednu serversku i jednu Android aplikaciju. Serverska aplikacija je implementirana kao SpringBoot aplikacija i koristi se za čuvanje podataka unesenih kroz Android aplikaciju kroz proces sinhronizacije baza podataka.

Kao što je već pomenuto, dve osnovne funkcionalnosti na kojima se ova aplikacija zasniva su mogućnost kreiranja obroka i mogućnost evidentiranja aktivnosti. S obzirom da je osnovni cilj prilikom razvijanja ove aplikacije bio što precizniji i efikasniji način obračuna kalorijskog bilansa, obezbeđena je opcija za kreiranje nove namirnice.

Na taj način se pri kreiranju obroka obezbeđuje precizniji obračun unesenih kalorija. Takođe, u cilju preciznosti su obezbeđene opcije za evidentiranje širokog skupa aktivnosti.

7. LITERATURA

[1] https://developer.android.com/studio (pristupljeno u oktobru 2020.)

[2] https://developers.google.com/location-context/activity-recognition (pristupljeno u oktobru 2020.)

[3] https://fitnessctt.com/kalorije-dnevna-potrosnja/ (pristupljeno u oktobru 2020.)

Kratka biografija:

Pavle Trifković rođen je 1996. godine u Aranđelovcu. Završio je elektrotehničku školu „Mihajlo Pupin“ u Novom Sadu 2015. godine. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu 2019. godine. Master studije je upisao 2019. godine na smeru Računarstvo i automatika.

kontakt: trifko96@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.85
DOI: https://doi.org/10.24867/12BE13Zivolic

RUDARENJE PODATAKA I NAPREDNE ANALITIČKE TEORIJE I METODE

DATA MINING AND ADVANCED ANALYTICAL THEORY AND METHODS

Smiljana Živolić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu prikazan je kratak pregled Poslovne analitike sa akcentom na teorije i metode Rudarenja podataka. Predstavljena je osnovna istraživačka analiza podataka, Klaster analiza kao i Linearna i Logistička regresija.*

Ključne reči: *Rudarenje podataka, poslovna analitika, algoritmi, statistika*

Abstract – *This paper presents a brief overview of Business Analytics with an emphasis on theories and methods of Data Mining. Basic research data analysis, Cluster analysis as well as Linear and Logistic regression are presented.*

Keywords: *Data Mining, Business Analitics, Algorithms, Statistics*

1. UVOD

Podaci su takoreći novo zlato i njegovo iskopavanje radi stvaranja poslovne vrednosti u današnjem kontekstu izuzetno umreženog i digitalnog društva zahteva skup veština koji tradicionalno nismo primenjivali u poslu ili u statistici ili čak u inženjerskim programima.

Ideja da senzorima možemo da povežemo fizičke objekte kao što su domovi, automobili, putevi, čak i kante za smeće i ulična svetla, sa digitalno optimizovanim sistemom upravljanja ide ruku pod ruku sa većim podacima i potrebom za dubljim analitičkim mogućnostima.

Nismo daleko od pametnog frižidera koji će osetiti da vam nedostaje, recimo, jaja, popuniti listu za kupovinu mobilne aplikacije vaše prehrambene prodavnice i organizovati Task Rabbit-a da vam pripremi trgovinu namirnicama. Ili frižider koji pregovara o dogovoru sa vozačem Ubera da vam isporuči večernji obrok. Niti smo daleko od senzora ugrađenih u puteve i vozila koji mogu izračunati zagušenost saobraćaja, pratiti istrošenost kolovoza, evidentirati upotrebu vozila i faktorirati ih na dinamičke cene zasnovane na upotrebi, stope osiguranja, pa čak i oporezivanje. Ovaj hrabri novi svet podstaknuće analitika i sposobnost da se podaci iskoriste za konkurentsku prednost.

Poslovna analitika je disciplina u nastajanju koja će zasigurno pomoći da idemo u korak sa ovim novim talasom.

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

Ovaj rad usmeren je na Poslovnu analitiku, njene delove kao i savremenu primenu u različitim sferama današnjice.

2. POSLOVNA ANALITIKA

Poslovna analitika je proces prikupljanja, sortiranja, obrade i proučavanja poslovnih podataka i korišćenje statističkih modela i iterativnih metodologija za korišćenje podataka u poslovne uvide.

Cilj poslovne analitike jeste da utvrdi koji su skupovi podataka korisni i kako se mogu iskoristiti za rešavanje problema i povećanje efikasnosti, produktivnosti i prihoda poslovanja.

Kao podskup poslovne inteligencije (*eng. Business Inteligence*), poslovna analitika se generalno primenjuje s ciljem identifikovanja podataka koji se mogu primeniti u određenu svrhu.

Poslovna inteligencija je obično opisna, usredsređena je na strategije i alate koji se koriste za prikupljanje, identifikovanje i kategorizaciju sirovih podataka i izveštavanje o prošlim ili trenutnim događajima.

Poslovna analitika je više propisana, posvećena je metodologiji pomoću koje se podaci mogu analizirati, prepoznati obrasci i modeli razvijati kako bi se razjasnili prošli događaji, stvorile predviđanja za buduće događaje i preporučile akcije za maksimalizovanje idealnih ishoda.

Sofisticirani podaci, kvantitativna analiza i matematički modeli koriste se od strane poslovnih analitičara da bi dizajnirali rešenja za probleme vođene podacima. Oni mogu da koriste statistiku, informacione sisteme, računarsku nauku i operativna istraživanja kako bi proširili svoje razumevanje složenih skupova podataka i veštačke inteligencije, dubokog učenja (*eng. Deep Learning*) i neuronskih mreža kako bi mikrosegmentisali dostupne podatke i identifikovali obrasce.

Ove informacije se zatim mogu iskoristiti za tačno predviđanje budućih događaja povezanih sa delovanjem potrošača ili tržišnim trendovima i za preporučivanje koraka koji potrošače mogu usmeriti ka željenom cilju.

Poslovna analitika se može posmatrati kroz niz kategorija i komponenti koje svako analitičko rešenje ima. U teoriji se pojavljuje osam osnovnih kategorija:

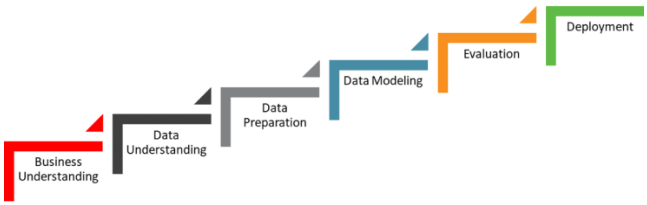
- Agregacija podataka
- Rudarenje podataka
- Asocijacija I identifikacija sekvence
- Rudarenje teksta
- Predviđanje
- Prediktivna analitika
- Vizualizacija podataka

3. RUDARENJE PODATAKA

Šta je Data Mining? Kakva je veza između Big Data i Data Mining-a? Zašto je rudarenje podataka danas veoma važno? To su sve pitanja sa kojima se danas sve češće susrećemo. Odgovori na njih biće dati u ovom i u narednim poglavljima.

Data Mining je interdisciplinarno polje koje zahteva znanje iz mašinskog učenja, veštačke inteligencije i matematičke statistike za pronalaženje i izdvajanje obrazaca iz skupova podataka koji prevazilazi mogućnosti SQL jezika.

Rudarenje podataja bavi se otkrivanjem znanja i pronalaženjem obrazaca u skupovima podataka kroz proces primene modela na podatke . Model, srž procesa rudarstva podataka, je tehnika i algoritmi koji se primenjuju na podatke za pronalaženje sličnosti, obrazaca i grupisanje podataka. Uobičajeni modeli pretraživanja podataka su klasifikacija, pravila pridruživanja i klasterizacija. U ovom istraživanju fokus je na grupisanju i pronalaženju čestih skupova predmeta. Rudarstvo podataka ima širok spektar primena u nauci i inženjerstvu. Na primer, u prognozi vremena, model klasifikacije se može koristiti za predviđanje vremena za sledeći dan na osnovu prethodnih podataka. Drugi primer je za predloge filmova u sistemima koji preporučuju.Na osnovu nekih korisničkih preferencija, mogu se predvideti drugi filmovi koje korisnik preferira. Algoritmi klaster modela takođe imaju širok spektar aplikacija kao što su dovršavanje scene, sažimanje podataka i tehnike oporavka podataka. Na kraju, česti set predmeta se široko koristi u maloprodajnim sistemima za predviđanje kupaca u kupovnim navikama, kao i u dizajnu kataloga.



Slika 1. Šest koraka Rudarenja podataka [1]

3.1. Izazovi u Rudarenju podataka

Iako je moćan proces, rudarenje podataka ometaju sve veća količina i složenost velikih podataka (*eng. Big data*). Tamo gde firme svakodnevno prikupljaju egzabajte podataka, donosioci odluka trebaju načine za izdvajanje, analizu i sticanje uvida iz svog obilnog spremišta podataka.

Izazovi velikih podataka su plodni i prodiru u svako polje koje prikuplja, skladišti i analizira podatke. Veliki podaci karakterišu četiri glavna izazova: obim, raznolikost, istinitost i brzina. Cilj rudarenja podacima je da posreduje u ovim izazovima i otkrije vrednost podataka.

Kako brzina podataka nastavlja da povećava obim i raznolikost podataka, preduzeća moraju da skaliraju ove modele i primenjuju ih u celoj organizaciji. Omogućavanje punih prednosti rudarenja podataka ovim modelima zahteva značajna ulaganja u računarsku infrastrukturu i procesorsku snagu. Da bi dostigle razmere, organizacije moraju da kupe i održavaju moćne računare, servere i softver dizajnirane da obrađuju veliku količinu i raznolikost podataka kompanije.

Povećani zahtevi za skladištenjem podataka primorali su mnoge kompanije da se okrenu računarstvu i cloud skladištenju. Iako je cloud osnažio mnoga savremena dostignuća u rudarstvu podataka, priroda usluge stvara značajne pretnje privatnosti i bezbednosti. Organizacije moraju zaštititi svoje podatke od zlonamernih podataka da bi održale poverenje svojih partnera i kupaca.

3.2. Vrste Rudarenja podataka

Rudarenje podataka ima dva osnovna procesa: učenje pod nadzorom (nadgledano) i bez nadzora (nenadgledano).

Nadgledno učenje

Cilj učenja pod nadzorom je predviđanje ili klasifikacija. Najlakši način da se konceptualizuje ovaj proces je traženje jedne izlazne promenljive. Proces se smatra učenjem pod nadzorom ako je cilj modela predviđanje vrednosti posmatranja.

Jedan od primera su filteri za neželjenu poštu, koji koriste nadzirano učenje za klasifikaciju dolaznih e-adresa kao neželjenog sadržaja i automatsko uklanjanje ovih poruka iz prijemnog sandučeta.

Uobičajeni analitički modeli koji se koriste u pristupima nadgledanog rudarenja podataka su:

Regresije

- ***Vremenske serije*** - Modeli vremenskih serija su alati za predviđanje koji koriste vreme kao primarnu nezavisnu promenljivu. Trgovci na malo, kao što je Maci's, primenjuju modele vremenskih serija da bi predvideli potražnju za proizvodima u funkciji vremena i koristili predviđanje za tačno planiranje i skladištenje zaliha sa potrebnim nivoom zaliha.
- ***Klasifikacija*** - Klasifikaciono drveće je tehnika prediktivnog modeliranja koja se može koristiti za predviđanje vrednosti i kategoričkih i kontinuiranih ciljnih promenljivih. Na osnovu podataka, model će stvoriti skupove binarnih pravila za razdvajanje i grupisanje najvećeg udela sličnih ciljnih promenljivih. Poštujući ta pravila, grupa u koju spada novo zapažanje postaće njena predviđena vrednost.
- ***Neuronske mreže*** - Neuronske mreže koriste ulaze i, na osnovu njihove veličine, „pucaće" ili „neće aktivirati" svoj čvor na osnovu zahteva praga. Ovaj signal ili njegov nedostatak se zatim kombinuje sa ostalim „ispaljenim" signalima u skrivenim slojevima mreže, gde se proces ponavlja sve dok se ne kreira izlaz.
- ***K-najbiliži sused***

Nenadgledano učenje

Zadaci bez nadzora se fokusiraju na razumevanje i opisivanje podataka kako bi se otkrili osnovni obrasci unutar njih. Sistemi preporuka koriste nenadgledano učenje kako bi pratili obrasce korisnika i pružali im personalizovane preporuke za poboljšanje korisničkog iskustva.

4. KLASTER ANALIZA

Generalno, klasterisanje je upotreba nenadgledanih tehnika za grupisanje sličnih objekata. U mašinskom učenju, nenadgledano se odnosi na problem pronalaženja skrivene strukture u neobeleženim podacima. Tehnike klasterovanja nisu pod nadzorom u smislu da naučnik unapred ne određuje labele koje primenjuje na klastere. Struktura podataka opisuje objekte od interesa i određuje kako najbolje da se objekti grupišu.

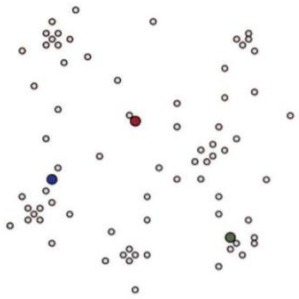
4.1. K-Means

S obzirom na kolekciju predmeta od kojih svaki ima *n* merljivih atributa, K-means je analitička tehnika koja, za izabranu vrednost *k*, identifikuje k klastera objekata na osnovu blizine objekata do centra k grupe. Centar se određuje kao aritmetički prosek (srednja vrednost) n-dimenzionalnog vektora svakog klastera atributa. Ovaj odeljak opisuje algoritam za određivanje k-vrednosti, kao i kako to najbolje primeniti tehniku na nekoliko slučajeva upotrebe.

Grupisanje se često koristi kao uvod u klasifikaciju. Jednom kada su klasteri identifikovani, mogu se primeniti oznake na svakom klasteru da klasifikuje svaku grupu na osnovu njenih karakteristika. Grupisanje je prvenstveno istraživačka tehnika otkrivanja skrivenih struktura podataka, možda kao uvod u fokusiraniju analizu ili procese odlučivanja. Neke specifične primene K-means-a su obrada slike, medicinska i korisnička segmentacija.

K-means algoritam za pronalaženje k klastera može se opisati u sledeća četiri koraka:

1. Izaberemo vrednost *k* i k početno za centroide. U ovom primeru, k = 3, a početni centroidi su označeni tačkama osenčenim crvenom, zelenom, i plavom na *slici 2*.



Slika 2. Korak 1 u pronalaženju K klastera [1]

2. Izračunavamo udaljenost od svake tačke podataka (x,y_i) do svakog centroida. Dodelimo svaku tačku najbližem centroidu. Ovo definiše prvih k klastera.
   U dve dimenzije, rastojanje d između bilo koje dve tačke, (x_1,y_1) i (x_2, y_2), u ravni se tipično izražava korišćenjem Euklidove mere daljine date u jednačini:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (1)$$

3. Izračunavamo centroid, centar mase svakog novo definisanog skupa iz koraka 2. Na *slici 10*. izračunati centroidi u koraku 3 su svetlo osenčene tačke odgovarajuće boje. U dve dimenzije, centroid (x_c,y_c) od m tačaka u k-means klasteru izračunava se na sledeći način u jednačini :

$$(x_c, y_c) = \left(\frac{\sum_{i=1}^{m} x_i}{m}, \frac{\sum_{i=1}^{m} y_i}{m}\right) \quad (2)$$

- Ponavljamo korake 2 i 3 dok se algoritam ne konvergira u odgovor.
- Dodeljujemo svaku tačku najbližem centoridu izračunatom u koraku 3.
- Izračunavamo centorid novodefinisanih klastera.
- Ponavljamo dok algoritam ne dođe do konačnog odgovora.

Konvergencija se postiže kada se izračunati centroidi ne promene ili kada centroidi i dodeljene tačke osciliraju napred-nazad od jedne do druge iteracije. Može doći do ovog drugog slučaja kada postoji jedna ili više tačaka na jednakim udaljenostima od izračunatog centorida.

5. REGRESIJA

Generalno, regresijska analiza pokušava da objasni uticaj koji skup promenljivih ima na ishod druge promenljive od interesa. Često se promenljiva ishoda naziva zavisnom promenljivom jer ishod zavisi od ostalih promenljivih. Ove dodatne promenljive se ponekad nazivaju ulazne promenljive ili nezavisne promenljive.

Linearna regresija je korisno sredstvo za odgovor na prvo pitanje, a logistička regresija je popularan metod za odgovor na drugo pitanja. Ovo poglavlje ispituje ove dve tehnike regresije i objašnjava kada je jedna tehnika pogodnija od druge.

Regresijska analiza korisno je objašnjenje koje može identifikovati ulazne promenljive koje imaju najveći statistički uticaj na ishod. Sa takvim znanjem i uvidom, promene životne sredine mogu se pokušati proizvesti povoljnije vrednosti ulaznih promenljivih. Na primer, ako se utvrdi da je nivo čitanja desetogodišnjaka odličan prediktor uspeha učenika u srednjoj školi i faktor njihovog pohađanja fakulteta, onda se može razmotriti, primeniti dodatni naglasak na čitanju i ocenjeno radi poboljšanja nivoa čitanja učenika u mlađem uzrastu.

5.1. Linearna regresija

Linearna regresija je analitička tehnika koja se koristi za modeliranje odnosa između nekoliko ulaznih promenljivih i kontinuirane promenljive ishoda. Ključna pretpostavka jeste da je veza između ulazne promenljive i promenljive ishoda linearna. Iako se ova pretpostavka može činiti restriktivnom, često je moguće pravilno transformisati ulazne ili ishodne promenljive kako bi se postigla linearna veza između izmenjenih promenljivih ulaznih i ishodnih.

Model linearne regresije pretpostavlja da postoji linearni odnos između ulaznih promenljivih i promenljive ishoda. Ovaj odnos se može izraziti kao što je prikazano u jednačini:

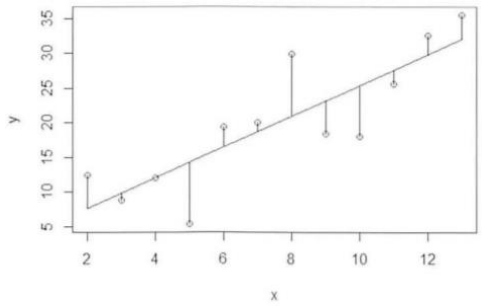
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{p-1} x_{p-1} + \xi \quad (3)$$

U linearnom modelu, β_j predstavljaju nepoznate *p* parametre. Procene za ove nepoznate parametre su odabrane tako da u proseku model pruža razumnu procenu prihoda osobe na osnovu starosti i obrazovanja. Drugim rečima, ugrađeni model treba da umanji ukupnu grešku između linearnog modela i stvarnih zapažanja.

Obični najmanji kvadrati su uobičajena tehnika za procenu parametara.
Da bismo ilustrovali kako ova tehnika radi, pretpostavimo da postoji samo jedna ulazna promenljiva, *x*, za izlaznu promenljivu *y*.
Cilj je pronaći liniju koja najbolje aproksimira odnos između izlazne promenljive i ulaznih promenljivih. Cilj najmanjih kvadrata je pronaći liniju kroz ove tačke koja minimalizuje broj kvadrata razlike između svake tačke i prave u vertikalnom smeru. Drugim rečima, pronađemo vrednosti -10 i.-1, tako da je zbir prikazan u sledećoj jednačini minimiziran.

$$\sum_{i=1}^{n} [y_i + (\beta_0 + \beta_1 x_1)]^2 \quad (4)$$

Na *slici 3.* prikazano je n pojedinačnih rastojanja koje treba kvadratiti, a zatim zbrojiti. Vertikala linije predstavljaju rastojanje između svake posmatrane vrednosti *y* i linije $y = \beta_0 + \beta_1 x$.



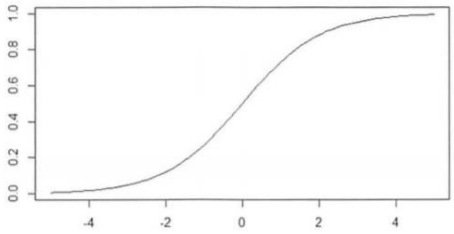
Slika 3.

5.2. Logistička regresija

Logistička regresija se zasniva na logističkoj funkciji f(y) datoj u formuli:

$$f(y) = \frac{e^y}{1+e^y}, \text{ za } -\infty < y < \infty \quad (5)$$

Primetimo da y → ∞, f(y) →1 i y → -∞, f(y) →0, tako da na *slici 4.* vidimo da vrednost logističke funkcije f(y) varira 0 do 1.



Slika 4.

Budući da je opseg f (y) u interval (0, 1), čini se da je logistička funkcija odgovarajuća funkcija za modeliranje verovatnoće da se dogodi određeni ishod. Kako se vrednost y povećava, verovatnoća ishoda se povećava. U bilo kojem predloženom modelu, da bi se predvidela verovatnoća ishoda, y treba da bude funkcija ulaznih promenljivih. U logističkoj regresiji y se izražava kao linearna funkcija ulaznih promenljivih.

6. ZAKLJUČAK

Praksa nauke o podacima može se najbolje opisati kao kombinacija analitičkog inženjerstva i istraživanja. Posao predstavlja problem koji bismo želeli da rešimo. Retko je poslovni problem jedan od naših osnovnih zadataka za rudarenje podacima. Razlažemo problem u podzadatke za koje mislimo da ih možemo rešiti, obično počevši od postojećih alata.

Za neke od ovih zadataka možda ne znamo koliko dobro možemo da ih rešimo, pa moramo da istražimo podatke i izvršimo procenu da bismo to videli. Ako to ne uspe, možda ćemo morati da pokušamo nešto sasvim drugo. U tom procesu možemo otkriti znanje koje će nam pomoći da rešimo problem koji smo zacrtali da rešimo ili ćemo otkriti nešto neočekivano što nas vodi do drugih važnih uspeha.

7. LITERATURA

[1] Galit Shmueli, Peter C. Bruce, Nitin R. Patel, Data Mining for Business Analitics, Third edition
[2] Foster Provost & Tom Fawcett, Data Science for Business
[3] T. H. Davenport and D. J. Patil, "Data Scientist: The Sexiest Job of the 21st Century," Harvard Business Review, October 2012.
[4] The R Project for Statistical Computing, "The Comprehensive R Archive Network."
[5] P.-N. Tan, V. Kumar, and M. Steinbach, Introduction to Data Mining, Upper Saddle River, NJ: Person, 2013.

Kratka biografija:

Smiljana Živolić rođena je u Novom Sadu 1993. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Primenjene računarske nauke i informatika odbranila je 2020.god.

kontakt:
smiljana.zivolic@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE14Panic

MODEL BATERIJE SA TEMPERATURNIM EFEKTOM I EFEKTOM STARENJA

BATTERY MODEL WITH TEMPERATURE AND AGING EFFECT

Nataša Panić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisan je matematički i simulacioni model litijum-jonske baterije u koji su uključeni temperaturni efekat i efekat starenja baterijeNa osnovu datog matematičkog modela, simulacioni model baterije pogodan za izvršenje u realnom vremenu je implementiran uz oslonac na softverski alat Typhoon HIL Control Center. Rezultati simulacije dobijenog modela sa dodatim efektima prikazani su grafički. Za testiranje i simulaciju modela korišćen je uređaj Typhoon HIL 402.*

Ključne reči: *model baterije, temperaturni efekat, efekat starenja, litijum-jonska baterija, simulacija u realnom vremenu, simulacija sa hardverom u petlji.*

Abstract –*This paper describes a model of a battery with temperature and aging effects that is simulated in real time with hardware-in-the-loop simulation. The mathematical model of the battery given in the paper is the basis of the battery model that includes both effects. Based on the given mathematical model, the battery was implemented in the Typhoon HIL Control Center with the signal processing tool in Schematic Editor. The simulations results of the obtained model with added effects are presented graphically. The device used for the testing and simulation was the Typhoon HIL 402.*

Keywords: *battery model, temperature effect, aging effect, lithium-ion battery, real-time simulation, hardware-in-the-loop.*

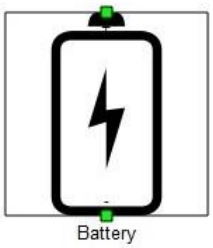
1. UVOD

Baterije su najpoznatiji tip skladištenja energije koji je zasnovan na elektrohemijskoj tehnologiji. Pomoću napona i kapaciteta se ocenjuje sposobnost sistema da obavlja različite funkcije potrebne za mrežne aplikacije [1].
Za potrebe rada korišćen je softver *Typhoon HIL Control Center*, kompanije *Typhoon HIL Inc*., koji omogućava simulaciju modela u realnom vremenu. Prototip simulacije u realnom vremenu koji se ovde koristi je simulacija sa hardverom u petlji (*engl. hardware-in-the-loop*). Komponenta baterije koja postoji u *Typhoon HIL Control Center*-u nema mogućnost analize uticaja temperature i životnog veka baterije na performanse baterije.
Iz tog razloga je modelovana nova komponenta baterije koja će imati mogućnost uključivanja temperaturnog efekta i efekta starenja. Nova komponenta je modelovana za četiri tipa baterije koji se koriste u industriji, a efekti su realizovani za litijum-jonsku bateriju. Tipovi baterije koji se najčešće koriste u industriji su: baterije sa olovnom kiselinom (*engl. Lead-Acid*), litijum-jonske (*engl. Lithium-Ion*), nikl-kadmijum (*engl. Nickel-Cadmium*) i nikl-metal-hidrid (*engl. Nickel-Metal-Hydride*).
Još jedan od razloga za modelovanje ove komponente baterije sa pomenutim efektima je taj što je postojeća komponenta baterije zasnovana na tabelama za pretraživanje (*engl. Look-Up tables*) koje se koriste za simulaciju ponašanja nelinearnih elemenata. LUT tabele opterećuju FPGA (*engl. Field Programmabile Gate Array*) procesor velikom brzinom i visokom rezolucijom. FPGA procesori se najčešće koriste prilikom simulacije u realnom vremenu [2]. Iz tog razloga se model baterije realizuje pomoću alata za obradu signala (*engl. signal processing*).
Rezultati modelovane baterije su posmatrani pomoću alata *HIL SCADA*, a testiranje je obavljeno uz pomoć *Typhoon HIL 402* uređaja.

2. MODEL BATERIJE SA TEMPERATURNIM EFEKTOM I EFEKTOM STARENJA

2.1. Uopšteno o modelu baterije

Model baterije je predstavljen matematičkim modelom koji za cilj ima analizu ponašanja različitih tipova baterije u zavisnosti od ulazne veličine – struje. Na izlazu modela baterije dobija se napon čija je vrednost stvarna vrednost napona u bilo kom trenutku. Razvijen je model za četiri tipa baterije koji se najčešće koriste u industriji. Na slici 1 je prikazana komponenta baterije koja će se nalaziti u biblioteci *Typhoon HIL Control Center*-a.



Slika 1: *Komponenta baterije u Typhoon HIL Control Center-u*

Realizacija modela baterije je započeta pomoću jednačine 1.1 za stanje napunjenosti (*engl. State of charge* – SOC). Stanje napunjenosti predstavlja napunjenost baterije izraženu u procentima pune napunjenosti i zavisi od ulazne struje *i*.

$$SOC = 100 * (1 - it / Q)\ [\%] \quad (1.1)$$

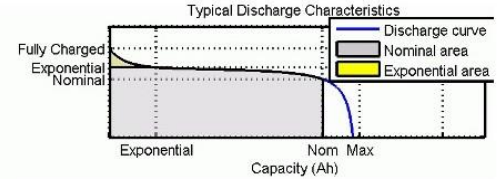
$$it = \int i * dt, \quad (1.2)$$

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Milan Rapaić, vanredni profesor i kao rezultat stručne prakse u kompaniji *Typhoon HIL Inc.* čiji je mentor bio M.Sc Adrien Genić.

gde je Q kapacitivnost baterije [Ah], a *it* struja pražnjenja [Ah] (*engl. discharge current*). Struja pražnjenja je definisana kao integral struje po vremenu (jednačina 1.2) i to iz razloga što je za potrebe rada neophodno da se analizira stanje napunjenosti tokom vremena. Stanje napunjenosti nam pokazuje kako zadato opterećenje, struja, utiče na punjenje i pražnjenje baterije. Za male vrednosti zadatog opterećenja, baterija će se sporije puniti ili prazniti, što znači da će se za veće vrednosti opterećenja baterija brže puniti ili prazniti.

Parametri modela se modifikuju tako da predstavljaju specifičan tip baterije i njegove karakteristike pražnjenja. Karakteristika pražnjenja sastoji se iz tri zone (slika 2):

1) Eksponencijalna zona (*engl. Exponentional zone*) - predstavlja pad napona kada je baterija puna. Širina pada napona zavisi od tipa baterije.
2) Nominalna zona (*engl. Nominal zone*) – predstavlja naelektrisanje koje može da se izvuče iz baterije sve dok napon ne padne ispod nominalnog napona baterije.
3) Zona pražnjenja (*engl. Discharge zone*) – predstavlja totalno pražnjenje baterije. Napon opada brzinom koja zavisi od kapacitivnosti baterije [3].



Slika 2: *Karakteristika pražnjenja*

Kontrolisani izvor napona, izražen u voltima, takođe zavisi od struje pražnjenja baterije *it* (jednačina 1.3).

$$E = E0 - K * \left(\frac{Q}{Q-it}\right) + A * e^{(-B*it)}, \quad (1.3)$$

gde je A pad napona tokom eksponencijalne zone [V], B inverzna vremenska konstanta eksponencijalne zone [1/Ah], K polarizaciona naponska konstanta [V/Ah] i E0 naponska konstanta [V/Ah].

Ovi parametri se računaju na sledeći način:

$$A = E_{full} - E_{exp} \quad (1.4)$$

$$B = \frac{3}{Qexp} \quad (1.5)$$

$$K = \frac{\left(Efull - Enom + A*\left(e^{-B*Qnom} - 1\right)\right)*(Q - Qnom)}{Qnom} \quad (1.6)$$

$$E0 = E_{full} - K + R_{internal} * i - A, \quad (1.7)$$

gde je E_{full} maksimalan napon punjenja baterije izražen u voltima, a E_{exp} napon eksponencijalne zone. Q_{exp} je kapacitivnosti baterija u eksponencijalnoj zoni izražen u procentima nominalne kapacitivnosti baterije Q_{nom}. $R_{internal}$ je unutrašnja otpornost baterije (Ω) [5]. Eksponencijalni napon se povećava kada se baterija puni, bez obzira na stanje napunjenosti baterije, a odmah se smanjuje kada se baterija prazni.

2.2. Temperaturni efekat

Temperaturni efekat je najčešće vezan za litijum-jonske baterije. Većina temperaturnih efekata vezana je za hemijske reakcije koje se dešavaju u baterijama, kao i u materijalima koji se koriste za baterije.

Prihvatljiva temperatura litijum-jonskih baterija je od -20˚C do 60˚C. Optimalna radna temperatura se obično nalazi u opsegu od 15˚C do 35˚C. Ako se vrednost temperature ne nalazi u optimalnom opsegu baterija će brzo degradirati i tako će se povećati rizik sa sigurnosnim problemima koji uključuju požar i eksploziju.

Visoka unutrašnja temperatura baterije može dovesti do pogoršanja performansi, gde se to najviše odnosi na gubitak kapaciteta i snage. Do gubitka kapaciteta i energije dolazi jer se pri visokim temperaturama gubi litijum i smanjuju se aktivne materije. U nekim slučajevima ovakva situacija može dovesti do samozapaljivanja ili eksplozije i zbog čega je neophodno pravilno upravlja radnom temperaturom baterija.

Temperaturni efekat se definiše sledećom jednačinom izlaznog napona u zavisnosti od temperature:

$$E(T) = E0(T) - K(T) * \left(\frac{Q(T)}{Q(T)-it}\right) + A(T) * e^{(-B*it)} \quad (1.8)$$

Kapacitivnost baterije Q se smanjuje sa povećanjem temperature T koja utiče na bateriju. Temperatura T je temperatura okoline, odnosno ambijenta koja je promenljiva. Kapacitivnost Q takođe varira u zavisnosti od struje pražnjenja *it*, što znači da temperatura utiče i na unutrašnju otpornost baterije. Degradacije kapacitivnosti se moraju prvo utvrditi pre izračunavanja konstanti degradacije. Konstante degradacije utiču na parametre jednačine (1.8) i zbog toga se parametri računaju na sledeći način:

$$E0(T) = E0 * (1 + k_{vt1} * (T - T_{ref})) \quad (1.9)$$

$$K(T) = K * (1 + k_{kt1} * (T - T_{ref})) \quad (1.10)$$

$$Q(T) = Q * (1 + k_{qt1} * (T - T_{ref}) + k_{qt2} * (T - T_{ref})) \quad (1.11)$$

$$A(T) = A * (1 + k_{at1} * (T - T_{ref})) \quad (1.12)$$

T_{ref} je referentna temperatura i njena vrednost iznosi 25˚C. k_{xt1} je konstanta degradacije prvog reda, a k_{xt2} je konstanta degradacije drugog reda gde je x parametar koji zavisi od temperature. [4] U ovom modelu baterije konstantne degradacije smo proglasili konstantnim vrednostima za svaki tip litijum-jonske baterije.

2.3. Efekat starenja

Efekat starenja je predstavljen kao životni ciklus baterije koji nam opisuje posledice uticaja perioda korišćenja baterije. Taj period nazivamo *životni ciklus* koji podrazumeva punjenje ili pražnjenje baterije. Starenje u zavisnosti od životnog ciklusa uključuje mnogo varijabli koje su međusobno zavisne, poput temperature i napona. Ove promenljive su povezane sa spoljašnjim uslovima i potrebom baterija. Glavni faktori koji se razmatraju su temperatura, SOC, napon i broj životnih ciklusa. Rad baterije se tokom životnog veka pogoršava zbog degradacije njegovih elektrohemijskih sastojaka, što dovodi do pogoršanja performansi i potrošnje.

Realizacija efekta starenja je zasnovana na životnom ciklusu baterije n. Životni ciklus je ciklus punjenja i pražnjenja baterije koji počinje od pražnjenja baterije do potpuno ispražnjenog stanja i zatim punjenje baterije do potpuno napunjenog stanja baterije. Efekat starenja dovodi do gubitka nominalnog kapaciteta, bržeg porasta temperature tokom rada, manjeg prihvatanja punjenja, nižeg napona, kao i do čestog samopražnjenja baterije.

Jednačina izlaznog napona u zavisnosti od životnog ciklusa baterije n (1.13) se koristi za realizaciju efekta starenja.

$$E(n) = E0(n) - K(n) * \left(\frac{Q(n)}{Q(n)-it}\right) + A(n) * e^{(-B*it)} \quad (1.13)$$

Ovde su takođe iskorišćene modifikovane jednačine sa konstantama degradacije za realizaciju efekta starenja. U sledećim jednačinama možemo uvideti kako životni ciklus baterije utiče na sve parametre:

$$E0(n) = E0 * (1 + k_{vn1} * (n - 1)) \quad (1.14)$$

$$K(n) = K * (1 + k_{kn1} * (n - 1)) \quad (1.15)$$

$$Q(n) = Q * (1+k_{qn1}*(n - h1) + k_{qn2} * (n - 1)) \quad (1.16)$$

$$A(n) = A * (1+k_{an1} * (n - 1)) \quad (1.17)$$

k_{xn1} je konstanta degradacije prvog reda, a k_{xn2} je konstanta degradacije drugog reda gde je x parametar koji zavisi od životnog ciklusa baterije [4].

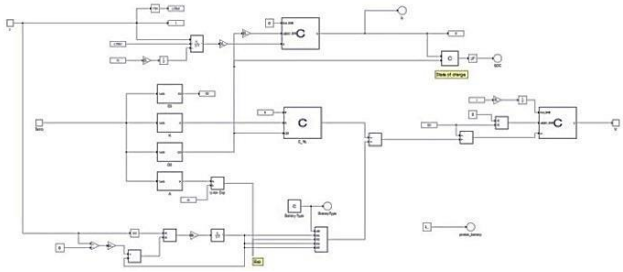
Zaključno sa temperaturnim efektom i efektom starenja jednačina izlaznog napona modela baterije glasi:

$$E(T,n) = E0(T,n) - K(T,n) * \left(\frac{Q(T,n)}{Q(T,n)-it}\right) + A(T,n) * e^{(-B*it)} \quad (1.18)$$

3. IMPLEMENTACIJA MODELA BATERIJE SA TEMPERATURNIM EFEKTOM I EFEKTOM STARENJA

Ovaj model baterije je modelovan na osnovu već postojeće komponente baterije u *Typhoon HIL Control Center*-u i na osnovu komponente baterije iz MATLAB-a. Cilj modelovanja baterije je da se uključe temperaturni efekat i efekat starenja, kao i poboljšanje performansi već postojeće komponente.

Na osnovu datog matematičkog modela iz drugog poglavlja model baterije je implementiran u *Typhoon HIL Control Center*-u uz pomoć alata za obradu signala (*engl. signal processing*) u *Schematic Editor*-u. Model baterije je modelovan za četiri osnovna tipa baterije koji se najčešće koriste u industriji: baterija sa olovnom kiselinom, litijum-jonska, nikl-kadmijum i nikl-metal-hidrid baterija. Na slici 3 prikazana je interna struktura modela baterije.



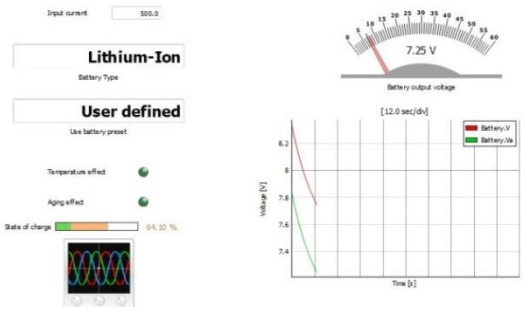
Slika 3. *Interna struktura modela baterije*

Osnovni model baterije i temperaturni efekat su implementirani u *Schematic Editor*-u, a efekat starenja je definisan u *Mask Editor*-u. Pored efekta starenja u *Mask Editor*-u su definisane sve konstantne promenljive, kao i nominalne vrednosti za svaki tip baterije i definisan je sam izgled maske komponente. Ovde je takođe definisano povezivanje interne strukture i maske komponente.

4. REZULTATI SIMULACIJE MODELOVANE BATERIJE

4.1. SCADA sistem i rezultati simulacije

Model baterije sa temperaturnim efektom i efektom starenja je simuliran pomoću alata *HIL SCADA* na uređaju Typhoon HIL 402. Na slici 4. prikazan je moguć SCADA sistem za nadzor i upravljanje komponentom baterije.

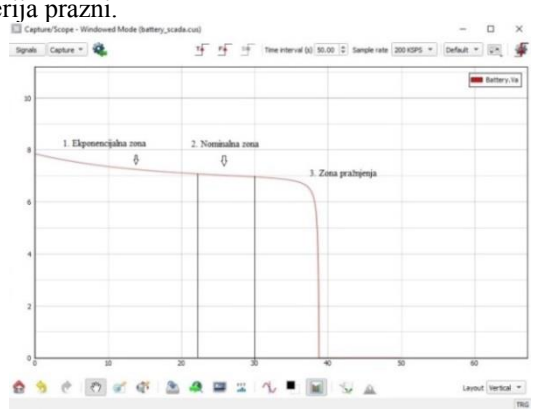


Slika 4. *SCADA komponente baterije*

SCADA sistem se sastoji od *HIL SCADA* kontrolnih elemenata (*engl. HIL SCADA widgets*) pomoću kojih se unosi ulazna struja, prikazuje tip baterije, kao i tip litijum-jonske baterije, signalizira nam da li su efekti uključeni, takođe nam pokazuje koliko je baterija puna ili prazna, a možemo videti i vrednost izlaznog napona baterije u bilo kom trenutku.

Rezultati simulacije su dobijeni tako što je korisnik zahtevao da se baterija isprazni. Iz tog razloga ulazna struje je pozitivna i iznosi 500A. Izabrani tip baterije je litijum-jonska baterija i njen tip definiše sam korisnik (*engl. User define*d). Ulazna struja je male vrednosti, pa se baterija sporije prazni.

Na slici 5. je grafički predstavljena karakteristika pražnjenja baterije sa obeleženim zonama. Prema grafiku se može zaključiti da se pri ulaznoj struji od 500A litijum-jonska baterija prazni nakon 38s. Kada vrednost izlaznog napona padne ispod vrednosti nominalnog napona, tada se baterija prazni.



Slika 5. *Karakteristika pražnjenja litijum-jonske baterije*

Za dalju analizu modela, biće uključena oba efekta. Za tip litijum-jonske baterije je odabran tip **7.4V 5.4Ah (LiCoO2)**, što znači da je nominalni napon ove baterije 7.4V, a nominalna kapacitivnost 5.4Ah. Pošto je uključen temperaturni efekat, zadata temperatura ambijenta je 30˚C. Kako je ambijentalna temperatura u opsegu optimalne ambijentalne temperature, kapacitivnost baterije ne opada, ali se baterija sporije prazni. U slučaju da ambijentalna temperatura nije u optimalnom opsegu, došlo bi do pogoršanja performansi.

Pored temperaturnog efekta, uključen je i efekat starenja. Životni vek koji je zadat je 10. Promene u kapacitetu i naponu će nastati tek nakon 10 ciklusa punjenja i pražnjenja baterije. Promene koje mogu nastati su: snižavanje napona, gubitak nominalnog kapaciteta, brži porast temperature tokom rada, često samopražnjenje baterije, kao i povećanja unutrašnje otpornosti baterije.

4.2. Poređenje postojeće komponente baterije sa komponentom baterije koja je modelovana sa temperaturnim efektom i efektom starenja

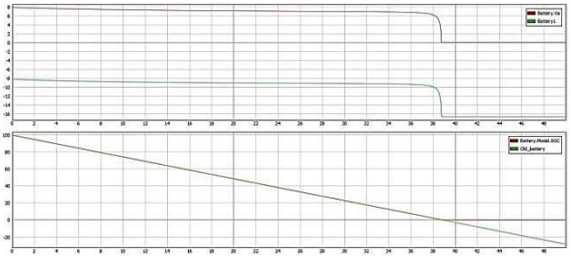
Neophodno je uporediti već postojeću komponentu baterije sa komponentom koja je modelovana sa temperaturnim efektom i efektom starenja. Kako bi se ove dve komponente poredile, nova komponenta neće imati nijedan efekat uključen.

Obe komponente su priključene na kontrolisani strujni izvor. Za obe komponente je izabrana litijum-jonska baterija. Nominalni napon je 7.2V, a nominalna kapacitivnost baterije je 5.4Ah. Obe baterije su pune jer im je inicijalno stanje napunjenosti postavljeno na 100%. Posmatra se kako se baterije prazne pri ulaznoj struji od 500A.

Na slici 6 su prikazani rezultati poređenja komponenti. Prvi grafik prikazuje izlazne napone sa padom napona na unutrašnjoj otpornosti obe komponente. Crveni signal je izlazni napon novomodelovane baterije, a zeleni već postojeće baterije u *Typhoon HIL Control Center*-u. Može se primetiti da su signali jednaki što se tiče oblika signala, ali takođe se može uočiti da izlazni napon već postojeće komponente ima amplitudu u početnom trenutku -8V, a ne 8V kao kod novomodelovane komponente.

Kod novomodelovane baterije je u implementaciji interne strukture pre samog izlaza postavljen blok „*C function*“ koji predstavlja ograničenje signala. Gornja granica je ulazna struja koja je pomnožena sa vrednošću unutrašnje otpornosti baterije i zatim ova vrednost prolazi kroz „*Unit Delay*“ blok koji odlaže signal za jedan period uzorkovanja.

Donja granica je dvostruka vrednost naponske konstante E0. Takođe, može se uočiti da je novom komponentom dobijena željena karakteristika pražnjenja.



Slika 6. *Rezultati poređenja komponenti*

Na drugom grafiku je predstavljeno stanje napunjenosti obe baterije. Crveni signal je stanje napunjenosti novomodelovane baterije, a zeleni već postojeće baterije. Signali stanja napunjenosti su jednaki dok se baterija ne isprazni. Uočava se da signal nove komponente ostaje na 0 kada se baterija isprazni, dok signal stare komponente linearno opada tokom vremena bez obzira što je baterija prazna. Kod nove komponente je uvedeno ograničenje za vrednost signala stanja napunjenosti, gde je donja granica 0, a gornja 100.

Obe baterije rade na sličnom principu, sa istim vrednostima parametara, a na osnovu grafika se vidi da se u istom vremenskom trenutku one isprazne.

5. ZAKLJUČAK

Na osnovu modela baterije iz MATLAB-a i *Typhoon HIL Control Center*-a dobijen je novi model baterije koji ima mogućnost uključivanja temperaturnog efekta i efekta starenja. Efekti omogućavaju klijentima da analize, koje su njima neophodne, budu potpune.

Ova komponenta baterije je modelovana uz pomoć alata za obradu signala i na taj način je obezbeđeno da FPGA procesor ne bude opterećen, kao što je to slučaj sa tabelama za pretraživanje (*engl. Look-Up tables*). Prilikom testiranja na uređaju HIL 402, ova komponenta je opteretila procesor samo 51%. Takođe, na ovaj način su smanjeni troškovi komponente baterije.

Prilikom modelovanja, u internu strukturu baterije su postavljeni integratori, filtri i ograničenja na određene signale kako bi se na izlazu dobili signali koji su u opsegu optimalnih vrednosti i koji nam daju sve korisne informacije. Za signal koji simulira stanje napunjenosti baterije uvedeno je takvo ograničenje da kada se baterija isprazni, dok ne krene ponovo da se puni, signal ima vrednost 0, a kada je puna izlazni signal ne može da pređe 100% napunjenosti baterije. Takođe, izlazni signal napona je stvarna vrednost napona u bilo kom trenutku i kriva pražnjenja baterije neće biti prikazana ispod x ose u negativnim vrednostima.

Svi tipovi baterije koji su implementirani u ovom modelu daju očekivanu karakteristiku pražnjenja. U rezultatima je pokazana očekivana karakteristika pražnjenja litijum-jonske baterije i karakteristika pražnjenja koja je dobijena ovim modelom baterije.

6. LITERATURA

[1] Andrei Ter-Gazarian. *Energy storage for power systems*. Institution of Electrical Engineers, London, United Kingdom.

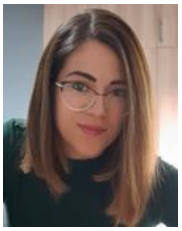
[2] https://www.typhoon-hil.com/fpga-powered-emulation/

[3] https://www.mathworks.com/help/physmod/sps/-powersys/ref/battery.html#References

[4] D. Song, C. Sun, Q. Wang, D. Jang. *A Generic Battery Model and Its Parameter Identification*. Energy, Ming and Environment Research Centre, National Research Council of Canada, Vancouver, Canada.

[5] Typhoon HIL Control Center/ Documentation Hub/ *Battery – Modeling and Application.* T-TN003(v1.1) June 12, 2013.

Kratka biografija:



Nataša Panić rođena je u Novom Sadu 1995. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Adaptivno i napredno upravljanje odbranila je 2020.god.

Kontakt: natasap1995@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.314.21
DOI: https://doi.org/10.24867/12BE16Stefanovic

ENERGETSKI TRANSFORMATORI U USTALJENIM SLOŽENOPERIODIČNIM REŽIMIMA

POWER TRANSFORMERS IN PERIODIC NON-SINUSOIDAL STEADY STATES

Nina Stefanović, Dejan Jerkan, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je razvijen matematički model trofaznog, dvonamotajnog energetskog transformatora sprege Dyn5, podesan za proračune ustaljenih složenoperiodičnih režima. Model obezbeđuje uvažavanje postojanja nenultog sprežnog broja transformatora, omogućavajući proračune ustaljenih složenoperiodičnih radnih režima uz istovremeno prisustvo nesimetrije i/ili neuravnoteženosti. Na ilustrativnim primerima je pokazano kakav je uticaj energetskih transformatora na uspostavljanje složenoperiodičnih režima usled harmonijskog izobličenja napona napajanja na njegovim primarnim priključcima.*

Ključne reči: *Energetski transformatori, Složenoperiodični režimi, Furijeova analiza, MATLAB Simulink*

Abstract *The paper presents a mathematical model of a three-phase, two-winding power transformer with Dyn5 connection, suitable for calculations of non-sinusoidal periodic steady states. The model ensures that the existence of a non-zero vector group number of transformers is taken into account, enabling calculations of periodic steady states with the simultaneous presence of asymmetry and/or imbalance. Illustrative examples show the influence of power transformers on the establishment of non-sinusoidal periodic states due to the harmonic distortion of the supply voltage at its primary connections.*

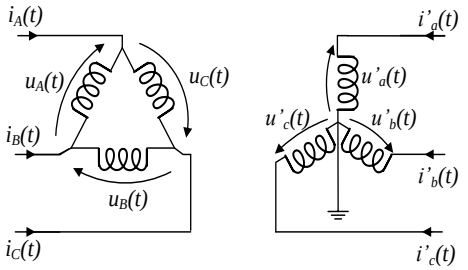
Keywords: *Power transformers, Periodic non-sinusoidal regimes, Fourier analysis, MATLAB Simulink*

1. UVOD

Energetski transformatori su jedni od najznačajnijih elemenata u elektroenergetskom sistemu (EES). Na putu pretvaranja energije od njenih primarnih izvora u konvencionalnim elektranama, preko visokonaponske prenosne mreže, pa sve do potrošača na niskom naponu, električna energija biva transformisana u nekoliko stepena, posredstvom energetskih transformatora. Smisao primene transformatora jeste energetski efikasan prenos električne energije, transformisanjem naponskih nivoa na kojima se određena količina električne energije ekonomski najisplativije prenosi i distribuira. Usled sveprisutnosti transformatora za očekivati je da je njihov uticaj na uspostavljanje radnih režima EES-a veoma značajan, ali i da se zbog toga nužno i sami transformatori mogu naći u raznim uslovima, često nedopustivim i nepovoljnim. Pored tipičnih trajno nedozvoljenih pogonskih stanja, poput pojave preopterećenja, previsokih napona ili kratkih spojeva, u poslednje vreme, a ponajviše usled povećane upotrebe nelinearnih elemenata u EES-u, dolazi do pojačane izloženosti energetskih transformatora složenoperiodičnim radnim režimima. Tema ovog rada upravo predstavlja razvoj matematičkog modela energetskog transformatora, koji na pravilan način uvažava uspostavljanje složenoperiodičnih režima uzimajući u obzir osobenosti trofaznih dvonamotajnih energetskih transformatora, uključujući i postojanje nenultog sprežnog broja. Budući da je pojam sprežnog broja, kao i premise pod kojima se on definiše vezan za rad transformatora u ustaljenim i simetričnim prostoperiodičnim režimima direktnog redosleda, izvršiće se odgovarajuća matematička generalizacija pomoću koje će se modeli transformatora za ustaljene prostoperiodične režime prevesti u odgovarajuće modele za proračun ustaljenih složenoperiodičnih režima, uz dopuštenu istovremenu pojavu nesimetrije i/ili neuravnoteženosti.

2. MODEL TROFAZNOG DVONAMOTAJNOG TRANSFORMATORA

Model se zasniva na principu Teorije električnih kola, poznatom kao princip superpozicije, koji podrazumeva linearnost elementa čiji se radni režim želi rekonstruisati, tako da će razvijeni model biti u mogućnosti da pravilno uvaži uticaj koji složenoperiodični režimi imaju na energetske transformatore i obratno, ali samo u slučaju da sam transformator nije uzročnik njihovom nastanku.



Slika 2.1. *Principska šema dvonamotajnog transformatora sprege Dynk*

Na slici 2.1 prikazana je principska šema trofaznog, dvonamotajnog transformatora sprege *Dynk*, gde *k* predstavlja sprežni broj transformatora, koji za raznorodne sprege može poprimiti bilo koju vrednost iz skupa {1, 3, 5, 7, 9, 11}. Velikim slovima *A*,*B* i *C* u indeksu označeni su

---

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

naponi i struje na njegovim višenaponskim, dok su malim slovima a, b i c označeni naponi i struje na niženaponskim priključcima. Izabrano je modelovanje transformatora ovog tipa sprege, jer je on po pravilu najčešći u eksploataciji u distributivnim mrežama, a u kojima je ujedno i dominantno zastupljena pojava složenoperiodičnih režima.

Ukoliko se transformator nalazi u ustaljenom prostoperiodičnom režimu, za njega je moguće izvesti matrični model u kompleksnom domenu, gde svaka od dvanaest terminalnih veličina (po šest napona i struja) biva reprezentovana svojim kompleksnim predstavnikom. Ne umanjujući opštost razmatranja, na dalje će se razmatrati transformator sprežnog broja $k = 5$.

Matrični model u faznom domenu, kojim se uspostavlja linearna veza između kompleksnih predstavnika dvanaest terminalnih veličina transformatora sprege *Dyn5* prikazan je relacijom 1:

$$\begin{bmatrix} \underline{I}_{hA} \\ \underline{I}_{hB} \\ \underline{I}_{hC} \\ \underline{I}'_{ha} \\ \underline{I}'_{hb} \\ \underline{I}'_{hc} \end{bmatrix} = \begin{bmatrix} \frac{2\underline{Y}_h{}^d}{3} & -\frac{\underline{Y}_h{}^d}{3} & -\frac{\underline{Y}_h{}^d}{3} & \frac{\underline{Y}_h{}^d}{\sqrt{3}} & 0 & -\frac{\underline{Y}_h{}^d}{\sqrt{3}} \\ -\frac{\underline{Y}_h{}^d}{3} & \frac{2\underline{Y}_h{}^d}{3} & -\frac{\underline{Y}_h{}^d}{3} & -\frac{\underline{Y}_h{}^d}{\sqrt{3}} & \frac{\underline{Y}_h{}^d}{\sqrt{3}} & 0 \\ -\frac{\underline{Y}_h{}^d}{3} & -\frac{\underline{Y}_h{}^d}{3} & \frac{2\underline{Y}_h{}^d}{3} & 0 & -\frac{\underline{Y}_h{}^d}{\sqrt{3}} & \frac{\underline{Y}_h{}^d}{\sqrt{3}} \\ -\frac{\underline{Y}_h{}^d}{\sqrt{3}} & \frac{\underline{Y}_h{}^d}{\sqrt{3}} & 0 & -\underline{Y}_h{}^d & 0 & 0 \\ 0 & -\frac{\underline{Y}_h{}^d}{\sqrt{3}} & \frac{\underline{Y}_h{}^d}{\sqrt{3}} & 0 & -\underline{Y}_h{}^d & 0 \\ \frac{\underline{Y}_h{}^d}{\sqrt{3}} & 0 & -\frac{\underline{Y}_h{}^d}{\sqrt{3}} & 0 & 0 & -\underline{Y}_h{}^d \end{bmatrix} \begin{bmatrix} \underline{U}_{hA} \\ \underline{U}_{hB} \\ \underline{U}_{hC} \\ \underline{U}'_{ha} \\ \underline{U}'_{hb} \\ \underline{U}'_{hc} \end{bmatrix}, \qquad (1)$$

gde $\underline{Y}_h{}^d$ predstavlja admitansu direktnog redosleda simetrije za harmonik reda h. Prilikom izvođenja relacije je usvojeno da su admitanse sva tri redosleda simetrije međusobno jednake, što u opštem slučaju ne mora biti na snazi. Admitanse direktnog i inverznog redosleda za statičke i uravnotežene elemente po prirodi jesu podjednake, dok za admitansu nultog redosleda to ne mora biti slučaj, jer njena vrednost zavisi od konstrukcije elementa i njegovog statusa prema neutralnoj tački. Izjednačavanje admitansi je u ovom radu izvršeno samo zarad kompaktnijeg zapisa, što ni u čemu ne umanjuje opštost. Admitansa direktnog redosleda simetrije predstavlja recipročnu vrednost impedanse kratkog spoja, koja se često može aproksimirati reaktansom induktivnog karaktera, zanemarivanjem aktivne otpornosti namotaja:

$$\underline{Y}_h{}^d = \frac{1}{R_k + jX_{hk}} = \frac{1}{R_k + j\omega_h L_k} \approx \frac{1}{h}\frac{1}{jX_k}, \qquad (2)$$

gde R_k predstavlja aktivnu otpornost kratkog spoja transformatora, a X_k reaktansu kratkog spoja na učestanosti osnovnog harmonika. Porastom reda harmonika aproksimacija biva sve preciznija predstava rednih parametara ekvivalentnih šema transformatora. Matrični model se može primeniti, primenom principa superpozicije, u rešavanju složenoperiodičnih režima nametnutih transformatoru, tako što se relacija (1) prilagodi i upotrebi u onoliko različitih prostoperiodičnih podproblema koliko složenoperiodične veličine kojima je transformator izložen ukupno imaju prostoperiodičnih harmonijskih komponenti.

Najpre je potrebno odrediti harmonijski sastav složenoperiodičnih veličina kojima je transformator izložen. Zatim se sve harmonijske komponente čije je prisustvo utvrđeno prevedu u odgovarajuće kompleksne predstavnike, a uz važnu napomenu da prelaskom u domen kompleksnih predstavnika harmonici na istoj učestanosti čine kompleksni podprostor koji je disjunktan sa ostalim podprostorima koje tvore kompleksni predstavnici harmonika na bilo kojoj drugoj učestanosti.

Nakon uvođenja potrebnog broja disjunktnih kompleksnih podprostora, može se pristupiti njihovom rešavanju, a na osnovu dodatnih relacija kojima se bliže opisuje priroda složenoperiodičnog režima koji se želi rekonstruisati. Postupnim rešavanjem disjunktnih podproblema u kompleksnim podprostorima moguće je izvršiti prevođenje izračunatih kompleksnih predstavnika nepoznatih veličina u pripadajuće harmonijske komponente u vremenskom domenu, gde se složenoperiodični odzivi zadatog problema dobijaju sabiranjem pojedinačnih harmonika koji pripadaju odgovarajućim terminalnim veličinama. Na taj način je izvršena dosledna primena principa superpozicije, raščlanjivanjem originalnog problema na odgovarajući broj prostoperiodičnih podproblema.

Zbog činjenice da je transformator modelovan kao linearni element, on svojim postojanjem neće dovoditi do stvaranja harmonijskih komponenti kojih nema ni u jednoj od složenoperiodičnih veličina kojima je izložen, ali svojom prirodom može uticati na to da pojedine harmonijske komponente budu suzbijene, a pojedine čak i pospešene. Relacija (1) u opštem slučaju ima dvanaest nepoznatih terminalnih veličina, dok je sama relacija šestog reda, na osnovu čega sledi da je neke od veličina potrebno poznavati unapred, ali i da je potrebno uvesti dodatne relacije koje u određenim režimima važe između pojedinih terminalnih veličina, kako bi relacija postala jednoznačno rešiva.

U narednom poglavlju će se razvijeni model koristiti za rekonstruisanje režima u kojima je unapred poznato naponsko pobuđivanje transformatora preko primarnih priključaka, dok će se sa njegove sekundarne strane modelovati različita pogonska stanja, čijim podrobnijim opisivanjem dodatnim relacijama, uz poznavanje napona primarne strane, matrična relacija (1) postaje jednoznačno rešiva za svaki od prostoperiodičnih podproblema.

3. PRIMENA RAZVIJENOG MODELA TRANSFORMATORA

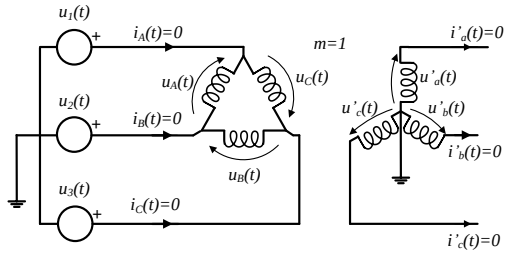
Pomoću razvijenog modela se mogu rekonstuisati razni karakteristični radni režimi transformatora, uvođenjem dodatnih relacija kojima se podrobnije opisuje njihova topologija.

Modelu je moguće nametnuti praktično bilo kakve talasne oblike terminalnih veličina, ali će se za potrebe njegove verifikacije usvojiti unapred poznati složenoperiodični talasni oblici napona primara sa harmonijskim sastavom koji se može očekivati u eksploataciji.

Isti harmonijski sastav napona primara će se zatim koristiti za sve radne režime koji će se u ovom poglavlju razmatrati.

3.1. Transformator u složenoperiodičnom režimu praznog hoda

Kao prvi karakteristični radni režim će se rekonstuisati rad transformatora u praznom hodu. Na slici 3.1 je data principska šema modelovanog transformatora u ovom radnom režimu. Zbog zanemarenja otočnih parametara prilikom izvođenja matematičkog modela, za očekivati je da je svih šest struja unapred poznato i jednako nuli, što upravo predstavlja dodatne relacije topologije razmatranog režima kojima se model upotpunjuje do njegove određenosti, što je i naznačeno na samoj slici 3.1.



Slika 3.1. *Principska šema transformatora u režimu praznog hoda*

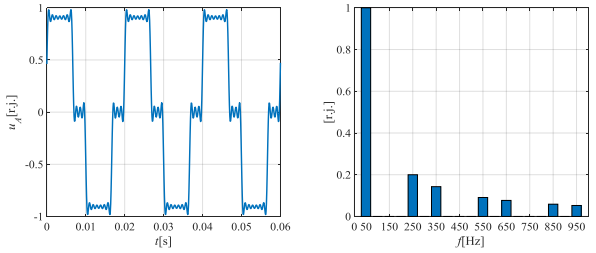
Idealni naponski izvori napajanja primarne strane su izabrani tako da predstavljaju trojku simetričnih veličina, koje zbog prirode viših harmonika u EES-u imaju isključivo harmonike neparnog reda. Za naponski izvor u fazi *A* prema tome važi:

$$u_1(t) = \sum_{l=1}^{N} U_{m_{2l-1}} \cdot \cos((2l-1)\omega_1 t + \varphi_{0_{2l-1}}), \qquad (3)$$

gde $U_{m_{2l-1}}$ predstavlja amplitudu harmonika reda $h = (2l-1)$, $\varphi_{0_{2l-1}}$ predstavlja fazni stav, ω_1 je ugaona učestanost osnovnog harmonika, a N određuje red harmonika. Naponi preostala dva izvora se dobijaju faznim pomeranjem sabiraka u izrazu (3) za $\pm\frac{2\pi}{3}(2l-1)$. Naponi faza primarne strane transformatora se od napona mreže dobijaju oduzimanjem odgovarajućih parova:

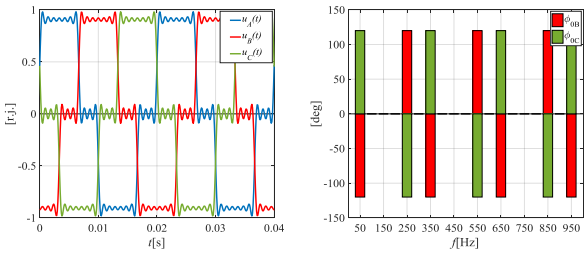
$$\begin{aligned} u_A(t) &= u_1(t) - u_2(t) \\ u_B(t) &= u_2(t) - u_3(t) \\ u_C(t) &= u_3(t) - u_1(t). \end{aligned} \qquad (4)$$

Na slici 3.2 je prikazan amplitudni spektar napona faze *A* primara u relativnim jedinicama, koji je dobijen na osnovu napona $u_{1,2,3}(t)$ u kojima amplituda harmonika opada obrnuto proporcionalno njihovom redu, dok je za početni fazni stav svakog od harmonika uzeta nulta vrednost.



Slika 3.2. *Napon faze A primara i njegov amplitudni spektar*

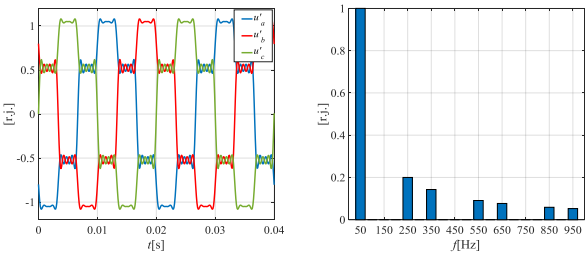
Na slici 3.3 su prikazani naponi primara za faze *A*, *B* i *C*, kao i fazni spektar faza *B* i *C*.



Slika 3.3. *Talasni oblici napona faza A, B i C, kao i fazni spektar faza B i C*

Uz zadati profil napona primara i unapred poznate struje u režimu praznog hoda, primenom relacije (1) za svaki red harmonika ponaosob se mogu dobiti nepoznati naponi sekundara, postupnim rešavanjem odgovarajućeg broja matričnih relacija (1), a zatim vraćanjem dobijenih kompleksnih predstavnika u vremenski domen i njihovim sumiranjem.

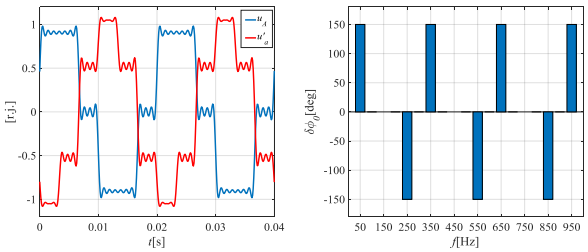
Na slici 3.4 su uporedo prikazani ovim postupkom dobijeni talasni oblici napona sekundara, kao i njihov amplitudni spektar u relativnim jedinicama.



Slika 3.4. *Talasni oblici napona faza a, b i c, kao i njihov amplitudni spektar*

Mada naponi sekundara imaju isti amplitudni spektar kao i oni sa primarne strane, primećuje se da su im talasni oblici potpuno drugačiji. Ovo se duguje činjenici da model transformatora uvažava nenulti sprežni broj, koji vrši fazno pomeranje veličina sa primarne strane za $5 \times 30° = 150°$ za harmonike reda $6l+1$, odnosno pomeranje za $-5 \times 30° = -150°$ za harmonike reda $6l+1$, što je u potpunosti u skladu sa osobinama viših harmonika neparnog reda koje se imaju u eksploataciji realnog EES-a.

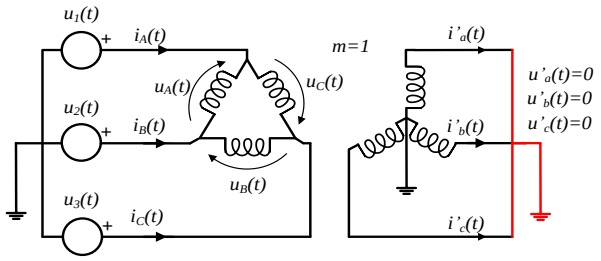
Na slici 3.5 su uporedo prikazani talasni oblici napona faze *A* primara i napona faze *a* sekundara, kao i spektar faznih pomeraja odgovarajućih parova njihovih harmonijskih komponenti, kojim se potvrđuje da razvijeni model na pravilan način uvažava postojanje nenultog sprežnog broja i njegov uticaj na talasne oblike napona sekundara u odnosu na primarne napone.



Slika 3.5. *Uporedni prikaz napona faze A i a , kao i spektar faznih razlika njihovih harmonijskih komponenti*

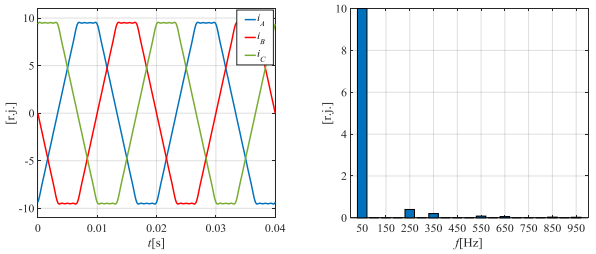
3.2. Transformator u složenoperiodičnom režimu tropolnog zemljospoja

Kao drugi primer primene modela će se izvršiti proračun ustaljenog složenoperiodičnog režima izazvanog pojavom tropolnog zemljospoja na transformatorskim sekundarnim priključcima. Kao izvori napajanja će se iskoristiti isti talasni oblici napona definisani relacijama (3) i (4), prikazani slikama 3.2 i 3.3. Tropolni zemljospoj karakterišu unapred poznati fazni naponi sekundara transformatora, koji su identički jednaki nuli, što je na slici 3.6 i naznačeno. Uz poznate talasne oblike napona primarne strane, topologija tropolnog zemljospoja opisana vrednostima napona sekundara čini svaku od matričnih relacija oblika (1) potpuno određenom za rešavanje.



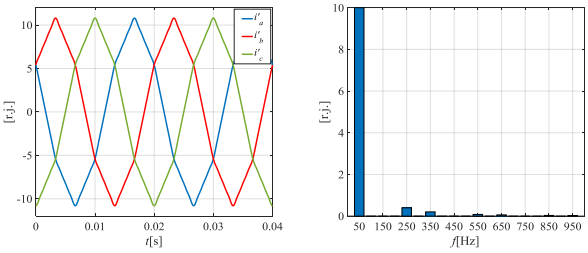
Slika 3.6. *Principska šema transformatora u režimu tropolnog zemljospoja*

Uz zadati profil napona primara i nulte vrednosti napona sekundara, primenom relacije (1) za svaki red harmonika ponaosob se mogu dobiti nepoznate struje sa oba kraja transformatora, postupnim rešavanjem odgovarajućeg broja matričnih relacija (1), a zatim vraćanjem dobijenih kompleksnih predstavnika u vremenski domen i njihovim sumiranjem. Na slici 3.7 su uporedo prikazani talasni oblici struja primara tokom tropolnog zemljospoja na sekundarnim priključcima, kao i njihov amplitudni spektar. Primetno je da su harmonijske komponente višeg reda manjih relativnih amplituda u odnosu na one u amplitudnom spektru napona napajanja, što je u svemu prema očekivanjima i u skladu sa relacijom (2), gde je očito da admitansa transformatora opada sa porastom reda harmonika, suzbijajući uspostavljanje struja viših harmonika.

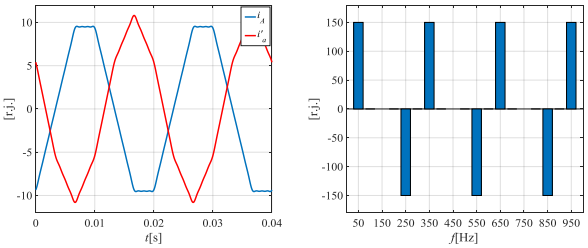


Slika 3.7. *Primarne struje tokom tropolnog zemljospoja i njihov zajednički amplitudni spektar*

Na slici 3.8 su prikazani talasni oblici sekundarnih struja, kao i njihov amplitudni spektar. Kao što je bio slučaj sa naponima u režimu praznog hoda, tako se i ovde u strujama sa različitih strana transformatora uočava isti amplitudni spektar, a različiti talasni oblici. Ovo je naravno ponovo posledica različitog dejstva nenultog sprežnog broja na harmonijske komponente različitog reda, što se jasno može uočiti na slici 3.8, gde su uporedo prikazani talasni oblici struja faze *A* primara i faze *a* sekundara, kao i spektar razlike faznih stavova njihovih odgovarajućih harmonijskih komponenti.



Slika 3.8. *Sekundarne struje tokom tropolnog zemljospoja i njihov zajednički amplitudni spektar*



Slika 3.9. *Uporedni prikaz struja faze A i a , kao i spektar faznih razlika njihovih harmonijskih komponenti*

4. ZAKLJUČAK

U radu su izložene teorijske osnove modelovanja trofaznih dvonamotajnih transformatora koje omogućavaju opisivanje njihovog rada u ustaljenim složenoperiodičnim režimima, uz istovremeno prisustvo nesimetrije i/ili neuravnoteženosti. Na primeru najčešće korišćenog tipa transformatora u distributivnim mrežama EES-a, sprege *Dyn5*, razvijeni model je verifikovan u karakterističnim ustaljenim složenoperiodičnim radnim režimima, kao što su režim praznog hoda, odnosno pojava tropolnog zemljospoja neposredno na njegovim sekundarnim priključcima. Prilaganjem rezultata proračuna u vidu talasnih oblika, te amplitudnog i faznog spektra pomoću modela izračunatih terminalnih veličina transformatora se potvrđuje da na pravilan način uvažava premise pod kojima je izveden, te da se može koristiti za rekonstruisanje praktično proizvoljnog složenoperiodičnog režima nametnutog transformatoru.

5. LITERATURA

[1] Branimir D. Reljin, „*Teorija električnih kola II*“, Akademska Misao Beograd, 2009.

[2] R. Ramirez, *“The FFT-Fundamentals and Concepts”*, Prentice-Hall Inc., New Jersey, 1985.

[3] J. Arrillaga, D. Bradley, P.Bodger, “*Power System Harmonics*”, John Wiley & Sons, Chichester,1985.

[4] Dr ing. Hrvoje Požar, „*Visokonaponska rasklopna postrojenja*“, Tehnička Misao, Zagreb.

[5] Dipl ing. Branko Mitraković, „*Трансформатори*“, Naučna Knjiga, Beograd 1968.

[6] www.mathworks.com

Kratka biografija:

Nina Stefanović rođena je u Zrenjaninu 1995. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Distribuirani energetski resursi odbranila je 2020. god.

Dejan Jerkan je docent na Fakultetu tehničkih nauka u Novom Sadu, na Katedri za Energetsku elektroniku i pretvarače. Oblast interesovanja su mu modelovanje i dijagnostika električnih mašina, kao i metoda konačnih elemenata

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 620.92
DOI: https://doi.org/10.24867/12BE17Maricic

KORIŠĆENJE HIBRIDNOG SISTEMA ZA NAPAJANJE PREČISTAČA OTPADNIH VODA U RUMENKI

HYBRID SYSTEM APPLICATION FOR WASTEWATER TREATMENT FACILITY IN RUMENKA

Gojko Maričić, Vladimir Katić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je dat predlog idejnog rešenja hibridnog sistema na bazi krovne fotonaponske i biogasne elektrane za napajanje prečistača otpadnih voda u Rumenki. Softver HOMER je iskorišćen za proračun energetskog menadžmenta oba izvora u sistemu. Izvršen je pregled energetskog bilansa i tehnoekonomska analiza.*

Ključne reči: *prečistač otpadnih voda, fotonaponski sistem, biogasna elektrana.*

Abstract – *The paper is proposing a conceptual solution of a hybrid system based on a roof-top photovoltaic and biogas power plant for the supply of wastewater treatment facility in Rumenka. HOMER software was used to calculate the energy management of both sources in the system. The energy balance review and techno-economic analysis were performed*

Keywords: *wastewater treatment facility, photovoltaic system, biogas power plant*

1. UVOD

Da bi se rešili ključni energetski problemi današnjice, kao i sve veće klimatske promene usled povećanja emisije gasova staklene bašte (CO_2 i dr.) i posledičnog globalnog zagrevanja, čovečanstvo se okreće široj primeni obnovljivih izvora, čiji resursi su neograničeni. Njih čine solarna energija, energija vetra, hidroenergija, energija biomase, biogasa, energija pokretanja morske vode, geotermalna energija i sl. Energija ovih izvora najčešće se pretvara u električnu ili toplotnu. Zbog njihovog intermitetnog karaktera, često se obnovljivi izvori kombinuju u hibridne elektrane, čime se postiže efikasnije korišćenje.

U ovom radu će se razmatrati napajanje električnom energijom postrojenja za prečišćavanje vode u Rumenki. Osnovna ideja je da se dalje doprinese ekološkom karakteru ovog postrojenja tako što će se zameniti električna energija dobijena iz termoelektrane onom iz jednog hibridnog sistema. Ovaj sistem čine krovna fotonaponska (FN) elektrana i biogasna elektrana, s tim da se biogas dobija dodatnom obradom otpadnog mulja, kao nuzprodukta prečišćavanja vode. Cilj rada je da se kroz energetski bilans utvrdi opravdanost ovog rešenja.

NAPOMENA:
Ovaj rad proistekao je iz master rada, čiji mentor je bio prof. dr Vladimir Katić.

2. OBNOVLJIVI IZVORI ENERGIJE

Korišćenje obnovljivih izvora energije je postalo civilizacijska nužnost sa ekonomskog, bezbedonosnog i ekološkog aspekta. Za dobijanje električne energije najčešće se koristi energija vode, vetra, sunca i biomase (biogasa).

2.1. Solarna energija

Solarni energija se koristi na dva osnovna načina:
1. Solarni termalni kolektori – koncentrisani ili nekoncentrisani, gde se pretvara u toplotnu i
2. FN solarni moduli – povezivanjem čine FN sistem kojim se dobija električna energija, što će se dalje razmatrati u radu.

Proteklih 20 godina protekle su u eksponencijalnom porastu prodaje FN solarnih modula i razvoja solarnih sistema. FN tehnologije koje se trenutno koriste i istražuju, uspevaju da iz godine u godinu unaprede svoju efikasnost, produže svoj životni vek i smanje troškove, odnosno cenu proizvodnje. Danas FN sistemi se prave od najmanjih snaga – nekoliko kW (krovni FN sistemi) do najvećih – više stotina MW (FN sistemi na tlu) i imaju ulogu snabdevača energijom, kao jedna od najisplativijih tehnologija. Od novougrađenih postrojenja na bazi obnovljivih izvora u 2019. godini, čak 48% čine solarni FN sistemi, a predviđa se da će se sličan tempo napretka nastaviti i u narednih 5 godina [1]. Zbog ekonomičnosti i jednostavne montaže u naseljenim mestima preferira se korišćenje krovnih FN sistema. Time se štedi prostor i bolje iskorišćuju mogućnosti postojeće distributivne mreže.

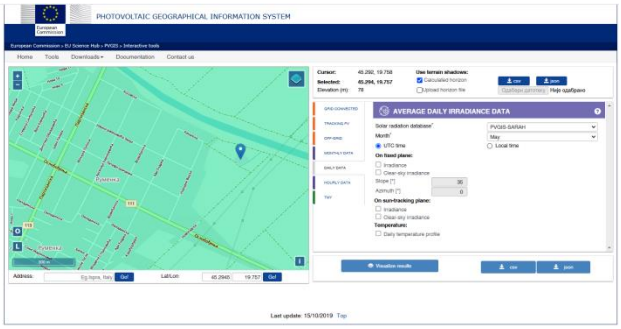
2.2. Biogas

Biogas je mešavina gasova nastala razlaganjem organske materije u odsustvu kiseonika. Glavni sastojci biogasa su metan i ugljen-dioksid. Biogas nastaje anaerobnom digestijom metanogenih ili anaerobnih organizama koji vare organsku materiju u zatvorenom sistemu. Iz 1 Nm³ deponijskog gasa dobija se 2 kWh električne energije, 2,15 kWh toplotne energije. Biogas se može dobiti iz razne vrste biomase, ali i iz različitog organskog otpada, pa i iz organske materije, koja preostaje u procesu tretmana prečišćavanja vode.

3. POTENCIJAL SUNČEVE ENERGIJE U VOJVODINI

Sunčevo zračenje zavisi od insolacionih uslova, veličine i karakteristike prijemnika sunčeve energije, te vremena izlaganja dejstvu sunčevog zračenja.

U Srbiji broj sunčanih sati se kreće od 2.000 do 2.300 sati, dnevna energija globalnog zračenja na horizontalnu površ u toku zimskog perioda od 1,0 kWh/m^2 do 1,7 kWh/m^2, leti od 5,4 kWh/m^2 do 6,9 kWh/m^2 na jugu. Srbija ima jedne od boljih resursa u Evropi, ali njem solarni potencijal nije dovoljno iskorišćen. Na teritoriji Vojvodine, godišnji prosek dnevne energije globalnog sunčevog zračenja na površinu nagnutu prema jugu pod uglom od 30° iznosi od 4,0-4,6 kWh/m² [2].
Za procenu nivoa zračenja na neku površinu i proračun FN sistema koristi se softver PVGIS (slika 1). Ovaj softverski alat omogućava dnevni, mesečni i godišnji prikaz prosečnog globalnog zračenja na horizontalne i kose površine (optimalni ugao nagiba površine). Procena deficita u godišnjem sunčevom zračenju zbog senke, nagiba i orijentacije terena takođe je obezbeđena. Ovaj softver omogućava procenu proizvodnje FN električne energije na bilo kojoj lokaiji u svetu, unošenjem njenih koordinata ili naziva mesta. Takođe, omogućava praćenje meteoroloških podataka na godišnjem nivou za devet klimatskih promenljivih [3].



Sl. 1. *Izgled PVGIS displeja* [3]

4. OPIS OBJEKTA PREČISTAČA

Radi eliminisanja štetnih materija u rečnim tokovima ili kanalima, otpadne industrijske vode ili iz domaćinstava podvrgavaju je prečišćavanju na bazi složenih biohemijskih procesa. U ovom radu razmatra se postrojenje za prečišćavanje otpadnih voda (PPOV), koje se nalazi u naselju Rumenka (slika 2).
Postrojenje se sastoji iz sledećih objekata: objekat za grubu rešetku, jedinstveni objekat crpne stanice i sigurnosnog bazena, prijemna stanica za septički mulj, objekat za biološko prečišćavanje – SBR bazeni i silos za mulj, mašinska hala, upravna zgrada, izlivna građevina i šahtne konstrukcije.
Otpadna voda dolazi do postrojenja gravitacionim putem, prvo prolazi kroz grubu rešetku, zatim se potiskuje na postrojenje za mehanički predtretman (fina rešetka, peskolov, hvatač masti). U SBR bazenima se odigravaju procesi punjenja, aeracije, mešanja, taloženja i dekantiranja. Prečišćena voda se odvodi u recepijent, višak mulja u silos za mulj, gde se vrši zgušnjavanje i dodatna stabilizacija. Vijčanom pumpom se ugušćeni mulj vodi na dehidraciju.
Ukupna instalisana snaga je 128,48 kW, pa je maksimalna jednovremena snaga 89,94 kW.
Postrojenje je povezano na distributivnu mrežu i napaja preko stubne trafostanice transformatorom snage 250 kVA. U slučaju nestanka struje uključuje se dizelagregat kao rezervno napajanje.

Sl. 2. *Izgled prečistača otpadnih voda*

5. HIBRIDNI SISTEM NAPAJANJA

5.1. Fotonaponski sistem

Za proizvodnju električne enerije iz solarne koristi se FN sistem, koji se postavlja na krovove objekata postrojenja prečistača. Kako je postrojenje povezano na distributivnu mrežu ovde se razmatra „on grid“ sistem. Sistem se sastoji iz FN panela povezanih u odgovarajuće nizove, noseće konstrukcije za panele, DC kablova i instalacije, invertora (DC/AC pretvarača), AC kablova, prekidača i druge instalacije, kao i brojila električne energije. Pored toga FN sistem ima i odgovarajuće komunikacione i bezbedonosne uređaje.
Za panele su odabrani oni snage 280 Wp sa vekom trajanja od 25 godina. Na krovovima moguće je montirati ukupno 89 panela, što obezbeđuje ukupnu snagu od 25 kWp.
Za noseću konstrukciju odabrane su dve vrste nosača, za ravni i za kosi krov. Nosači su aluminijumski, fleksibilni, otporni na udare i sneg, sa mogućnošću montaže panela horizontalno i vertikalno.
Za invertor je predložen onaj snage 25 kW (slika 3), mada je opravdano odabrati i one od 10% do 20% niže snage. Ovaj uređaj se lako i brzo instalira i izrađen u IP66 zaštiti. Sa obe strane invertora povezuju se zaštitna orema, tj. DC i AC ormani sa odgovarajućim osiguračima, prenaponskom zaštitom , diferencijalnom zaštitom. Pošto su paneli fiksirani nema potrebe za korišćenjem kontrolera za FN panele.
Na kraju za uvezivanje sistema ostaju kablovi. Za jednosmernu struju koristi se nadzemni kabel, koji se polaže duž postojećih kablovskih trasa, a za AC stranu koristi se podzemni kabel PP00 4x10mm^2.

Sl. 3. *Izgled FRONIUS invertora*

5.2. Biogasni sistem

Za dobijanje biogasa iz postrojenja otpadnih voda potrebno je posebno tretirati mulj, koji se taloži u SBR bazenima objekta za biološko prečišćavanje (SBR bazeni i silos za mulj).

Iz silosa za mulj, u kome se nalazi sirovi mulj, potrebno je povećati kocentraciju mulja na 5%. Mulj se pumpama prebacuje do ugušćivača mulja u kome se gravitacijom i zgrtanjem mulja pospešuje proces. Ugušćivač je pokriven i vazduh ispod pokrivača se menja pomoću ventilatorai vodi na lava filtere.

Količina viška aktivnog mulja se mora prepumpavati, i ona se dobija računskim putem da bi se zadržala željena kocentracija.

Ugušćeni mulj se transportuje u digestore pomoću pumpnih stanica. U anaerobnim digestorima mulj truli pri temperaturi od 33°C do 37°C, sa vremenom zadržavanja od 20 dana. Anaerobna digestija smanjuje neprijatne mirise i količinu bakterija u mulju, čineći stabilizovani mulj relativno bezopasnim i neškodljivim.

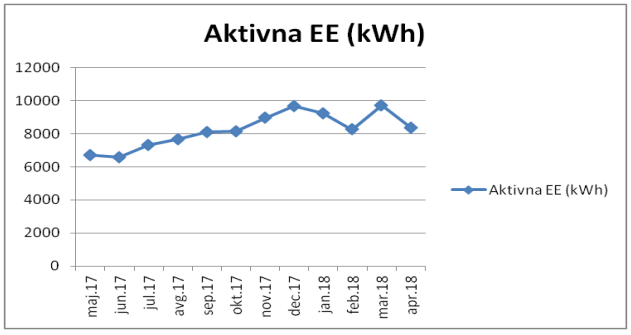
Dobijeni biogas se sastoji od metana (55-70%) i ugljen-dioksida (45-30%). Skladišti se u rezervoaru gasa i koristi kao gorivo kotlovske jedinice za grejanje digestora i objekata, odnosno za potrebe gasnih motora radi proizvodnje električne energije.

Gasni motor spreže se sa sinhronim električnim generatorom i time dobiva agregat, koji vrši proizvodnju električne energije.

6. PROCENA ENERGETSKOG BILANSA

6.1. Procena proizvodnje i potrošnje energije

Procena potrošnje je izvršena na osnovu jednogodišnjih podataka potrošnje električne energije dobijene od JKP "Vodovodi i kanalizacija" Novi Sad (slika 4). Sa grafika se vidi da je potrošnja industrijskog tipa, tj. tokom cele godine je ujednačena, bez velikih odstupanja.



Sl. 4. *Potrošnja PPOV Rumenka*

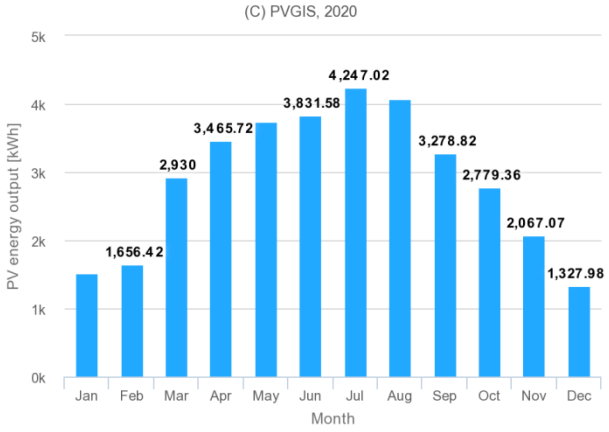
6.2. Procena proizvodnje FN sistema "PVGIS"-om

Photovoltaic Geographical Information System (PVGIS) omogućava procenu proizvodnje sistema u toku jedne godine. Pomoću PVGIS softvera dolazimo do rezultata izračunavanja energetske efikasnosti instalirane solarne elektrane od 25 kW. Sistem je nepokretan, ugao nagiba prema jugu je 36° i orijentacijom od 0°. Procenjeni gubici na osnovu softverske simulacije : gubici usled temperaturnih uticaja – 6,95%, gubici zbog ugaone refleksije – 2,82% i drugi sistemski gubici – 2% ukupni gubici: 11,77%. U tabeli 1 su prikazane vrednosti : prosečna mesečna proizvodnja električne energije sistema (Em) u kWh/m^2, prosečnog sunčevog zračenja na FN modul (Hm) u kWh/m^2 i standardne devijacije (SDm) u kWh [3].

Proizvodnja električne energije dostiže svoj maksimum od 4.247 kWh u toku meseca jula, dok je očekivani minimum proizvodnje u toku decembra, i iznosi 1.328 kWh, što je prikazano na slici 5.

Tabela 1. *Mesečna proizvodnja električne energije, sunčevo zračenje i standardna devijacija*

Mesec	Em	Hm	SDm
Januar	1513,3	61,9	278,4
Februar	1656,4	68,7	410
Mart	2930	125	528,1
April	3465,7	153,9	500,8
Maj	3744,2	170,3	285,5
Jun	3831,6	177,6	298,1
Jul	4247	199,4	308,3
Avgust	4066,4	189,8	458,9
Septembar	3278,8	148,2	480,6
Oktobar	2779,4	120,6	437,1
Novembar	2067,1	86,3	270,1
Decembar	1328	54,8	354,8



Sl. 5. *Procena mesečne proizvodnje električne energije solarnih panela prema PVGIS softveru [3]*

7. TEHNO-EKONOMSKA ANALIZA

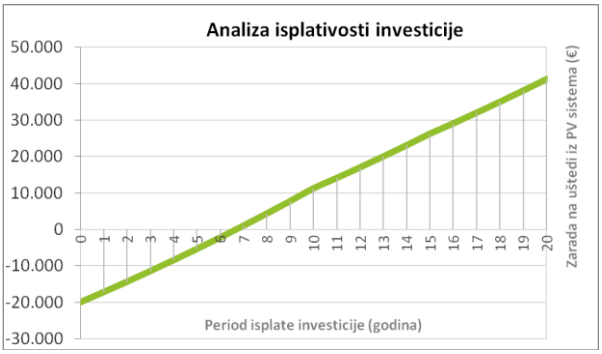
Sagledavši investiciju koja nam je potrebna za izgradnju solarne elektrane došli smo do vrednosti od 22.779,20 €. U tabeli 2 prikazana je potrebna oprema.

Nakon toga izveden je obračun uštede na električnoj energiji prema cenama Elektroprivrede Srbije.

Dobijena razlika vrednosti potrošnje i proizvodnje se koristi za procenu isplativosti investicije. Na slici 6 prikazana je isplativost investicije kroz period od 20 godina.

Tabela 2. *Predmer opreme solarne elektrane*

Materijal	J.M.	Kol.	Cena (€)	Uk. Cena
Solarni panel TT285-60P, 280 Wp	kom	89	95,00	8.455,00
Aluminijumski nosači za montažu	kom	1	2.457,00	2.457,00
Invertor FRONIUS Eco 25.0-3-S	kom	1	2.942,00	2.942,00
DC orman sa zaštitnom opremom	kom	1	758,00	758,00
AC orman sa zaštitnom opremom	kom	1	856,00	856,00
Set kablova i konektora za povezivanje	pauš	1	758,00	758,00
Uređaj za kontrolu toka energije	kom	1	650,00	650,00
Ostali sitan montažni materijal	pauš	1	440,00	440,00
		Ukupno (bez PDV)		17.316,00
		UKUPNO:		**20.779,20**



Sl. 6. *Grafički prikaz isplativosti investicije*

8. ZAKLJUČAK

Obnovljivi izvori energije predstavljaju budućnost energetske industrije, pogotovo solarna energija. Kroz ovaj rad smo videli da proizvodnja električne energije pomoću solarnih panela podiže energetsku efikasnost, i da se relativno brzo isplaćuje kao investicija.

Objašnjen je i princip rada biogasnog postrojenja, međutim kapacitet postrojenja je nedovoljan da bi se takav poduhvat isplatio.
Ostaje prostora za dalju analizu izgradnje biogasnog postojenja, nakon povećanja kapaciteta PPOV Rumenka.

9. LITERATURA

[1] „Global Market Outlook For Solar Power / 2020 – 2024", Solar Power Europe 2020. https://www.solarpowereurope.org/

[2] M. Lambić i dr., "Studija o proceni ukupnog solarnog potencijala – solarni atlas i mogućnosti proizvodnje i korišćenja solarne energije na teritoriji AP Vojvodine", Studija za Pokrajinski sekretarijat za energetiku i mineralne sirovine, Tehički fakultet "M.Pupin", Zrenjanin, 2011.godine, veb sajt: http://www.psemr.vojvodina.gov.rs/index.php/studije/item/8-studije-potencijala-i-mogucnosti-primene-energije-sunca-u-apv

[3] PVGIS softver, http://re.jrc.ec.europa.eu/pvgis/apps4/pvest.php?lang=en&map=europe

Kratka biografija:

Gojko Maričić rođen je u Gračacu, Hrvatska 1987. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Energetska elektronika i električne mašine odbranio je 2013.god., a master rad iz iste oblasti 2020. god.

Vladimir Katič je rođen 1954. god. u Novom Sadu. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu, a magistrirao i doktorirao na Univerzitetu u Beogradu. Od 2002. godine je redovni profesor na Univerzitetu u Novom Sadu. Oblasti interesovanja su mu energetska elektronika, kvalitet električne energije, obnovljivi izvori električne energije i električna vozila.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 620.92
DOI: https://doi.org/10.24867/12BE18Jokovic

TEHNIKE REKONSTRUKCIJE U FOTONAPONSKIM ELEKTRANAMA U SLUČAJU SENČENJA PANELA

RECONFIGURATION TECHNIQUES IN PHOTOVOLTAIC POWER PLANTS IN CASE OF PARTIAL SHADING OF PHOTOVOLTAIC PANELS

Marija Joković, Vladimir Katić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu govori se o problemu smanjenja proizvodnje električne energije u fotonaponskim elektranama uuslovima promenljivog zračenja. Predlaže se metoda analize rada u uslovima senčenja putem isključivanja i prespajanja osenčenih panela u nizu čime se dovodi do povećanja u proizvodnji električne energije.*

Ključne reči: *Fotonapsko pretvaranje, fotonaponski panel, invertor, senčenje, proizvodnja električne energije*

Abstract - *The paper presents the problem of reducing electricity production of PV power plant under variation of solarradiation conditions. It proposed a method of analysis PV power plant works under partial shading condition by switching off and rebinding shaded PV modules of PV array which leads to an increase in production of electric energy.*

Keywords: *Photovoltaic conversion, PV module, inverter, shading, electricity generation*

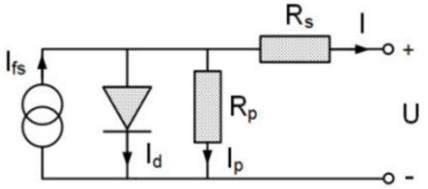
1. UVOD

Solarna energija je najobilniji dostupan vid energije. Prednosti solarnih sistema su prvenstveno ekološkog karaktera: proizvedena energija je čista bez bilo kakvih štetnih nus proizvoda, osnovni energent je neiscrpan i besplatan, vek trajanja je dug (preko 25 godina) uz minimalno održavanje. Velika nedostatak je da nije konstantna tokom dana, kao i da ima promenljivu prirodu tokom cele godine. Negativan uticaj senčenja FN panela najčešći je problem koji se javlja pri radu FN elektrane. Pojava senke na FN panelima drastično smanjuje njihovu snagu i proizvedenu energiju.

Cilj ovog rada jeste smanjenje uticaja promenljivog zračenja na rad FN elektrane FTN1. Zbog toga se predlaže nova metoda analize rada FN elektrane u uslovima promenljivog zračenja. Rezultat istraživanja sa tehničkog aspekta treba da bude smanjenje gubitaka snage u razmatranom funkcionalnom delu FN elektrane FTN1.

2. MODEL FN ĆELIJE I FN PANELA

Ekvivalentna šema fotonaponske ćelije jeste šema jednodiodnog modela, koji uvažava postojanje paralelne i redne otpornosti fotonaponske ćelije kojima se modeluje parazitna struja FN ćelije i kontaktna otpornost i otpornost samog poluprovodnika, respektivno – slika 1. Struja idealnog strujnog izvora u modelu je direktno proporcionalna sunčevom zračenju (modeluje struju saturacije).



Slika 1. *Ekvivalentna šema jedno-diodnog modela fotonaponske ćelije*

Struja ćelije jednaka razlici stuje svetla i struje koja protiče kroz diodu te se mogu definisati dve struje, struja svetla I_{fs} i struja diode I_d gde važi:

$$I = I_{fs} - I_d. \tag{1}$$

Struja kroz diodu je proporcionalna inverznoj struji zasićenja I_0 i ima eksponencijalu zavisnost koja je karakteristična za diodu.

$$I_d = I_0 \cdot \left(e^{\frac{V_d}{n \cdot V_{th}}} - 1 \right), \tag{2}$$

gde je V_d napon na diodi, n faktor idealnosti diode a V_{th} napon održavanja diode dat sa:

$$V_d = U + I \cdot R_s, \tag{3}$$

$$V_{th} = \frac{k \cdot T}{q}. \tag{4}$$

V_{th} je srzmeran temperaturi i zavisi od Bolzmanove konstante (k) i konstante elementarnog naelektrisanja (q). Uvrštavanjem jednačina 2), 3) i 4) u početnu relaciju 1) dobija se zavisnost struje od napona ćelije:

$$I = I_{fs} - I_0 \cdot \left(e^{\frac{q \cdot (U + I \cdot R_s)}{n \cdot k \cdot T}} - 1 \right). \tag{5}$$

Za dalji proračun modela potrebna je vrednost struje kratkog spoja modela $I_{sc(T1)}$ i napon praznog hoda solarnog $U_{oc(T1)}$ izvora. Obe vrednosti se dobijaju ogledom na temperaturi T_1. Aproksimacijom može da se pretpostavi da je struja I_d u praznom hodu panela jednaka struji $I_{sc(T1)}$ tj. $I_d = I_{sc(T1)}$, podrazumevjući da se oba ogleda vrše na temperaturi T_1. Svođenjem (2) za temperaturu T_1 se dobija:

$$I_{sc(T_1)} = I_{0(T_1)} \cdot \left(e^{\frac{q \cdot V_{oc(T_1)}}{n \cdot k \cdot T_1}} - 1 \right). \tag{6}$$

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

Struja I_0 za bilo koju temperaturu se može dobiti u zavisnosti od struje $I_{0(T1)}$, za dobijenu temperaturu T_1, u funkciji od tražene temperature T po relaciji:

$$I_0 = I_{0(T_1)} \cdot \left(\frac{T}{T_1}\right)^{\frac{3}{n}} \cdot e^{\frac{-q \cdot V_g}{n \cdot k}\left(\frac{1}{T}-\frac{1}{T_1}\right)}. \quad (7)$$

gde je V_g (band gap voltage) napon zabranjene zone. Struja osvetljaja za temperaturu T_1 ($I_{fs(T1)}$) je proporcionalna struji kratkog spoja pri nominalnoj osvetljenosti na temperaturi T_1 i odnosom datog nominalnog osvetljaja.

$$I_{fs(T_1)} = \frac{G}{G_{nom}} \cdot I_{sc(T_1,nom)}. \quad (8)$$

Struja osvetljaja I_{fs} za temperaturu T je data relacijom:

$$I_{fs} = I_{fs(T_1)} \cdot \left(1 + K_0 \cdot \left(T - T_1\right)\right), \quad (9)$$

gde je K_0 koeficijent koji predstavlja zavisnost temperatura T_1 i T_2 i dat je izrazom (10):

$$K_0 = \frac{I_{sc(T_2)} - I_{sc(T_1)}}{I_{sc(T_1)} \cdot \left(T_2 - T_1\right)}, \quad (10)$$

a struja $I_{sc(T2)}$ se dobija eksperimentalno, kao $I_{sc(T1)}$ za temperaturu T_2.

Konačna zavisnost struje od napona je data jednačinom (5), s tim da se navedene ralacije parametara neće uvrštavati u krajnji izraz jer bi postao preobiman. Struja I_o se uvrštava u izraz (5) proračunata po (7) a struja I_{fs} iz (9).

Za potrebe modela panela kojim se vrši simulacija zahteva se zavisnost napona od struje, izražavajući jednačinu (5) po struji:

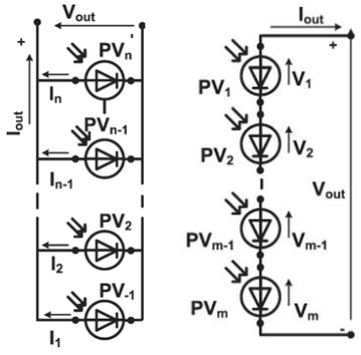
$$I_0 \cdot \left(e^{\frac{q \cdot (U + I \cdot R_s)}{n \cdot k \cdot T}} - 1\right) = I_{fs} - I. \quad (11)$$

Iz 11) se matematičkim operacijama logaritmovnja dobija napon:

$$U = \frac{n \cdot k \cdot T}{q} \cdot \ln\left(1 + \frac{I_{fs} - I}{I_0}\right) - I \cdot R_s. \quad (12)$$

Relacija 12) je konačan izraz zavisnosti struje od napona kojom se matematički modeluje rad solarnog panela [1].

Snaga koju može proizvesti jedna fotonaponska ćelija je relativno mala, te se iz tog razloga veći broj fotonaponskih ćelija povezuje u grupu, čime se formira fotonaponski modul. Na slici 2 prikazana je paralelna i redna veza fotonaponskih ćelija koje čine modul.



Slika 2. *Paralelna i radna veza FN ćelija [2]*

Moduli se povezuju redno kako bi se dobio veći radni napon i paralelno kako bi se dobila veća radna struja. Spajanjem većeg broja fotonaponskih modula dobija se fotonaponski panel. Fotonaponski paneli se uglavnom, povezuju serijski i formiraju FN niz.

3. SENČENJE FOTONAPONSKIH PANELA

Senčenje FN panela je jedan od uzroka smanjenja proizvodnje energije FN niza. Moduli FN niza dobijaju nehomogeno sunčevo zračenje, a time postoji neusklađenost između trenutnih struja modula. Kako osenčeni moduli generišu manje struje od neosenčenih, struja osenčenih modula ograničava izlaznu struju FN niza, a proizvodnja snage delimično osenčenog FN niza se smanjuje. Uticaj senčenja na FN polje nije linearno, što otežava predikciju proizvodnje FN elektrane.

Moduli FN niza izloženi delimičnom senčenju primaju različite nivoe sunčevog zračenja, što može davati višestruke lokalne maksimume na P-U karakteristici. Broj maksimuma zavisi od složenosti senke, odnosno od broja različitog nivoa zračenja na FN panelima. Sa promenom karakteristike FN panela, menja se napon praznog hoda i struja kratkog spoja.

Izlazna snaga FN panela linearno zavisi od zračenja. Istovremeno snaga FN panela zavisi i od temperature panela. Jednačina koja opisuje ove zavisnosti glasi

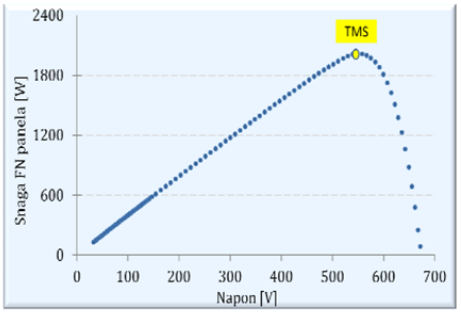
$$P = P_r \frac{G}{G_r}\left[1 + \gamma\left(T - T_r\right)/100\right], \quad 13)$$

gde su:

P_r, G_r i T_r redom snaga, zračenje i temperatura pri nekim uslovima, najčešće referntnim, koje nazivamo standardni uslovi testiranja FN panela. γ temperaturni koeficijent snage, izražen u W/°C [3].

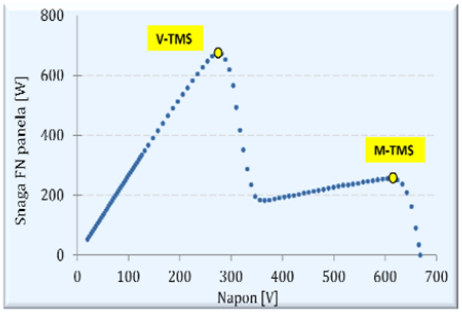
Svako stanje koje prouzrokuje generisanje različite struje i/ili napona FN panela u jednom FN nizu ili polju se naziva neusklađenost električnih parametara FN panela. Značajniji gubici u FN nizu se mogu javiti kada FN niz ili polje rade u uslovima koji dovode do neusklađenosti FN panela (engl. mismatch condition). Razlog tome je promena oblika I-U, odnosno P-U karakteristike, uz pojavu dve ili više tačaka maksimalne snage. Neusklađenost FN panela nastaje i tokom senčenja delova FN niza/polja.

Kod FN panela koji nije u senci postoji samo jedna tačka maksimalne snage, ona je na slici 3 označena sa TMS. Karakteristika FN panela zavisi od trenutne temperature i solarnog zračenja. Zbog toga se TMS panela stalno menja. U situaciji bez senke, nalaženje TMS od strane invertora, bez obzira na stalne promene, ne predstavlja problem.



Slika 3. *P-Ukarakteristika FN niza bez senke [3].*

Kada se FN panel nađe u senci, zbog pojave dva različita nivoa zračenja na FN nizu, najčešće se na P-U karakteristici pojavljuju dva izražena maksimuma snage. Ova situacija je prikazana na slici 4.



Slika 4. *P-U karakteristika FN niza sa senkom [3]*

Po pravilu, broj maksimuma karakteristike FN niza se poklapa sa brojem različitog nivoa senčenja FN niza, a među maksimumima postoji jedan koji je najveći. Ovaj maksimum je obeležen sa V-TMS i naziva se veća tačka maksimalne snage (engl. GMPP, Global Maximum Power Point). Manja tačka maksimalne snage je obeležena sa M-TMS (engl. LMPP, Local Maximum Power Point). U zavisnosti od algoritma za nalaženje TMS invertor će naći ili V-TMS ili M-TMS. Bilo bi poželjno da invertor nalazi V-TMS.

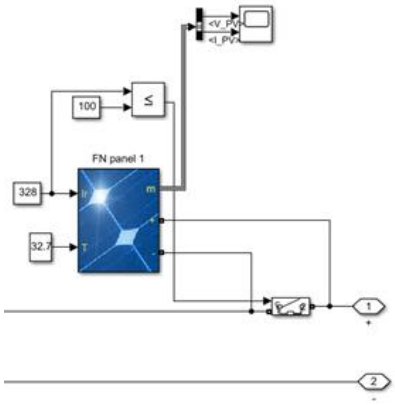
4. MODELOVANJE FN ELEKTRANE FTN1 U MATLABU

Matlab model FN elektrane FTN1 sastoji od više podsistema. Svaki podsistem igra važnu ulogu u povezivanju fotonaponske elektrane na električnu mrežu. Osnovni blokovi su: podsistem niza panela, Boost pretvaraču, MPPT kontroler čija je svrha da se dobije maksimalno iskorišćenje ovog niza i centralnom invertoru na koji se niz povezuje.

Razvijeni model omogućava simulaciju ponašanja FN niza i u nehomogenim ambijentalnim uslovima.

Opisaće se samo podsistem niza panela zbog osobina potrebnih za realizaciju ove metode rekonfiguracije.

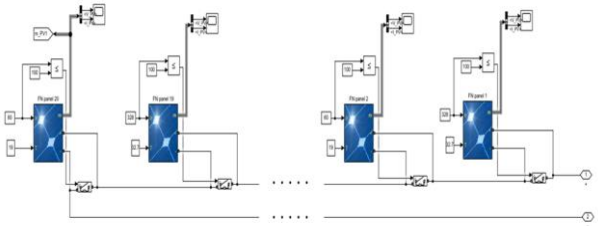
Blok jednog FN panela u MATLAB/Simulink softveru, prikazan je na slici 5. Ulazni parametri bloka su solarno zračenje (G) i temperatura panela (T) Solarno zračenje je osnovni ulazni podatak na osnovu čije promene model daje izlazne veličine kao što su napon, struja i snaga.



Slika 5. *Blok FN panela*

Model FN niza je formira prostom rednom vezom blokova FN panela sa slike 5. Promena broja FN panela u nizu modela postiže se jednostavnim prevezivanjem, a samim tim i eliminacijom određenog broja blokova FN panela. Na slici 6 je prikazan model FN niza od 20 blokova, odnosno FN panela, FN elektrane FTN1.

Razvijeni model omogućava simulaciju ponašanja FN niza i u nehomogenim ambijentalnim uslovima. Za svaki FN panel u nizu može se definisati različito solarno zračenje i temperatura, mada je u praksi najčešći slučaj nehomogenog zračenja situacija sa dva različita para ambijentalnih uslova: [G(1), T(1)] i [G(2), T(2)]. Ako AU označava broj uređenih parova solarnog zračenja i temperature panela [G(x), T(x)], tada za model FN niza sa slike 7.5 važi da je $1 \leq AU \leq 20$. Ako je FN niz pod homogenim uslovima zračenja tada važi da je AU=1 i svi ulazi za zračenje i temperaturu u blok FN panela su jednaki: G(1)=G(2)=...=G(20) i T(1)=T(2)=...=T(20). U slučaju nehomogenog zračenja važi da je AU≠1i maksimalan broj različito uređenih parova za prikazani model FN niza je AU=20.



Slika 6. *Redna veza 20 panela niza FN elektrane FTN1*

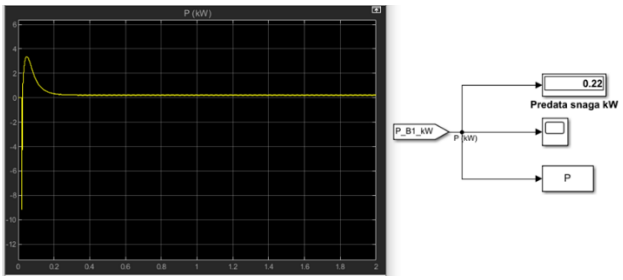
Kako bi u potpunosti omogućili nesmetani rad panela u nehomogenim uslovima, za svaki panel u nizu postavljen je prekidač. Kada panel radi u uslovima slabe ozračenosi tj. kada je panel u senci (zračenje manje od 100W/m^2), to je znak prekidaču da se zatvori. Zatvoreni prekidač prespaja vezu između panela se leve i desne strane panela u senci i panel u senci je prektično izbačen iz rada. Ovime se dobija nesmetani rad elektrane sa panelima u senci koji značajno utiču na pad struje a time i izlazne snage FN elektrane.

5. EKSPERIMENTALNA VERIFIKACIJA REZULTATA PREDLOŽENOGMODELA

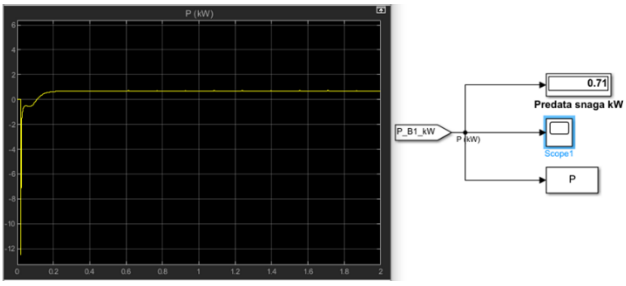
U radu su analizirane dve situacije. U slučaju nehomogenog zračenja menjao se broj FN panela u nizu od minimalne vrednosti - 10 do maksimalne vrednosti - 20. Za svaki broj FN panela u nizu vršena je simulacija pri čemu je za slučaj homogenog zračenja AU=1, a za slučaj nehomogenog zračenja AU=2 (slučaj sa dva para zračenja i temperature [G(1), T(1)] i [G(2), T(2)]). Za slučaj nehomogenog zračenja svi FN paneli izloženi sunčevim zracima imaju isto zračenje i temperaturu, na primer [G(1), T(1)], dok FN paneli u senci imaju drugi uređeni par, na primer [G(2), T(2)]. Da bi simulacioni rezultati bili uporedivi sa merenim vrednostima, koji su se obavljali u letnjem periodu, važi da je G(2)=60W/m^2,a T(2)= 19°C (Za G(1), T(1) je uzeto više vrednosti radi testiranja modela za više sločajeva).

Pošto se u slučaju simulacije sa nehomogenim zračenjem menja broj senčenih i osunčanih panela, uvode se dve nove promenljive. Ukupan broj FN panela u nizu je konstantan i iznosi 20, od čega je *n* u senci a *N=20-n* osunčano. Na slici 7 prikazan je dobijeni grafik i izlazna snaga za prvu simulaciju, kada je 10 FN panela u senci a 10 ozračeno, u slučaju nehomogenih uslova bez rekonfiguracije. Može se primetiti da je invertor našao samo manju tačku maksimalne snage.

Na slici 8 prikazan je dobijeni grafik i izlazna snaga za prvu simulaciju kada je izvršena rekonfiguracija.



Slika 7. *Nehomogeni uslovi bez rekonfiguracije*

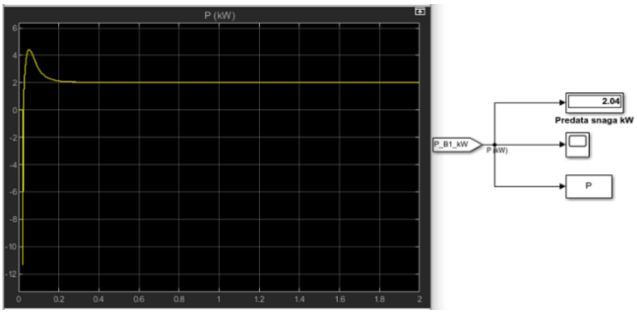


Slika 8. *Nehomogeni uslovi sa rekonfiguracijom*

Izlazna snaga je oko tri puta veća kada se prespoje 10 panela koji su u senci a rad elektrane nastavi sa ostalih 10 redno povezanih panela u niz koji se može smatrati homogenim.

Može uočiti jedan nedostatak na dobijenim graficima, a to je nagli propad snage u početnim trenucima simulacije. Ovo se dešava usled kasnijeg uključenja invertora zbog smanjenog napona koji je posledica smanjenog broja FN panela. Ovo se odražava na dobijenu snagu simulacije tj. zbog smanjenog napona, snaga FN niza je manja, uključenje invertora je kasnije i kraće je vreme generisanja energije.

Uz to invertor sa smanjenim naponom ima manju efikasnost. Nakon nekoliko delova sekunde ova vrednost se ustaljuje na vrednost *M-TMS* kada se simulira slučaj bez rekonfiguracije ili TMS kada se simulira slučaj sa rekonfiguracijom. Na slici 9 prikazan je dobijeni grfik i izlazna snaga za poslednju simulaciju kada je svih 20 panela ozračeno.



Slika 9. *Nehomogeni uslovi sa i bez rekonfiguracije*

Ovo je jedini slučaj kada se dobijaju isti rezultati i sa i bez rekonfiguracije. Razlog toga je što su u ovom slučaju svi paneli ozračeni, tj. nema panelau senci. Rezultati ostalih simulacija prikazanisu u tabeli 1.

Vrednosti maksimalnih snaga kada niz radi u nehomogenim uslovima senčenja bez rekonfiguracije višestruko su manje od vrednosti maksimalnih snaga kada je izvršena rekonfiguracija.

Invertor u slučaju nehomogenog zračenja bez rekonfiguracije nalazi samo M-TMSčime se snaga elektrane značajno smanjuje u odnosu na snagu koja se dobija kada su osenčeni paneli izostavljani iz simulacije.

Prespajanjem FN panela dobija se niz panela sa jednakim ozračenjem i temperaturom gde invertor uvek naplazi TMS, što za rezultat ima višestruko veću snagu.

Tabela 1. *Rezultati dobijeni simulacijama za različiti broj osenčenih panela*

G [W/m^2]	T [°C]	N/n	Nehomogeni uslovi bez rekonfigu-racije	Nehomogeni uslovi sa rekonfigu-racijom
			M-TMS [kW]	TMS [kW]
328	32.7	10 / 10	0.22	0.71
374	35.9	14 / 6	0.23	1.13
398	36.8	16 / 4	0.25	1.37
431	38.6	18 / 2	0.29	1.65
480	40.4	20 / 0	2.04	2.04

6. ZAKLJUČAK

U ovom radu je prikazana praktična primena metode analize rada u uslovima senčenja putem isključivanja i prespajanja osenčenih panela u nizu.Rezultati simulacije su potvrdili zamisao da se može iskoristiti veća snaga fotonaponskog panela kada se umesto projektovanog punog broja FN panela u nizu, koji su pod nehomogenim uslovima senčenja, koristi smanjen broj FN panela u nizu, pri čemu su svi izloženi sunčevom zračenju. Pošto se formira niz koji je homogeno ozračen, karakteristika fotonaponskog niza poseduje samo jedan maksimum, što invertor bez problema nalazi i na taj način se snaga elektrane povećava u odnosu na slučaj kada su paneli u nizu nehomogeno ozračeni, što je u radu i pokazano.

7. LITERATURA

[1] Z. Ivanović, V.A. Katić, *„Obnovljivi izvori električne energije - vežbe“*, Fakultet tehničkih nauka, Novi Sad, 2018.

[2] D. Nguyen and B. Lehman, *„An Adaptive Solar Photovoltaic Array Using Model-Based Reconfiguration Algorithm“,* IEEE Transactions on Industrial Electronics, July 2008.

[3] Z. Čorba, *„Novi metod analize rada fotonaponskog sistema u uslovima varijacije sunčevog zračenja“,* doktorska disertacija, FTN, Novi Sad, 2016.

Kratka biografija:

Marija Joković rođena je u Kladovu 1996. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Energetska elektronika i električne mašine odbranila je 2020.god.

Vladimir Katić rođen je 1954. godine u Novom Sadu. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu 1978. god., a magistrirao i doktorirao na Univerzitetu u Beogradu 1981. i 1991. godine, respektivno. Od 2002. godine je redovni profesor Univerziteta u Novom Sadu. Oblasti interesovanja su 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/10.24867/12BE19Stojanovic

KRATKOROČNA PROGNOZA PROIZVODNJE FOTONAPONSKE ELEKTRANE

SHORT-TERM FORECAST OF PHOTOVOLTAIC POWER PLANT PRODUCTION

Živko Stojanović, Vladimir A. Katić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROENERGETIKA - ENERGETSKA ELEKTRONIKA I ELEKTRIČNE MAŠINE

Kratak sadržaj – *Predmet istraživanja ovog master rada je određivanje, poređenje i ispitivanje najefikasnijih metoda kratkoročnog predviđanja proizvodnje fotonaponskih sistema, u pogledu tačnosti, preciznosti i tehnoekonomske isplativosti. Izabrana metoda je testirana najednoj fotonaponskoj elektrani u Novom Sadu.*

Ključne reči: *metode predviđanja, fotonaponska elektrana, kratkoročno*

Abstract – *The aim of this master's thesis is to determine, compare and test the most efficient methods for photovoltaic energy production short-term forecasting, in terms of accuracy, precision, and techno-economic profitability. The selected method is tested on a photovoltaic power plant in Novi Sad.*

Keywords: *forecasting methods, photovoltaic power plant, short-term*

1. UVOD

Brz napredak industrije i tehnologije proizvodnje fotonaponskih (FN) ćelija poslednjih godina, rezultirao je višestrukim padom cena komponenti FN sistema na tržištu, a takođe i velikim rastom instalisanih kapaciteta, koji su priključeni na elektroenergetski sistem [1,2].

Dalji tehnološki napredak FN sistema usmeren je ka razvoju energetski efikasnijih i jeftinijih FN ćelija (na primer tehnologije tankih filmova, organskih FN ćelija i mineralne tehnologije FN ćelija), kao i ka razvoju pretvaračkih sistema za njihovo povezivanje na mrežu [3,4].

Sa tehno-ekonomskog aspekta, solarna energija je najbrže rastuća obnovljiva tehnologija i sektor sa najvećim stepenom investicija [5]. Ulaganje u solarne elektrane je isplativo jer ova postrojenja imaju izuzetno niske troškove održavanja i ne traže dodatno angažovanje radnika, dugoročno je stabilno i sigurno donosi prihod [6].
Ključni aspekt planiranja rada FN elektrane i njene isplativosti je mogućnost predviđanja proizvodnje, kako dugoročno (mesečno, godišnje), tako i kratkoročno (polusatno, satno). Predviđanje proizvodnje solarne energije zahteva poznavanje položaja Zemlje prilikom rotacije oko Sunca, atmosferskih uslova, karakteristika solarne elektrane, stanja u mreži, posedovanje istorijskih podataka i drugih parametara.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Vladimir A. Katić.

Cilj rada jeste da se razmotri mogućnost uspostavljanja dovoljno dobrog modela predikcije, koji bi mogao da funkcioniše bez ili sa malo meteoroloških podataka (temperatura, oblačnost, vlažnost, itd.) koji zbog svoje brojnosti predstavljaju opterećenje za računarski model, bazu podataka i zahtevaju veoma skup hardver.

2. ZNAČAJ PROGNOZE

Potrebe za preciznom prognozom proizvodnje FN elektrane povećavaju se razvojem FN elektrana i njihovim sve većim udelom u energetskim bilansima. Time se postiže povećanje pozdanosti i smanjenje troškova, sigurnije upravljanje elektroenergetskim mrežama, efikasnije trgovanje električnom energijom dobijenom iz FN elektrane i dr. [7]. Dodatno, predviđanje dovodi do smanjenja broja jedinica u stanju pripravnosti i smanjenju operativnih troškova rada u celom sistemu [8]. Posebno, ono predstavlja mogućnost adekvatnijeg upravljanja u slučaju agregiranja više manjih FN elektrana u virtuelne elektrane, jer se tada može umanjiti negativan uticaj promenljivosti sunčevog zračenja [9,10].

U proteklih nekoliko godina, razvijen je veliki broj metoda i tehnika predviđanja proizvodnje FN sistema. Jedna klasifikacija ovih metoda bazira se na horizontu predviđanja, tj. intervalu vremena između stvarnog vremena i efektivnog vremena predviđanja [10]. Čak i ako ne postoji široko usaglašen kriterijum za klasifikaciju najčešća klasifikacija je sledeća:

- dugoročno (1–10 godina),
- srednjeročno (1 mesec – 1 godina),
- kratkoročno (1 sat – 1 sedmica),
- vrlo kratko (1 minut-nekoliko minuta).

Poslednja dva horizonta su takođe poznata kao '*unutar jednog dana*' [10], dok je najvažniji horizont za prognozu 24 sata sledećeg dana [11].

3. POREĐENJE METODA I NAČIN OCENJIVANJA

Generalno, upoređivanje tehnika predviđanja je izazovno, budući da su faktori koji utiču na performanse brojni i menjaju se u različitim situacijama:

- dostupnost istorijskih podataka i podataka vremenske prognoze,
- vremenski horizont i rezolucija,
- vremenski uslovi,
- geografski položaj, i uslovi instalacije.

U slučaju statističkih metoda, odgovarajuća predobrada podataka (na primer, uklanjanje noćnog uzorka kada nema proizvodnje električne energije) je takođe neophodna kako bi se postigli dobri rezultati i smanjili računski troškovi.

U cilju kratkoročne prognoze proizvodnje električne energije FN sistema, u praksi se najčešće koriste dve grupe metoda MLP (engl. *Multi-Layer Perceptron*) i ANN (engl. *Artificial Neural Network*). Uzimajući karakteristične indekse grešaka, kao indekse performansi metoda predviđanja FN proizvodnje električne energije za dan unapred (prema [12]), kao što su:

- normalizovane srednje apsolutne greške, NMAE%;
- srednje apsolutne procentualne greške, MAPE%,
- izmerene srednje apsolutne greške, WMAE%

upoređene su dve najčešće korišćene grupe metoda MLP i ANN. Rezultati poređenja i odgovarajuća ocena za svaki indeks, dati su u tabeli 1. Može se zaključiti da je MLP superiornija u odnosu na ANN, što ukazuje veća ukupna ocena (tabela1).

Uzimajući kriterijume, odnosno indekse performansi navedenih u prethodnom poglavlju, zaključenoje da MLP grupa metoda pokazuje bolje performanse u odnosu na ANN. U ovom master radu, biće izvršeno formiranje modela za prognozu srednje snage za jedan sat unapred, na osnovu višestruke linearne regresije (MLR), koja pripada MLP grupi metoda. Ona je trenirana putem metode veštačke inteligencije, što omogućuje dobre adaptivne karakteristike, prilagođavanje na nove situacije, samoučenje i veliku brzinu rada. Ono što je jedinstveno je da pokazuje veliku preciznost predviđanja sa redukovanim setom podataka (na primer, bez meteoroloških podataka), što znatno smanjuje troškove.

4. PRIMENA MLR METODE NA TEST PRIMERU

Metodologija za kratkoročnu prognozu proizvodnje FN elektrane ilustrovana je na realnom primeru elektrane u Novom Sadu, uporebom metode višestruke linearne regresije (MLR). Kao primer FN elektrane u Novom Sadu Instalisana snaga elektrane iznosi P_{inst} = 140 kW a elektrana je montirana na zemlji, u okviru jedne radionice za obradu mašinskih delova.

Tabela 1. *Evaluacija indeksa performansi za MLP i ANN metode za šest sunčanih dana*

Indeks performansi	**MLP**	**ANN**
NMAE%	2%	varij.
Ocena	1	0
MAPE%	23.6%	10,0%
Ocena	1	1
WMAE%	10,1%	12,4%
Ocena	1	0
Ukupna ocena	**3**	**1**

Cilj primene MLR je bio da se ispita da li može da se uspostavi dovoljno dobar model za predikciju, koji bi mogao da funkcioniše bez meteoroloških podataka (temperatura, oblačnost, vlažnost itd.).

Drugim rečima, izvršeno je ispitivanje višestruke linearne regresije kao modela za predviđanje satnih profila proizvodnje, u okviru koga se utvrđuje zavisnosti narednog sata planirane proizvodnje sa ostvarenom proizvodnjom iz prethodnog sata, uz korelaciju za svaki mesec rada.

Zbog toga je korišćen redukovani skup podataka koji se sastoji samo od hronoloških podataka (sat u toku dana i mesec u toku godine). Kao dodatna prediktorska varijabla korišćena je snaga elektrane u prethodnom satu. Kako se prognoza vrši za jedan sat unapred, ova varijabla je ključna za određivanje prognozirane snage.

Za razliku od prognoze potrošnje, gde u velikoj meri na prognozu utiču i drugi parametri, kao što su dan u nedelji, tip dana (da li je radni ili neradni), kao i temperatura od koje jako zavisi potrošnja električne energije za grejanje i hlađenje, u ovom modelu korišćene su samo tri prediktorske promenljive.

Naravno, model bi sigurno bio tačniji kada bi se sagledala i temperatura i brzina vetra. Ove veličine jako utiču na temperaturu panela, a od nje opet, zavisi i proizvedena snaga panela.

Primenom formule (1) izvršeno je treniranje modela i proračun koeficijenata regresije.

$$Q = \sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_{i1} - \beta_2 X_{i2} - \cdots - \beta_{p-1} X_{i,p-1})^2 \quad (1)$$

gde su:
Q - minimiziranje sume kvadrata,
Y_i - *i*-ta vrednost zavisne promenljive,
β_{p-1} - odgovarajući regresioni koeficijent,
β_0 - odsečak, koji predstavlja vrednost zavisne promenljive u slučaju kada su sve nezavisne promenljive jednake nuli ,
$X_{i,p-1}$ *i*-ta vrednost odgovarajuće promenljive.

Praktično, treniranje modela izvršeno je na celom skupu podataka, jer nije bilo podataka za više godina unazad. Koeficijenti regresije određeni su minimizacijom sume kvadradtnih odstupanja, korišćenjem programa *Matlab* i funkcije *fitlm*. Korišćenjem ove funkcije i programskog koda koji je dat u dodatku ovog rada, dobijaju se koeficijenti regresije dati u tabeli 2.

Model pokazuje dobro slaganje sa tačnim vrednostima prikazano u tabeli 2, budući da je vrednost koeficijenta korelacije vrlo visoka: R=0,943.

Srednja kvadratna greška (RMSE - engl. *Root Mean Square Error*) odstupanja može se primeniti na različite modele prognoze kako bi se odredila tačnost prognoze. Ova veličina definiše se izrazom (2):

$$RMSE = \sqrt{\frac{\sum(x_i - \bar{x}_i)^2}{n}} \quad (2)$$

gde x_i i $\bar{x}_i$ predstavljaju stvarnu i prognoziranu vrednost na *n* uzoraka.

Normalizovana RMSE vrednost (engl. *Normalized RMSE*) računa se na osnovu sledećeg izraza:

$$NRMSE = \frac{1}{P_{max}} \sqrt{\frac{\sum(x_i - \bar{x}_i)^2}{n}} \quad (3)$$

Tabela 2. *Koeficijenti regresije*

Estimated coeficients:

	Estimate	SE	tStat	pValue
(Intercept)	0.053962	0.40869	0.13204	0.89496
Month	-0.0082844	0.02256	-0.36721	0.71347
Time_01:55	3.60E-14	0.53932	6.68E-14	1
Time_02:55	3.15E-14	0.53932	5.84E-14	1
Time_03:55	0.2988	0.53932	0.55403	0.57957
Time_04:55	2.4002	0.53932	4.4503	8.68E-06
Time_05:55	8.0757	0.53945	14.97	4.75E-50
Time_06:55	15.551	0.54123	28.733	1.37E-173
Time_07:55	19.146	0.55014	34.803	1.09E-248
Time_08:55	18.298	0.56916	32.149	1.87E-214
Time_09:55	13.578	0.59249	22.916	6.38E-113
Time_10:55	6.4749	0.60906	10.631	3.09E-26
Time_11:55	0.41662	0.61086	0.68203	0.49524
Time_12:55	-3.9569	0.60058	-6.5884	4.70E-11
Time_13:55	-8.1882	0.58427	-14.014	3.81E-44
Time_14:55	-11.751	0.56558	-20.776	1.25E-93
Time_15:55	-11.283	0.54948	-20.534	1.51E-91
Time_16:55	-6.7928	0.54142	-12.546	8.44E-36
Time_17:55	-2.3426	0.53951	-4.342	1.43E-05
Time_18:55	-0.57195	0.53933	-1.0605	0.28896
Time_19:55	-0.049803	0.53932	-0.092344	0.92643
Time_20:55	-2.65E-16	0.53932	-4.92E-16	1
Time_21:55	9.11E-15	0.53932	1.69E-14	1
Time_22:55	-1.50E-15	0.53932	-2.79E-15	1
Time_23:55	2.26E-05	0.53932	4.20E-05	0.99997
v	0.91499	0.0043114	212.22	0

Oznake u tabeli 2 imaju sledeće značenje:

- Estimate — vrednost koeficijenta u modelu.
- intercept – konstanti član linearnog modela
- SE — standardna greška
- tStat — t-statistika (ili Studentova statistika) za svaku od pretpostavki da je odgovarajući koeficijent različit od nule.
- pValue — p-vrednost za t-statistiku hipoteze da je odgovarajući koeficijent jednak nuli ili ne.

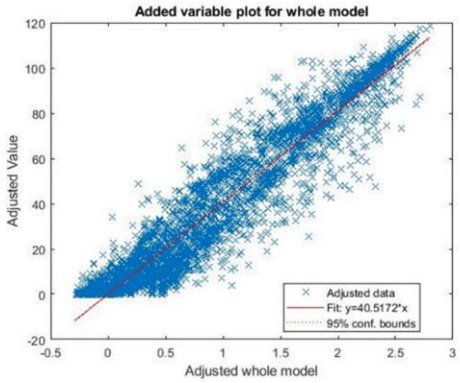
Srednja kvadratna greška u ovom slučaju iznosi 7,3 kW ili ako se normalizuje na vrednost podeljenu sa maksimalnom snagom elektrane NRMS = 5,2%.

Tačnost sunčeve prognoze zavisi od regiona. Na primer, RMSE su se kretali od oko 20% do 35% u Španiji, dostižući 40% do 60% u Centralnoj Evropi [13].

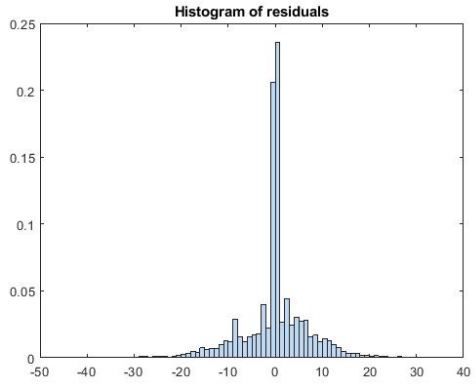
Pokazuje se da se tačnost prognoze značajno se poboljšava kako se povećava geografska oblast koja se razmatra povećava. Dijagrami odstupanja procenjenih od stvarnih vrednosti, kao i histogram vrednosti reziduala prikazani su na slikama 1 i 2.

Karakteristični dnevni dijagrami proizvodnje za jedan zimski (04.01.2018.) i jedan letnji dan (21.08.2018.) dati su na slikama 3 i 4.

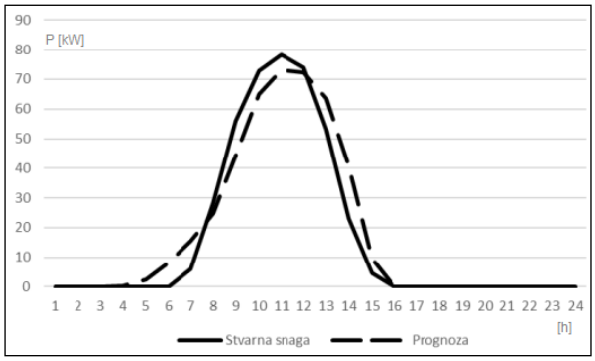
Iz dijagrama datih na slikama 3 i 4 vidljivo je da su najveća odstupanja u periodima najvećih snaga. Takođe, veća su odstupanja zimi, nego leti, kao i pri promenama solarnog zračenja. Ovo se može objasniti činjenicom da je u zimskim mesecima mnogo veća varijacija snage u toku meseca, te je i trening podataka podložan većem broju grešaka.



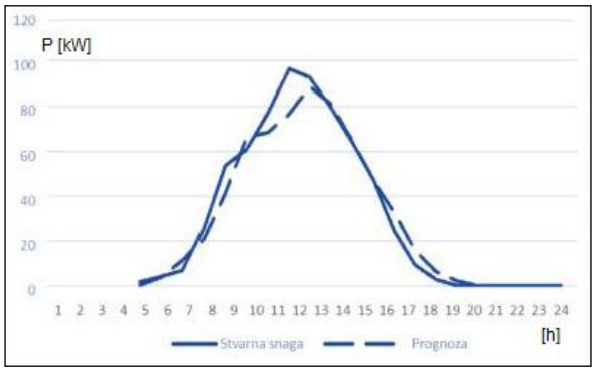
Sl. 1. *Odstupanja procenjenih od tačnih vrednosti*



Sl. 2. *Histogram reziduala*



Sl. 3. *Satna proizvodnja elektrane za karakterističan zimski dan*



Sl. 4. *Satna proizvodnja elektrane za karakterističan letnji dan*

Na tržištu električne energije greška u prognozi može da ima velike finansijske posledice ukoliko znatno odstupa od prijavljenih količina proizvodnje za dan unapred. Pod uslovom da proizvođač snosi balansnu odgovoronost na

tržištu, sva odstupanja plaća po cenama poravnanja koje određuje operator prenosnog sistema (Elektromreže Srbije - EMS u Srbiji) [14].

Tehno-ekonomskom analizom [14] u toku jednog dana, vrednost penala usled loše prognoze može da iznosi i više od 500 evra, što na godišnjem nivou, čak i za tako malu elektranu predstavlja veliki iznos. Ova analiza je uprošćena, jer se ne vodi računa o unutardnevnom tržištu niti o dozvoljenom odstupanju elektrane, ali je dovoljno ilustrativna da pokaže važnost tačne prognoze proizvedene energije.

5. ZAKLJUČAK

U elektroenergetskom sistemu, zbog svojih tehnološko-tehničkih svojstava, solarni FN sistemi su komercijalno najzastupljenija tehnička rešenja koja apsorbuju i konvertuju energije sunca direktno u električnu energiju korišćenjem modularnih panela i skalabilnih invertora za transformaciju jednosmerne struju u naizmeničnu, a koji mogu da se prilagode različitim potrebama.

Skalabilnost sistema i cenovna pristupačnost tehnologije uticalo je na stvaranje ogromnog tržišta na svetskom nivou i razvoju podsticajnih mehanizama kroz regulatorno-ekonomske okvire za omasovljavanje primene FN sistema prvenstveno na nivou distributivnih mreža, gde se sama solarna elektrana postavlja u neposrednoj blizini potrošača u mreži (distribuirane elektrane). Instalacijom distribuiranih elektrana, pored ekološkog efekta, uticalo bi se pozitivno na smanjenje termičkih gubitaka koji se javljaju pri prenosu i distribuciji električne energije, čime se produžava životni vek infrastrukture.

U Srbiji, najveći izazov u primeni modela za predviđanje potrošnje, samim tim i proizvodnje, jeste dostupnost podataka. Komercijalno dostupna softverska rešenja na koja se mogu osloniti prvenstveno snabevači električne energije su efikasna, ali i skupa i mogu zadovoljiti potrebe previđanja proizvodnje električne energije iz solarnih elektrana, jer dovode u korelaciju spoljne uslove: iradijaciju na specifičnoj lokaciji (kW/m^2), temperaturu, vlažnost vazduha, itd. u spregu sa instalisanom snagom elektrane i tehničkim karakteristikama opreme. Međutim, raspolaganje istorijskim podacima za više hidrometeoroloških parametara za više lokacija nije zanemarljiv trošak. U ovom radu je dokazano je da je moguće koristiti modele za predikciju sa ograničenim setom vrednosti, koji bi se bazirali samo na istorijskim podacima o proizvodnji, sezonalnim ili mesečnim razgraničavanjem, koja je ujedno prihvatljiva snabdevaču ili elektrani kao balansnoj strani za troškove balansiranja.

6. LITERATURA

[1] M.A.Green, K. Emery, Y. Hishikawa, W. Warta, E. D. Dunlop, Solar cell efficiency tables (Version 45), Progress in Photovoltaics, 23 (1), 2015, pp. 1-9;

[2] Global market outlook for photovoltaics until 2014, 2014, EPIA;

[3] A.M. Ismail, R. Ramirez-Iniguez, M. Asif, A. B. Munir, F. Muhammed-Sukki, Progress of solar photovoltaic in ASEAN countries: A review, Renewable and Sustainable Energy Reviews, 48 (2015), pp. 339-412

[4] Global market outlook for photovoltaics 2014-2018, EPIA

[5] Global landscape of Renewable Energy Finance. IRENA, 2018.

[6] Reikard G. Predicting solar radiation at high resolutions: a comparison of time series forecasts. Solar Energy 2009; 83: 342–9.

[7] Connecting the Sun-Solar Photovoltaics on the Road To Large Scale Grid Integration. EPIA, 2012.

[8] Antonanzas, J.; Osorio, N.; Escobar, R.; Urraca, R.; Martinez-De Pison, F.; Antonanzas-Torres, F. Review of photovoltaic power forecasting. Sol. Energy 2016, 136, 78–111.

[9] Antonanzas, J.; Osorio, N.; Escobar, R.; Urraca, R.; Martinez-De Pison, F.; Antonanzas-Torres, F. Review of photovoltaic power forecasting. Solar Energy 2016, 136, 78–111.

[10] Mills, A.; Wiser, R. Implications of Wide-Area Geographic Diversity for Short-Term Variability of Solar Power; Technical Report LBNL-3884E; Lawrence Berkeley National Laboratory: Washington, DC, USA, Sept. 2010.

[11] B. Parida, S. Iniyan, R. Goic, A review of solar photovoltaic technologies, Renewable and Sustainable Energy Reviews, 15 (2011), 3, pp. 1625-1636.

[12] Nespoli, A.; Ogliari, E.; Leva, S.; Massi Pavan, A.; Mellit, A.; Lughi, V.; Dolara, A. Day-Ahead Photovoltaic Forecasting: A Comparison of the Most Effective Techniques. Energies 2019, 12, 1621.

[13] IEA (2020), Solar PV, IEA, Paris https://www.iea.org/reports/solar-pv

[14] Agencija za energetiku Republike Srbije; dostupno na: https://www.aers.rs/

Kratka biografija:

Živko Stojanović rođen je u Bijeljini 1994. god. Gimnaziju "Filip Višnjić", završio je u Bijeljini, 2013 god. Diplomirao je na Fakultetu tehničkih nauka, studijski program Energetika, elektronika i telekomunikacije 2017. god. Od 2017. god. upisuje master akademske studije Elektroenergetika - Energetska elektronika i električne mašine na svom matičnom fakultetu.

Vladimir A. Katić, red.prof. rođen je 1954. godine u Novom Sadu. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu 1978. god., a magistrirao i doktorirao na Univerzitetu u Beogradu 1981. i 1991. godine, respektivno. Od 2002. godine je redovni profesor Univerziteta u Novom Sadu. Oblasti interesovanja su energetska elektronika, obnovljivi izvori električne energije, kvalitet električne energije i električna vozila.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.314
DOI: https://doi.org/10.24867/12BE20Milanko

МОДЕЛОВАЊЕ ЕКОЛОШКЕ ПУНИОНИЦЕ ЗА ЕЛЕКТРИЧНА ВОЗИЛА

MODELING OF ECOLOGICAL CHARGING STATION FOR ELECTRIC VEHICLES

Марко Миланко, Владимир Катић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У раду је представљен модел еколошки прихватљиве фотонапонске пунионице са супербрзим пуњачем за електрична возила. Моделвање је извршено у MATLAB/Simulink програмском пакету.*

Кључне речи: *Електрични аутомобили, Li-ion батерије, Пуниоица, Фотонапонска електрана*

Abstract – *This paper presents a model of an environmentally friendly photovoltaic charging station with a superfast charger for electric vehicles. The modeling was performed in the MATLAB/Simulink software package.*

Keywords: *Electric vehicles, Li-ion batteries, Charging station, Photovoltaic power plant*

1. УВОД

Уз електрификовану флоту путничких аутомобила, емисија угљен-диоксида могла би се значајно смањити у зависности од начина производње електричне енергије. Повећана потрошња електричне енергије због потреба за пуњењем електричних аутомобила у возним парковима може се задовољити повећаном количином производње обновљиве енергије у електроенергетским системима. Са возним парком електричних возила у транспортном систему настаје потреба за успостављањем инфраструктуре за пуњење која ће дистрибуирати снагу електричним возилима.

Зависно од броја електричних возила и начина пуњења, интеграција електричних возила укључује додатне количине захтеване енергије на укупни профил оптерећења што може довести до повећања пикова на дијаграмима оптерећења. Обрасци пуњења електричних возила стохастичког карактера су јер на њих утиче индивидуално понашање возача на путу као и расположиве могућности за пуњење што потврђује претпоставку да ће увођење електричних возила утицати на промене оптерећења.

Повећана варијација оптерећења и пикови могу створити потребу за надоградњом мрежне инфраструктуре ради превенције у смањивању губитака, ризика од преоптерећења или оштећења компоненти. Међутим, уз добро испланиране подстицајне мере за кориснике електричних возила, уз одговарајућу инфраструктуру пуњења, електрична возила се могу користити као флексибилна оптерећења која могу помоћи у ублажавању варијација оптерећења и вршних оптерећења у електроенергетском систему.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Владимир Катић, ред. проф.

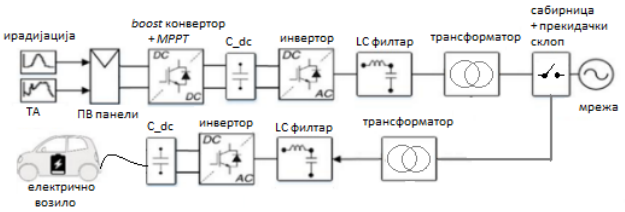
2. ПУНИОНИЦА

Систем еколошки прихватљиве пуниоице састоји се од неколико важних целина. Соларна електрана се састоји од четири соларна поља и она се прикључује на остатак система преко *DC/DC*претварача, који је моделован као подизач напона и чија је улога да на свом излазу да напон и струју одговарајућег квалитета. То изводи помоћу *MPPT* алгоритма [1,2]. Следећи елемент је инвертор који једносмерне електричне величине, које на његов улаз долазе преко подизача напона, претвара у наизменичне и тако синхронизује соларну електрану са остатком система.

После њега се налазе *LC* филтар који иа улогу да обезбеди правилну таласност изалзног напона и струје, и трансформатор, који подиже напон ради ефикаснијег транспорта енергије од извора до пуњача и/или мреже. Други део система је модел самог супербрзог пуњача и електричног возила које је моделовано једном батеријом.

Пуњач се састоји од кондензаторског филтера, инвертора чија је улога, поред трансформације наизменичних величина у једносмерне, и контрола пуњења батерије. Такође, и овде су присутни *LC* филтар и трансформатор. Трећи подсистем представља електричну мрежу бесконачне снаге.

Сва три подсистема вежу се на сабирницу у склопу које се налази и прекидачки механизам помоћу којег се управља пуниоицом, тј. врши избор мода за пуњење. Блок шема овог система представљена је на слици 1.



Слика 1 *Блок шема система*

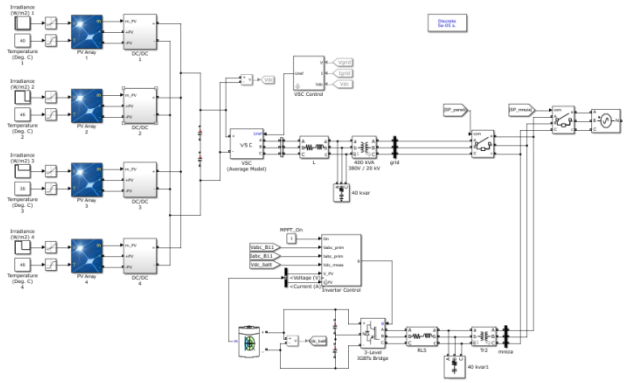
Систем је предвиђен за рад у једном од следећа три режима:

1) Ноћ, потпуно засенчење панела или квар неког/их од елемената у соларној електрани. Целокупна енергија потребна за пуњење батерије електричног возила се добија из електричне мреже.
2) Пуњач за електрична возила се напаја из соларне електране, без обзира на тренутне промене ирадијације и
3) Пуњач за електрична возила се напаја из соларне електране, а при томе постоји механизам за повезивање пуњача самрежом у случају смањивања

излазне снаге електране испод задате вредности, у овом случају задата је граница од 50%P_n.

3. MATLAB/*Simulink* МОДЕЛ СИСТЕМА

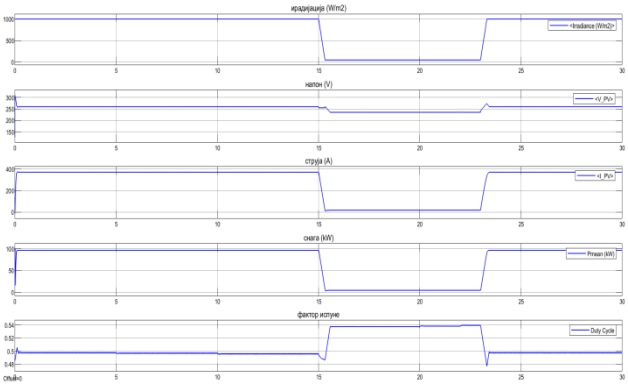
За моделовање еколошки прихватљивог система превоза коришћен је софтверски алат *MATLAB/Simulink*. Целокупан изглед модела приказан је на слици 2, са које се може уочити да се модел састоји од више подсистема. Сваки подсистем игра важну улогу у реализацији ове симулације.



Слика 2. *Модел система у MATLAB/Simulink програму*

4. РЕЗУЛТАТИ СИМУЛАЦИЈЕ

На слици 3. на првом дијаграму је дата вредност ирадијације доведене на улазе модела фотонапонских панела првог соларног поља. Дијаграми ирадијације разликују се за сва четири соларна поља из разлога што је постојала намера да се симулира различита осунчаност на различитим локацијама где су соларна поља инсталирана, односно да сенка може пасти на једно или део једног соларног поља, а не мора и на све остале. Биће представљени само дијаграми првог соларног поља јер су остала три аналогна, само са различитим тренутним вредностима ирадијације, а онда последично и напона и снаге итд.

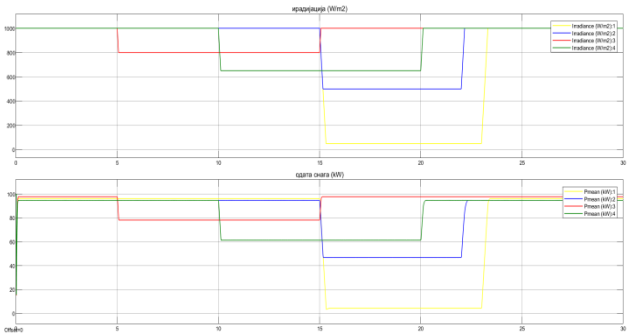


Слика 3. *Ирадијација, напон, струја, снага и фактор испуне првог соларног поља*

Са дијаграма се уочава да ирадијација има вредност од 1000 W/m^2до петнаесте секунде, након чега линеарно опада до вредности од 50 W/m^2 до шеснаесте секунде и задржава ту вредност све до двадесет и треће секунде. Тада почиње линеарно да расте до двадесет и четврте секунде и вредности од 1000 W/m^2 коју задржава до краја симулације, односно до тридесете секунде.

На другом и трећем дијаграму слике 3. приказани су напон и струја соларног поља, са којих се може видети да при негативној промени вредности осунчаности (од 15. до 24. секунде) деградирају и вредности напона и струје, што за последицу има и смањену вредност снаге коју производи соларно поље и што се може видети на четвртом дијаграму. Промена вредности фактора испуне приказана је на последњем дијаграму.

На слици 4. приказана су два графика. На првом је приказана вредност ирадијације сваког од четири соларна поља у свакој од 30 секунди симулације. Жутом бојом је обележена вредност осунчаности првог соларног поља, плавом бојом осунчаност другог соларног поља, док су црвеном и зеленом бојом обележене вредности ирадијације трећег и четвртног соларног поља, респективно.

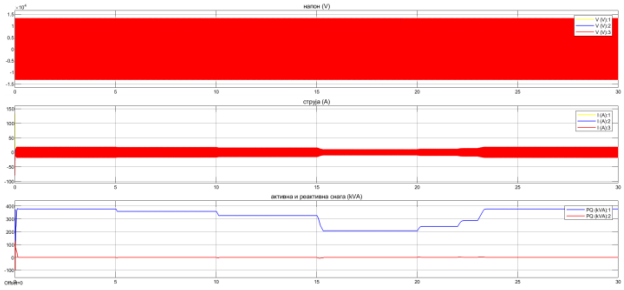


Слика 4. *Зависност излазне снаге соларних поља од ирадијације*

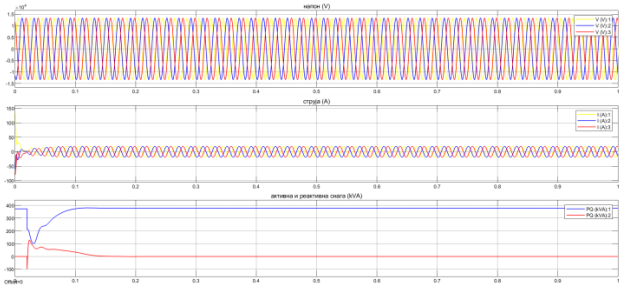
Са првог графика уочава се да је осунчаност од 1000 W/m^2на сва четири соларна поља присутна само у првих 5 и последњих 8 секунди, и стога је само у та два периода соларна електрана у могућности да испоручи номиналну снагу од 400 *kW*.

На другом графику су приказане вредности активне снаге сваког појединачног соларног поља у времену. Боје за свако соларно поље и снагу које оно испоручава су аналогне са онима на графику изнад, за вредности ирадијације. Уочава се да у периоду од петнаесте до двадесет и друге секунде соларна електрана испоручује испод 50% номиналне снаге (195 *kW*) и то је уједно и „најкритичнији“ период током којег се значајно успорава процес пуњења.

На ликама 5. и 6. приказани су напон, струја као и активна и реактивна снага мерени на излазу соларне електране, после трансформатора. Уочава се како вредности струје варирају у односу на тренутну снагу соларне електране, док се напон одржава константним. На слици 5. приказане су величине у зависности од времена за цео период симулације, односно 30 секунди, а на слици 6. је узет период од једне секунде како би се могли уочити таласни облици струје и напона. Поред тога, на слици 6. су уочљиви и прелазни процеси на самом старту симулације, у периоду од неколико стотинки, због присутности уређаја енергетске електронике велике снаге у систему. Активна снага представљена је плавом бојом, а реактивна црвеном.

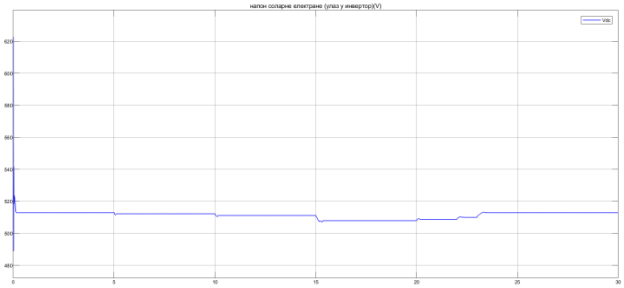


Слика 5. *Напон, струја, акривна и реактивна снага на излазу соларне електране*



Слика 6. *Напон, струја, активна и реактивна снага на излазу соларне електране, увећано*

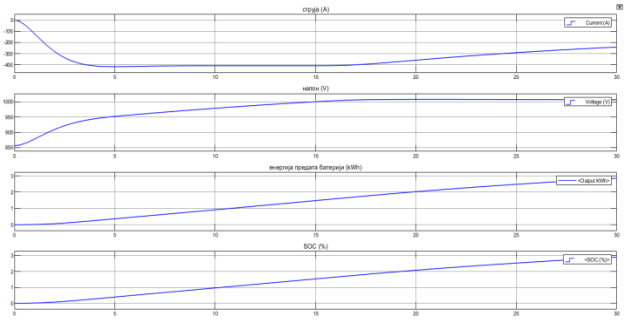
На слици 7. приказан је напон једносмерног кола соларне електране, односно напон мерен на излазу из подизача напона, односно напон који се доводи на улаз инвертора. Овај напон је добијен деловањем *MPPT* контролера и подизача напона комбиновано и мора се одржавати у одговарајућим границама од 500 *V* (+/- 4%). На овом напону се такође примећују последице смањења и повећања осунчања у виду пропада и пикова напона. Пропади и пикови присутни у тренуцима промене ирадијације, у петој, десетој, петнаестој итд. секунди присутни су из разлога што је примењен усредњени модел једносмерног претварача и *P&O MPPT* алгоритма, који нису у стању довољно брзо да испрате нагле промене ирадијације.



Слика 7. *Напон соларне електране мерен на улазу у инвертор*

4.1 Режим 1)

На слици 8. виде се четири графика. На њима су представљене промене струје, напона, енергије и стања напуњености батерије(*SoC– State of Charge*)) током 30 секунди симулације. На првом графику се види да струја добива негативну вредност, а то је из разлога што се батерија пуни, односно смер струје је из система у батерију. Да је батерија та која испоручује енергију систему, вредност струје би била позитивна. На старту симулације струја расте до вредности од 400 *A* и одржава се константном, према *CCCV* [3] алгоритму (прво се струја одржава константном, а затим напон), док напон не достигне задату вредност, у овом случају 1020 *V*. У периоду док се струја одржавала константном, напон је растао. Када је достигао задату вредност, отпочела је друга *(CV)* [3] фаза пуњења, током које се напон одржава константним, док струја пада.

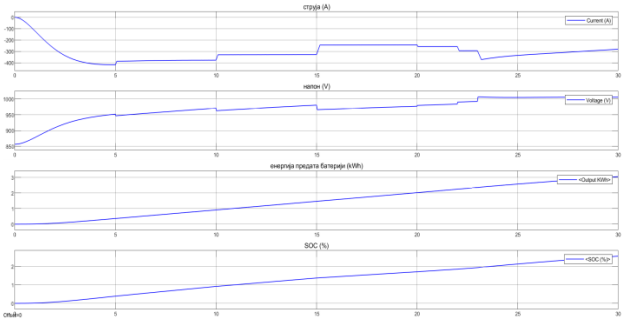


Счика 8. *Струја, напон, енергија и SOC батерије у процесу пуњења 1)*

На другом графикону приказана је енергија коју пуњач испоручује батерији. У овом случају, када се пуњач снабдева директно из електричне мреже, за период од 30 секунди симулације, укупна енергија предата батерији достигла је вредност од 3.046 *kWh*. И, на последњем графику приказано је стање напуњености батерије, које је у овом сценарију, дакле када се узима снга из електричне мреже, достигло вредност од 2.894 %.

4.2 Режим 2)

У овом случају, на слици 9. уочавају се пропади струје и напона сразмерно са наиласком сенке преко соларних поља, односно са смањењем тренутне излазне снаге соларне електране. Могу се уочити сразмерни пропади струје и напона као последица засењења сваког соларног поља појединачно. Од пете до петнаесте секунде соларно поље три (црвена линија на графику са слике 4) одаје 20 % мање санге. Затим од десете до двадесете секунде соларно поље четири (зелена линија на графику са слике 4) одаје 40 % мање снаге. Од петнаесте до двадесет и друге секунде соларно поље два (плава линија на графику са слике 4) одаје 55 % мање снаге од номиналне и од петнаесте до двадесет и треће секунде соларно поље број један (жута линија на графику са слике 4) одаје 95 % мање снаге од номиналне.



Слика 9. *Струја, напон, енергија и SOC батерије у процесу пуњења 2)*

На друга два графика са слике 9. се поново налазе тренутне вредности енергије испоручене батерији и стања напуњености батерије, чије криве у овом случају расту мање стрмо. Такође је могуће уочити благе промене нагиба криве стања напуњености у

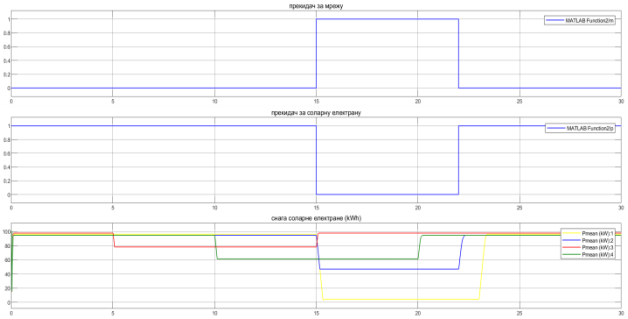
тренуцима промене осунчаности соалрних поља (пета, десета, петнаеста, двадесета, двадесет друга и двадесет и трећа секунда).

У овом сценарију, енергија коју је пуњач предао батерији за период од 30 секунди, износи 2.857 *kWh*, а батерија је напуњена до 2.562 % свог максималног капацитета.

Може се приметити како је батерија мање напуњена у односу на први сценарио, када је пуњена из мрже, што свакако представља недостатак оваквог једног система, који има за примарни циљ пуњење електричног возила из еколошки чистог извора, соларне електране. Рачуном се долази до чињенице да би се за исто време које потребно да се батерија напуни од 0 до 100 % преко електричне мреже, преко соларне електране напунила до 88 % свог капацитета, у условима овакве осунчаности. Ових 12 % разлике представља практичан недостатак у случају другог сценарија и он се може делимично решити постављањем прекидачког алгоритма у систем, у циљу унапређивања перформанси пуњача када се он напаја из соларне електране.

Прекидачки алгоритам је осмишљен тако да ради на принципу укључивања/искључивања одређених делова система у зависности од тренутне вредности иридијације или излазне снаге соларне електране. Када тренутна излазна снага соларне електране падне на испод 50 % њене номиналне вредности, прекидачки склоп аутоматски прикључује пуњач на мрежу. У пракси би се још додала опција да се тада соларна електрана прикључује на мрежу или стационарну батерију, уколико постоји, како се не би дозволио губитак енергије од 50 или мање процената од њене номиналне.

На слици 10. је приказан принцип рада прекидачког склопа, чији је алгоритам реализован у ексетрној *MATLAB* функцији.

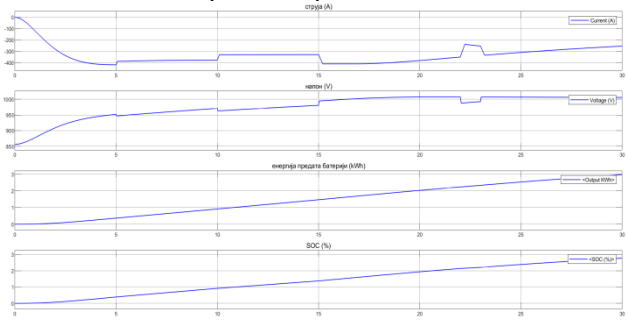


Слика 10. *Уклопна стања прекидача у зависности од тренутне излазне снаге соларне елктране*

Применом оваквог решења, могуће је унапредити претходни случај. Имплементацијом прекидачког склопа дошло се до сценарија број три.

4.3 Режим 3)

Прва два графика са слике 11 разликују се од оних на слици 9 само у временском периоду од петнаесте до двадесет и друге секунде. Вредности напона и струје не разликују се од оних приказаних на силици 9. првих петнаест и последњих осам секунди симулације. Разлика је дакле видљива у тих седам секунди и она се огледа у повећању вредности напона и струје на вредности које смо имали у случају првог сценарија, тј. када је пуњач црпио енергију искључиво из електричне мреже. У том периоду вредности су једнаке управо из разлога што се у овом случају, дејством алгоритма прекидачког склопа, пуњач прикључио на мрежу и тиме су санирране последице превеликог пада тренутне излазне снаге соларне електране.



Слика 11. *Струја, напон, енергија и SOC батерије у процесу пуњења 3)*

5. ЗАКЉУЧАК

Рад описује моделе и симулације рада еколошки прихватљиве фотонапонске пунионице за електрична возила у три предвиђена режима. Дати су резултати низа симулација за сваки од ова три режима, а претходно су приказани резултати симулације модела саме соларне електране. Резултати су коментарисани и помоћу њих је објашњен начин рада система. Показано је да модел успешно обавља предвиђене задатке и даје очекиване резултате за сваки режим.

6. ЛИТЕРАТУРА

[1] З. Ивановић, В.А. Катић, „*Обновљиви извори електричне енергије – вежбе*“, ФТН Издаваштво, Нови Сад, 2018.

[2] https://pdf.sciencedirectassets.com „*Study on the optimal charging method for lithium-ion batteries*“

[3] https://www.researchgate.net/publication/228816125 „*Model Predictive Control Based Fast Charging for Vehicular Batteries*“

Кратка биографија:

Марко Миланко рођен је у Книну 1995. Факултет техничких наука уписује школске 2014/2015 године. На мастер студијама се определио за смер Дистрибуирани електроенергетски ресурси на којима је одбранио мастер тезу 2020 године.

Владимир Катић рођен је 1954. годинеу НовомСаду. Докторирао је на Универзитету у Београду 1991. године. Од 2002. године је редовни професор Универзитета у Новом Саду. Области интересовања су енергетска електроника, квалитет електричне енергије, обновљиви извори електричне енергије и електрична возила.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42.045
DOI: https://doi.org/10.24867/12BE22Bjekovic

GENERATOR WEB APLIKACIJA BAZIRANIH NA ANGULARJS I SPRING RAZVOJNIM OKVIRIMA

CODE GENERATOR FOR WEB APPLICATIONS BASED ON ANGULARJS AND SPRING FRAMEWORKS

Božo Bjeković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je prikazan generator koda zadužen za generisanje osnove aplikacije kako serverske, tako i klijentske strane. Serverska strana izgenerisana je u Java programskom jeziku uz oslonac na Spring radni okvir, dok je klijentska strana u AngularJS-u. Proces generisanja bazira se na definisanom UML dijagramu modela*

Ključne reči: *Generator koda, UML, šabloni*

Abstract – *The paper presents a code generator for generating the application base, for both server and client side. The server side is generated in Java programming language based on the Spring framework, while the client side is in AngularJS. The generation process is based on a defined UML model diagram.*

Keywords: *Code Generator, UML, patterns*

1. UVOD

Primena softverskog inženjerstva nad složenim sistemima je izrazito kompleksna zbog heterogenosti sistema i izazova koji proizilaze iz različitih domena razvoja softvera. Stoga, zahtevan softver se uglavnom razvija od strane domenskih eksperata sa drugačijim tačkama gledišta, razumevanjima sistema i njegovih funkcionalnosti. Primena jezika opštih namena često ne mogu da reše ovakav problem heterogenih sistema, što dovodi do tendencije za upotrebom jezika specifičnih za domen.

Savremeni poslovni sistemi koji opisuju složene probleme postaju teži za kreiranje i za sam proces održavanja. Klijenti koji traže kompikovane zahteve sa jedne strane i zahtevni softverski projekti sa druge, nekada podrazumevaju implementaciju velikog broja povezanih domenskih entiteta.

Ručno implementiranje ovakvih sistema zahteva dosta vremena, a pored toga javljaju se i ostali problemi. Svaki programer ima svoj stil pisanja koda, što prouzrokuje nekonzistentnost u samoj implementaciji rešenja, pojava grešaka je mnogo veća u odnosu na automatizaciju procesa i migracija sistema na nove platforme je gotovo nemoguća.

Iz ovoga proizilazi potreba za kreiranjem rešenja, koje će na osnovu definicije konkretnog modela, automatizovati postupak i olakšati dalji razvoj softvera.

U ovom radu je prikazano upravo takvo rešenje, koje omogućava generisanje osnove aplikacije i njenih bazičnih funkcionalnosti u Javi i AngularJS-u na osnovu opisa modela.

Motiv za razvoj ovog rešenja javlja se kao potreba da se ubrza i olakša inicijalno kreiranje osnove jednog funkcionalnog sistema. U današnje vreme, gde broj korisnika na internetu konstantno raste, razvijanje samo serverske strane koja će pored obrade podataka prikazivati i vizuelni sadržaj korisniku, predstavlja veliko opterećenje za server. Razlozi za ovo jesu razne animacije i sitne promene koje se tiču samog korisničkog izleda, a i dalje trebaju pomoć servera. Kao odgovor na ovo, došlo je do razvoja odvojenih serverskih i klijentskih strana, gde će se klijent strana baviti samo prikazivanjem korisničkog interfejsa. Pored toga, klijentska strana je u mogućnosti da obavlja i dodatnu obradu podataka, te samim tim omogućava prebacivanje određenih funkcionalnosti sa servera, čime se vrši rasterećivanje istog. Ovim je rešen problem nepotrebnog opterećivanja servera i omogućeno razdvajanje implementacionog od prezentacionog sloja.

Međutim, sa druge strane, potrebno je više vremena za razvoj, što čini glavni motiv za razvoj generatora koda, koji će omogućiti i olakšati programerima da na osnovu kreiranog modela sistema, dobiju kreiranu osnovu na kojoj će nastaviti da razvijaju sopstvenu poslovnu logiku, bez da troše dragoceno vreme kreirajući arhitekturu sistema za obe strane, klijentsku i serversku. Serverska strana je kreirana na principu slojne arhitekture (layer architecture) sačinjene od tri sloja, gde se prvi sloj bavi logikom baze podataka, osnovnim operacijama i samom konekcijom prema bazi. Drugi sloj je zadužen za poslovnu logiku, dok je treći sloj zapravo sloj veb kontrolera. Klijentska strana je definisana na sličan način, s tim što postoje samo kontroleri, koji se bave protokom podataka i servisi koji komuniciraju sa samim serverom.

2. DEFINICIJA POJMOVA

2.1. Model

Model je apstraktni prikaz strukture, funkcije ili ponašanja sistema [1]. To je uprošćena predstava kompleksne realnosti. Svi odgovori koji se izvedu iz modela, moraju da važe i u realnom sistemu. Danas, sistemi i softver postaju veoma kompleksni i ne mogu se razumeti bez odgovarajućeg modelovanja. Njegov cilj je da uobliči na vidljiv, često formalan način ono što je suštinsko za razumevanje nekog aspekta sadržaja, strukture ili ponašanja. Nivo apstrakcije u procesu

---

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević, red. prof.

modelovanja utiče na validnost modela, odnosno na uspešnost predstavljanja realnog sistema. Neki od dodatnih ciljeva modelovanja su [2]:

- Pomaže u vizuelizaciji sistema onakvog kakav jeste ili onakvog kakav želimo da bude
- Omogućava specifikaciju strukture i ponašanja sistema
- Dokumentuje odluke koje su donesene
- Obezbeđuje zajednički jezik za stejkholdere
- Omogućava jasnoću i razumevanje

2.2. Meta-model

Meta-model opisuje koncepte i moguću strukturu modela definišući koncepte domena i njihove veze kao i ograničenja i pravila modelovanja [1]. Proces kreiranja meta-modela naziva se metamodelovanje. Metamodelovanje se u nekim sferama ljudske aktivnosti naziva još i semantičko modelovanje, kreiranje šeme ili standardizacija domena [3]. Jezik kojim definišemo meta-modele naziva se meta-metamodel i koristi se za definisanje meta-modela, a i samog sebe, pa ih iz tih razloga se naziva samodefinišućim.

2.3. Platforma i transformacija

Platforma predstavlja skup podsistema i tehnologija koji pružaju koherentan skup usluga putem interfejsa i propisanih načina upotrebe, koje sve aplikacije podržane tom platformom mogu koristiti, bez potrebe da imaju saznanje o načinu implementacije pruženih funkcionalnosti [4].

Transformacija je process u kome se vrši pretvaranje jednom modela u drugi. Uvek su zasnovane na izvornom meta-modelu, jer je izvorni model tačno jedan primerak konkretnog meta-modela [1]. Pravila transformacije mogu se zasnivati samo na konstrukcijama meta-modela i ona nedvosmisleno definišu izlaz.

2.4. Generator koda

Generatori koda su alati koji na automatizovan način transformišu modele u kod za interpretaciju u određenoj sintaksi i time pružaju veći kvalitet softverskog rešenja i produktivnost u radu. Ovaj proces zavisi i rukovodi se meta-modelom, jezikom za modelovanje sa njegovim konceptima, semantikom i pravilima, te ulaznom sintaksom potrebnom za ciljano okruženje [5]. Generatori koda se mogu različito klasifikovati, međutim kao najčešća podela postoje deklarativne i operativne. Ova klasifikacija se zasniva na pristupu koji se koristi za određivanje generatora. U deklarativnom pristupu opisano je mapiranje između elemenata izvornog (meta-model) i ciljanog programskog jezika. Operativni pristupi, kao što su pravila transformacije grafova, definišu korake potrebne za proizvodnju ciljanog koda i datog izvornog modela.

Efikasnost generisanog koda se uglavnom ogleda u tome koliko iskusan programer ga je napisao. Glavni argument protiv generatora koda jeste tvrdnja da generisani kod ne može da ispuni stroge zahteve i efikasnost izvođenja koji su osnovni problemi pri razvoju softvera za uređaje sa ograničenom memorijom i resursima za obradu [5]. Kada kod generiše generator koji je napravio iskusni programer, uvek će proizvoditi bolji kod nego što prosečni programer piše ručno. Upravljanje memorijom, optimizacija, model programiranja i stilovi se dosledno primenjuju. U idealnom slučaju, generisani kod bi trebao izgledati kao kod koji je ručno napisao iskusni programer koji je definisao generator.

2.5. Jezici specifični za domen

Jezici opšte namene su mali jezici, fokusirani na određeni aspekt softverskog sistema, a ne za određenu oblast [9]. Jako teško je postići kreiranje kompletnog programa koristeći samo DSL (Domain Specific Language), međutim upotrebom više DSL-ova, uglavnom napisanih na jeziku opšte namene, možemo postići veću pokrivenost generisanog koda.

Dobro dizajnirani i kreirani jezici specifični za domen su od velike pomoći programerima, i mnogo je lakše programirati na takav način nego sa tradicionalnim bibliotekama. Ovo poboljšava produktivnost programera, što je uvek poželjno, i može poboljšati komunikaciju sa domenskim ekspertima što je takođe važan faktor u razvoju softverskog rešenja.

3. IMPLEMENTACIJA RJEŠENJA

Generator koda je napisan u Java programskom jeziku, za kreiranje meta-modela i konkretnog modela rešenja je zadužen MagicDraw alat i kao obrađivač šablona (template engine) je upotrebljen FreeMarker.

3.1. Tehnologije

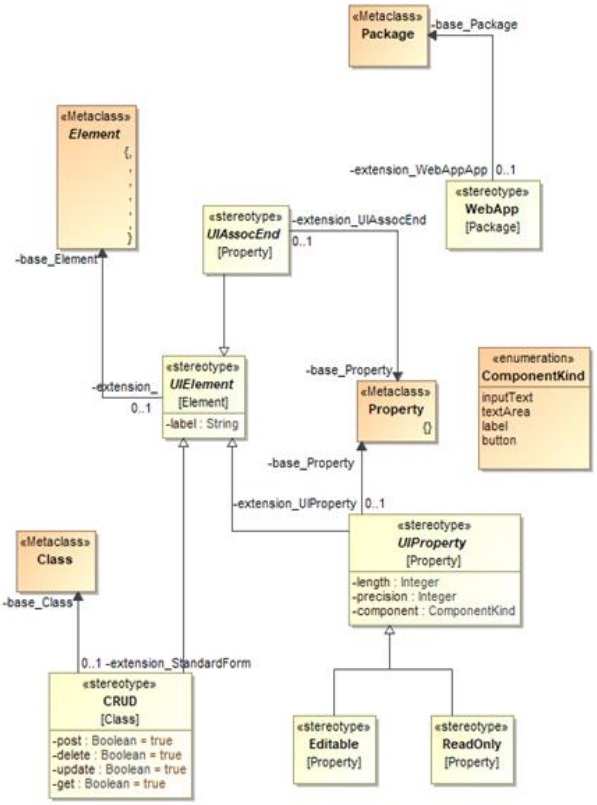
Razvojno okruženje korišćeno za razvoj i implementaciju softverskog rešenja jeste Eclipse IDE. Generator koda je razvijen i upotrebljen kao plugin unutar MagicDraw alata koji ga ujedno i snabdeva ulaznim podacima, u ovom slučaju kreiranim UML dijagramom. Kao rezultat generatora, dobija se odvojena server (back-end) i klijent (front-end) aplikacija. Server aplikacija je izgenerisana u programskom jeziku Java, oslanjajući se na Spring framework, dok je klijentska aplikacija kreirana uz pomoć AngularJS jezika.

3.2. Meta-model

Meta-model, prikazan na slici 1, predstavlja osnovu na kojoj je kreiran model, čija struktura će biti objašnjena detaljnije u narednom poglavlju. Osnovni model proširen je stereotipima (stereotype) koji definišu kako se postojeća meta-klasa može nadograditi. Stereotipi se mogu kombinovati i mogu se primeniti na bilo koji elemenat modela (aktivnostima, atributima, zavisnostima i drugim). WebApp stereotip koji nasleđuje Package element predstavlja glavni paket koji grupiše sve klase i druge elemente modela zajedno. Svaki dijagram mora da se nađe unutar jednog paketa. UIElement stereotip, koji nasleđuje Element klasu meta-modela, proširuje element sa novim poljem label i predstavlja komponentu korisničkog interfejsa sa svojim nazivom. CRUD (Create Read Update Delete) stereotip nasleđuje Class elemenat meta-modela i predstavlja opis mogućih operacija nad primenjenim entitetom. Polja koja se nalaze unutar njega su: post (mogućnost kreiranja entiteta), delete (mogućnost

brisanja entiteta), update (mogućnost izmene entiteta) i get (mogućnost čitanja entiteta). Sva polja unutar pomenutog stereotipa su opciona i prilikom generisanja koda, na osnovu ovih polja određuje se koji tipovi metoda će biti kreirani. UIProperty stereotip nasleđuje Property element meta-modela i definiše opis polja nad kojim se primenjuje. Definiše dužinu (length polje), preciznost (precision polje) i tip polja (component polje). Tip polja je označen enumeracijom ComponentKind i definiše moguće tipove polja. U ovo slučaju to su tekstualno polje (inputText i textArea), oznaka (label) i dugme (button). UIProperty stereotip je dodatno proširen sa dva stereotipa Editable i ReadOnly. Editable stereotip označava da li neko polje ima mogućnost izmene, dok sa druge strane ReadOnly označava zabranu izmene.

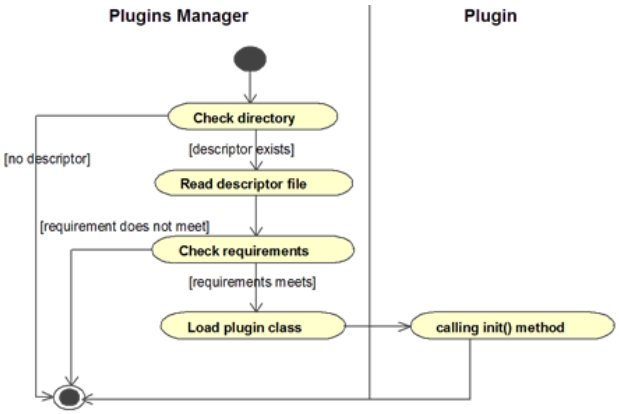
Opisani meta-model se dalje koristi prilikom kreiranja konkretnog modela veb aplikacije i na osnovu njega se proverava ispravnost. MagicDraw alat vrši proveru ispravnosti kreiranog modela na osnovu zadatog meta-modela. Prikazani meta-model se može koristiti za više tipova jednostavnih veb aplikacija koje podržavaju CRUD operacije nad modelima. Jedan od takvih primera aplikacije jeste veb forum, gde će se na osnovu kreiranog modela generisati projekat sa celokupnom podrškom za CRUD operacije, sto će biti prikazano i objašnjeno u narednom poglavlju.



Slika 1. Meta-model jezika za opis definicije podataka

3.3. Implementacija generatora

Kreiranje i dodavanje plugin-a su način na koji se može proširiti i promeniti funkcionalnost samog MagicDraw alata, što jeste i glavna svrha plugin arhitekture. Na slici 2 prikazan je način provere i dodavanja novog plugina u MagicDraw alat. Prilikom pokretanja alata, pretražuje se plugin direktorijum i pokušava se pronaći plugin descriptor kojeg čita plugin menadžer. Ukoliko je sve ispravno navedeno plugn manadžer poziva init() metodu koja je obavezna u kreiranom plugin-u.



Slika 2. Način dodavanja plugin-a

Unutar pomenute metode, vrši se dodavanje novog podmenija unutar glavnog menija MagicDraw-a i registrovanje svih neophodnih generatora modula koji će biti korišćeni. Generatori modula se registruju uz odgovarajuće parametre koji se prosleđuju u vidu ključeva na osnovu čega se dobijaju konkretne vrednosti iz XML zapisa. Svaki generator modula je zadužen za generisanje odgovarajućeg dela aplikacije i postoje sledeći:

- Generator modela
- Generator enumeracija
- Generator sloja za pristup podacima (Repository layer)
- Generator poslovne logike (Service layer)
- Generator kontrolera (Controller layer)
- Generator AngularJS kontrolera
- Generator AngularJS servisa
- Generator HTML stranica
- Generator AngularJS rutiranja
- Generator početne stranice

Nakon kreiranog modela, u ovom slučaju veb foruma, i nakon njegove provere ispravnosti unutar MagicDraw alata, sledi proces generisanja izvršnog koda. Proces se sastoji od rednog pozivanja generatora modula koji su zaduženi sa odgovarajuće delove aplikacije. Klikom na generate dugme, unutar dodatog pomenija strartuje se process generisanja. Generator koda, kreiran kao plugin, preuzima model iz MagicDraw alata i parsira ga. U toku parsiranja pomenutog modela, izdvajaju se i prepoznaju odgovarajući elementi, poput klasa, enumeracija, paketa, atributa i drugih.

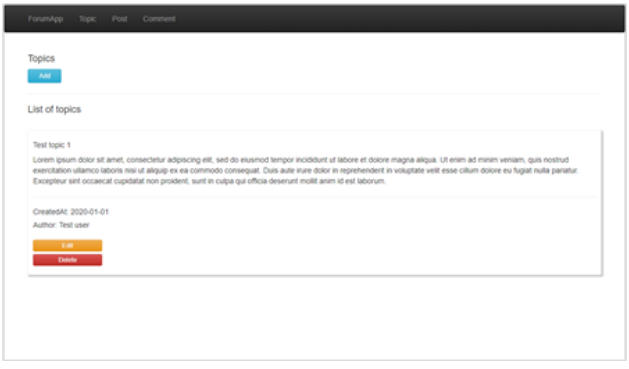
Prolaskom kroz sve elemente modela proverava se tip i na osnovu njega kreira se odgovarajuća instanca klase. Postoje tri provere tipa elemenata za proveru i to su klasa, enumeracija i paket. Ukoliko se radi o klasi, prvo se proverava da li pripada odgovarajućem paketu. Ako je ovaj uslov ispunjen, sledi proces kreiranja klase sa svim potrebnim zavisnostima, poljima i metodama.

Na listingu 1 prikazan je deo šablona koji se koristi u procesu generisanja entiteta i odnosi se na generisanje polja. U odnosu na tip veze, generiše se odgovarajući tip polja u izvornom kodu.

```
<#list properties as property>
    <#if property.upper == 1 >
    public ${property.type} get${property.name?cap_first}
    (){
        return ${property.name};
    }

    public void
        set${property.name?cap_first}(${property.type}
        ${property.name}){
           this.${property.name} = ${property.name};
    }

    <#elseif property.upper == -1 >
    public Set<${property.type}>
        get${property.name?cap_first}(){
        return ${property.name};
    }

    public void set${property.name?cap_first}(
        Set<${property.type}> ${property.name}){
        this.${property.name} = ${property.name};
    }

    <#else>
        <#list 1..property.upper as i>
    public ${property.type}
        get${property.name?cap_first}${i}(){
        return ${property.name}${i};
    }

    public void
        set${property.name?cap_first}${i}(${property.type}
        ${property.name}${i}){
        this.${property.name}${i} = ${property.name}${i};
    }

        </#list>
    </#if>
</#list>
```

Listing 1. Primer FreeMarker šablona

Prilikom pokretanja aplikacije uočava se traka za navigaciju koja prikazuje navedeno ime aplikacije, u ovom slučaju veb foruma. Pored toga, navedeni su svi entiteti, kao linkovi, kreirani unutar UML dijagrama modela pomoću kojih se vrši navigacija unutar aplikacije. Na prvom ekranu prikaza je strana teme (Topic), sa naslovom i dugmetom za kreiranje nove. Pored toga, ostatak ekrana namenjen je za prikaz kreiranih tema, njihovu izmenu i brisanje. Klikom na dugme "Add" otvara se modalni dijalog sa formom za kreiranje nove teme.



Slika 3. Prikaz dodog entiteta

4. ZAKLJUČAK

Zadatak ovog rada bio je da prikaže način kreiranja osnove funkcionalnog sistema i njegove arhitekture na osnovu definisanog modela. S obzirom da se u današnje vreme razvoj veb aplikacija svodi na kreiranje dva odvojena dela, serverski koji se bavi obradom podataka i klijentski koji se bavi prikazom, programeri trebaju više vremena kako bi kreirali osnove i postavili projekat za obe strane. Rešenjem, tj. generatorom koda prikazanim u ovom radu, omogućava se programerima da na osnovu kreiranog modela sistema dobiju postavljenu osnovu za obe strane i fokusiraju se na kreiranje konkretne logike aplikacije.

Pod ovim se podrazumeva da serverska strana poseduje kreirane entitete, slojnu arhitekturu koja je podeljena na deo upravljanja bazom podataka, deo poslovne logike sistema i deo kontrolera, odnosno sam veb API.

Pored ovoga, napravljene su i sve navedene CRUD operacije naznačene u procesu modelovanja, upotrebom MagicDraw alata. Klijentska strana, na osnovu kreiranog modela, sadrži generisane HTML stranice za svaki entitet, kontrolere zadužene za kontrolisanje podataka i servise spremne za komunikaciju sa serverskom stranom. HTML stranice poseduju forme koje služe za kreiranje i izmenu entiteta, kao i prikaz svih postojećih entiteta. Servisi imaju odgovarajuće metode, takođe generisane na osnovu naznačenih CRUD operacija unutar modela.

Postoji mnogo prostora za dalje razvijanje i unapređenje ovog generatora. Neki od pravaca mogu biti proširenje rešenja da podržava generisanje u drugim programskim jezicima i platformama, što se odnosi na obe strane. Poboljšanje meta-modela u cilju podrške dodatnih koncepata, jeste isto jedan od pravaca. Takođe, postoji mogućnost razvoja dodatnog generatora koji bi se, pored komunikacije sa osnovnim, striktno bavio generisanjem korisničkog interfejsa i na taj način bi se dobila celina u kojoj bi se programeri još više usmereni na konkretno pisanje poslovne logike.

5. LITERATURA

[1] M. Voelter, T. Stahl, Model-Driven Software Development: Technology, Engineering, Management, John Wiley & Sons, 2006, Foreword.

[2] D. Z. Jeremic, "Uvod u modelovanje korišćenjem UML-a, https://www.vps.ns.ac.rs/Materijal/mat22111.pdf.

[3] I. Dejanović, Prilog metodama brzog razvoja softvera na bazi proširivih jezičkih specifikacija, 2011.

[4] G. Milosavljević, Prezentacije sa predavanja predmeta Metodologije brzog razvoja softvera.

[5] S. Kelly and J.-P. Tolvanen, Domain-Specific Modeling, Wiley-IEEE Computer Society Pr, 2008.

Kratka biografija:

Božo Bjeković rođen je 31.05.1994. godine u Trebinju. Osnovnu školu „Sveti Sava", u Gacku, završio je 2009. godine. Gimnaziju „Pero Slijepčević" u Gacku završio je 2013. godine. Iste godine upisao je Fakultet tehničkih nauka u Novom Sadu, smer Softversko inženjerstvo i informacione tehnologije. Osnovne akademske studije završio je u oktobru 2017. godine. Iste godine je upisao master akademske studije na istoimenom smeru.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 620.93
DOI: https://doi.org/10.24867/12BE23Perge

ISPITIVANJE UTICAJA OBNOVLJIVIH IZVORA UTICAJA PRIMENOM TEST MREŽE

ANALYSIS OF THE INFLUENCE OF RENEWABLE SOURCES USING TEST GRIDS

Pavle Perge, Vladimir Katić, Aleksandar Stanisavljević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu razmatra se uticaj obnovljivih izvora električne energije na IEEE 13-bus test mrežu. Mreža je modifikovana tako što je u nju priključen distribuirani generator – vetroelektrana. Cilj ovog rada je da se utvrdi uticaj priključenog distribuiranog generatora na napone kratkog spoja i samu mrežu. Modelovanje je rađeno u programskom paketu DIgSILENT Power Factory.*

Ključne reči: *kratak spoj, IEEE test mreža, vetroelektrana*

Abstract – *This paper discusses the impact of renewable energy sources on the IEEE 13-bus test network. The network has been modified by connecting a distributed generator - a wind power plant. The aim of this paper is to determine the influence of the connected distributed generator on short-circuit voltages and the network itself. The modeling was done in the DIgSILENT Power Factory software package.*

Keywords: *short circuit, IEEE test grid, wind farm*

1. UVOD

Pod distribuiranim generatorima (DG) podrazumevaju se generatori manjih snaga, koji se priključuju direktno na distributivnu mrežu i koji za pogon koriste obnovljive izvore energije ili viškove energije iz nekog drugog tehnološkog (proizvodnog) procesa. Samo ime DG vezuje se za izvore energije, koji su rasprostranjeni po distributivnoj mreži. Iz ovoga sledi da se distribuirana proizvodnja odnosi na električnu energiju dobijenu na lokaciji potrošača ili u njegovoj blizini [1].

Priključenje ovih izvora i njihova sinhronizacija sa mrežom obavlja se preko odgovarajućih energetskih elektronskih pretvarača, koji su nelinearne prirode i zasnovani na digitalnim tehnikama upravljanja. Iz tog razloga značajno je razmotriti njihovu interakciju sa mrežom, odnosno obostrani uticaj na parametre kvaliteta električne energije.

Uticaj može biti dvojak. Sa jedne strane DG-i poboljšavaju naponske prilike u distributivnoj mreži, smanjuju gubitke u prenosu, poboljšavaju pouzdanost napajanja, ali i doprinose povećanju harmonijske distorzije napona. S druge strane, poremećaji naponskih prilika u mreži (propadi ili poskoci napona, nesimetrija, fliker i sl.) utiču na pravilan i stabilan rad energetskih pretvarača sa strane mreže, odnosno mrežnih invertora.

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

Distribuirani generatori mogu se podeliti na više načina: prema vrsti energenta, snazi, vrsti generatora, načinu priključenja na mrežu, brzini upravljanja, nivou učešća (prisustva u mreži).

Podela DG prema vrsti energenta:

- vetroelektrana
- solarna (FN) elektrana
- hidroelektrana
- biogasna elektrana
- geotermalna elektrana
- mikroturbina (gasna)
- dizel agregat
- elektroenergetska skladišta

Prvih pet kategorija spadaju u obnovljive izvore , a šesta i sedma u fosilna goriva.

U ovom radu će se posmatrati propadi napona na sabirnicama mrežnih invertora, kao posledica kvarova na mreži. Radi referentnijih rezultata, istraživanje će se vršiti na standardizovanoj, IEEE test mreži sa dodatim DG-om (vetroelektranom) i to IEEE 13-bus [2]. Koristiće se posebno razvijeni model test mreže u demo-verziji programskog paketa DIgSILENT Power Factory [3]. Cilj rada je da utvrdi uticaj priključenog DG, kao i vrsta kvara na fazne napone.

2. TEST MREŽE

Pokazalo se da je testiranje različitih scenarija u softverskim alatima vremenski dugotrajno i zahtevno. Zbog toga, pojavila se potreba da se ustanove jedan ili više referentnih modela distributivnih mreža sa standardizovanom kompleksnošću, strukturom i dobro poznatim i dokumentovanim parametrima, koji mogu da predstavljaju distributivne ili prenosne mreže. Navedeni test i referentni modeli su izabrani kao optimalni da mogu da pokriju različite potrebe za testiranjem u različitim uslovima i u zavisnosti od tipova opterećenja i same strukture elektroenergetskog sistema. Uglavnom ovi modeli predstavljaju uprošćene modele realnih elektroenergetskih sistema. Od velikog broja test mreža ovde će se koristiti IEEE test mreža (IEEE - *Institute of Electrical and Electronics Engineers*) i to ona sa 13 (*IEEE 13-bus*) [2]. Da bi se razmatrao uticaj DG-a, ovoj test mreži će se dodati vetrogenerator na jednoj sabirnici IEEE 13-bus.

Postoji dve vrste test mreža : prenosne i distributivne.

Distribuirana proizvodnja (engl. Distributed generation-DG) je termin koji se u elektroenergetici koristi za proizvodnju električne energije na lokaciji potrošača , dok se pod terminom distribuirani izvori-generatori misli na generatore koji se priključuju na distributivnu, NN ili SN mrežu. U IEEE standardu se distribuiranim generatorima smatraju jedinice snage do 10MW.

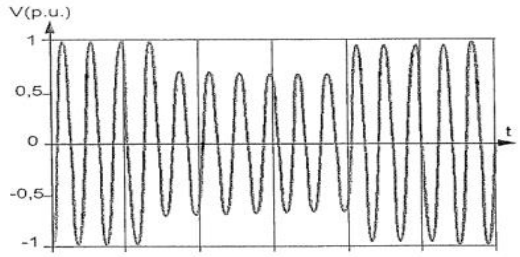
3. KVALITET ELEKTRIČNE ENERGIJE - PROPADI NAPONA I KRATKI SPOJEVI

Pitanje kvaliteta električne energije je usko vezano za osnovne postavke elektroenergetskog sistema, pa je od velike važnosti za njegovo funkcionisanje. Kvalitet električne energije predstavlja zajednički interes potrošača i proizvođača električne energije, a problemi u vezi sa kvalitetom su postavljeni u žižu interesovanja savremene distributivne mreže. Od posebnog interesa si viši harmonici i propadi napona, jer su njihovi efekti posebno izraženi u radu EES-a i potrošača [4]. U ovom radu posebna pažnja će biti na simulaciji kratkih spojeva .

Propad napona predstavlja kratkotrajno smanjenje efektivne vrednosti napona od 10% do 90% nominalne vrednosti, obično u trajanju od polovine periode do 1 min. Na slici 1 prikazan je tipičan vremenski oblik propada napona. Uzroci propada napona su kratki spojevi, uključenje velikih potrošača (asinhronih motora i elektrolučnih peći), povezvanje distributivnih transformatora i druga preopterećenja u mreži. Posledice propada napona su otkazivanje opreme, prekid programa, gubitak informacija, prekid proizvodnog procesa ili oštećenje industrijskih proizvoda. Kod DG-a propadi mogu da izazovu prevelike struje i pregorevanje tranzistorskih prekidača u invertoru, gubitak sinhronizacije i ispad DG-a.

Tehničke prednosti primene DG:

- poboljšavaju naponske prilike i kvalitet el. energije u distributivnoj mreži
- smanjuju se gubici u mreži i opterećenja dalekovoda u prenosu el.energije
- jednostavnije upravljanje potrošnjom i izravnavanje dijagrama opterećenja
- povećavaju pouzdanost napajanja
- umanjuju emisiju CO_2
- mogu davati podršku mreži tokom kvara (Fault Ride Trought – FRT, Low Voltage Ride Trough LVRT) i napajati mrežu u tim situacijama.



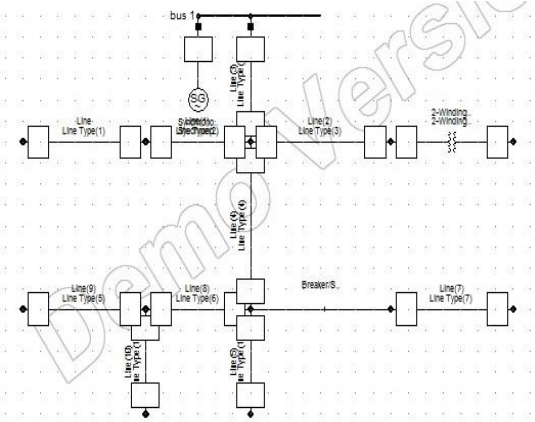
Slika 1. *Vremenski oblik propada napona [4]*

4. MODELOVANJE IEEE TEST MREŽE

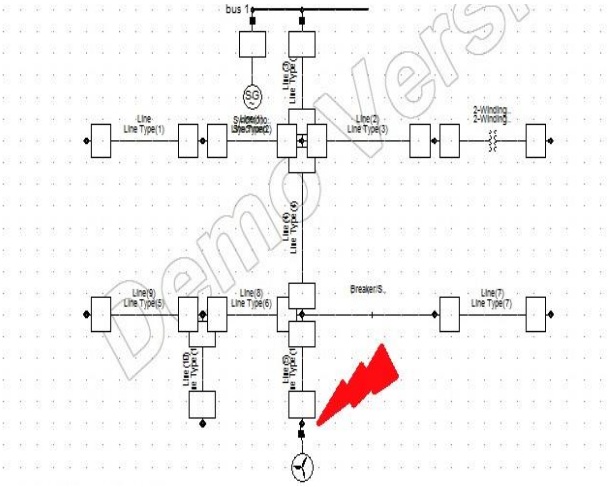
Za ovo istraživanje izvršeno je modelovanje IEEE test mreže sa 13 sabirnica dopunjenih sa DG-om (vetroelektranom) u programskom jeziku DIgSILENT Power Factory [3].

4.1 Model IEEE 13-bus test mreže

IEEE-13 test mreža je dizajnirana za testiranje algoritama u neuravnoteženim trofaznim radijalnim sistemima.Ova test mreža je uprošćen model realne distributivne mreže. Ono što karakteriše ovu mrežu je da je ona kratka, nebalansirana i u značajnoj meri opterećena mreža.Mreža se sastoji od 13 čvorova koji su međusobno povezani sa 10 vodova, ima jedan glavni izvor napajanja kao i dva transformatora (jedan distributivni transformator ΔY 115/4.16 KV I jedan linijski transformator YY 4.16/0.480 KV), dve kondenzatorske baterije, kao I neuravnotežene distributivne potrošače.



Sl. 2. *Model modifikovane IEEE 3-bus test mreže modelovane u DIgSILENT-u*



Sl. 3. *Model modifikovane IEEE 13-bus test mreže sa povezanim vetrogeneratorom i označenim mestom kvara (sabirnica 680)*

5. REZULTATI SIMULACIJA

Simulirani su jednopolni (*1pks*) i tropolni (*3pks*) kratki spojevi na sabirnici 680 kako bi se uočio uticaj vetrogeneratora na mesto kvara.

5.1. Rezultati simulacija na IEEE 13-bus test mreži

Posmatrani su 1pks i 3pks i njihov uticaj na nivo propada napona na mestu priključenja vetrogeneratora, tj. na sabirnici 680.

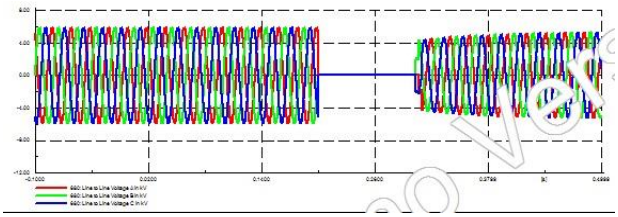
5.1.1. 3PKS na sabirnici 680

Na slikama 4 i 5 prikazani su rezultati simulacija, odnosno fazni naponi na sabirnici 680 u slučaju 3pks, sa i bez priključenog vetrogeneratora.

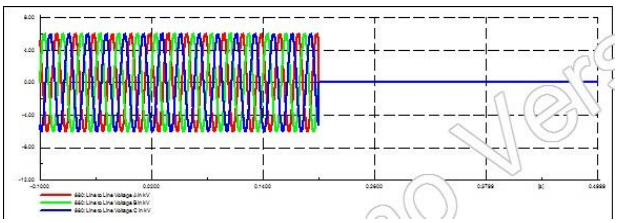
Mogu se uočiti odgovarajući poremećaji napona i uticaj vetrogeneratora na napone kratkog spoja.

Sa slike 4 vidi se da nakon prestanka kratkog spoja sistem pokušava da se vrati u normalu sa znatno manjim amplitudama i različitim periodama oscilovanja.

Sa slike 5 vidi se da sistem nakon kratkog spoja ne uspeva da se vrati u normalan rad i da su naponi u sve 3 faze nula (negativan uticaj DG).



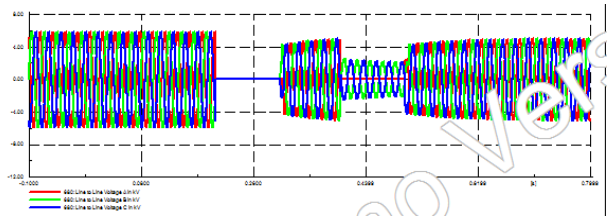
Sl.4. *Naponi kratkog spoja u slučaju 3pks bez povezanog vetrogeneratorom*



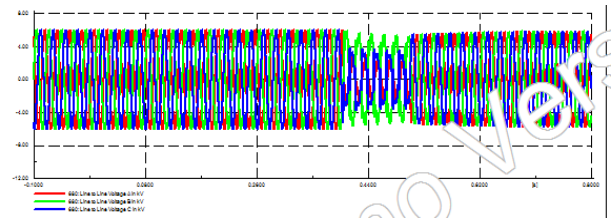
Sl.5. *Naponi kratkog spoja u slučaju 3pks sa povezanim vetrogeneratorom*

5.1.2. 1PKS na sabirnici 680 u fazi A

Na slikama 6 i 7 prikazani su rezultati simulacija, odnosno fazni naponi na sabirnici 680 u slučaju 1pks (u fazi A) sa i bez povezanog vetrogeneratora.



Sl.6. *Naponi kratkog spoja u slučaju 1pks bez povezanog vetrogeneratora u fazi A*



Sl.7. *Naponi kratkog spoja u slučaju 1pks sa povezanim vetrogeneratorom u fazi A*

Sa slike 6 se vidi da je napon faze A na nuli , dok naponi u ostale dve faze imaju propad i neku vrednost koja je različita od nule.

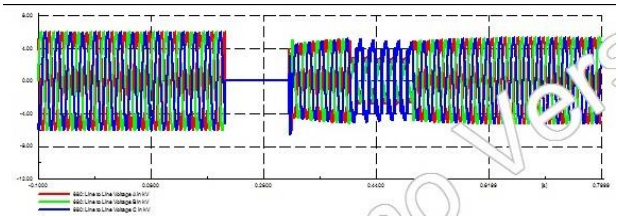
Sa slike 7 vidi se da sva tri fazna napona imaju neku vrednost čak i u trenutku kratkog spoja iako je napon u fazi A značajno manji od napona u preostale dve faze. Nakon 200 ms sistem se vraća u normalu, sa istom amplitudom kao i pre kratkog spoja.

5.1.3. 1PKS na sabirnici 680 u fazi B

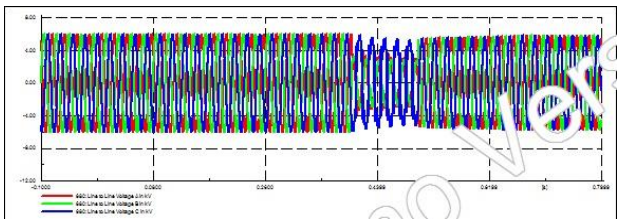
Na slikama 8 i 9 prikazani su rezultati simulacija, odnosno fazni naponi na sabirnici 680 u slučaju 1pks (u fazi B) sa i bez povezanog vetrogeneratora.

Sa slike 8 se vidi da je napon faze B na nuli, dok naponi u ostale dve faze imaju propad i neku vrednost koja je različita od nule.

Sa slike 9 se vidi da sva tri fazna napona imaju neku vrednost čak i u trenutku kratkog spoja iako je napon u fazi B značajno manji od napona u preostale dve faze. Nakon 200 ms sistem se vraća u normalu, sa istom aplitudom kao i pre kratkog spoja.



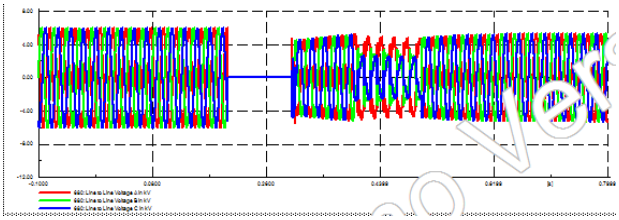
Sl.8. *Naponi kratkog spoja u slučaju 1pks bez povezanog vetrogeneratora u fazi B*



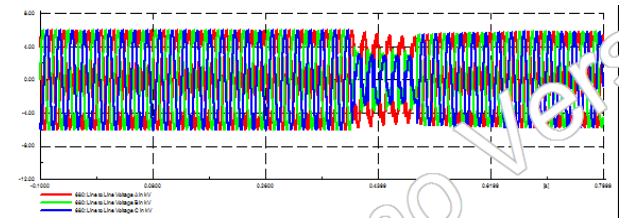
Sl.9. *Naponi kratkog spoja u slučaju 1pks sa povezanim vetrogeneratorom u fazi B*

5.1.4. 1PKS na sabirnici 680 u fazi C

Na slikama 10 i 11 prikazani su rezultati simulacija, odnosno fazni naponi na sabirnici 680 u slučaju 1pks (u fazi C) sa i bez povezanog vetrogeneratora.



Sl.10. *Naponi kratkog spoja u slučaju 1pks bez povezanog vetrogeneratora u fazi C*



Sl.11. *Naponi kratkog spoja u slučaju 1pks sa povezanim vetrogeneratorom u fazi C*

Sa slike 10 se vidi da su sve vrednosti faznih napona različite od nule, uz blagi propad napona, dok napon u fazi C ima najveći propad.

Sa slike 11 se vidi da sva tri fazna napona imaju neku vrednost čak i u trenutku kratkog spoja iako je napon u fazi C značajno manji od napona u preostale dve faze. Nakon 200 ms sistem se vraća u normalu, sa istom amplitudom kao i pre kratkog spoja.

5.2. Rezultati simulacija na IEEE 13-bus test mreži

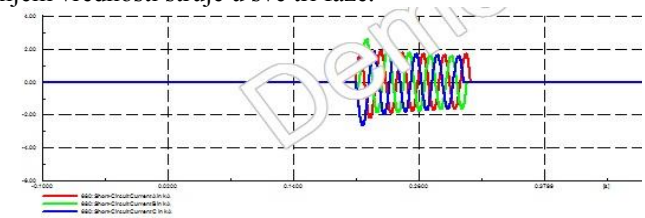
U ovom slučaju posmatrane su struje kratkih spojeva u svim fazama prilikom 3pks i 1pks na sabirnici 680 sa i bez povezanog vetrogeneratora

5.2.1. 3PKS na sabirnici 680

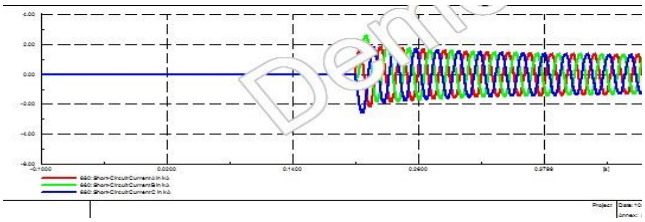
Na slikama 12 i 13 prikazani su rezultati simulacija tj. fazne struje u slučaju 3pks .

Sa slike 12 vidi se nastanak kratkog spoja i njegovo isčezavanje nakon 300ms. Nakon toga struja u svim fazama ima vrednost nula.

Sa slike 13 se vidi da struja kratkog spoja ne isčezava u periodu od 300ms, nego nastavlja sa postepenim smanjenjem vrednosti struje u sve tri faze.



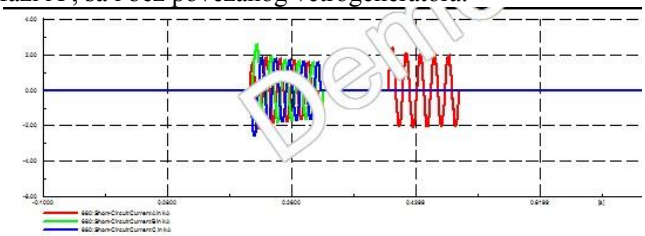
Sl.12. *Struja kratkog spoja za 3pks bez povezanog vetrogeneratora*



Sl.13. *Struja kratkog spoja za 3pks sa povezanim vetrogeneratorom*

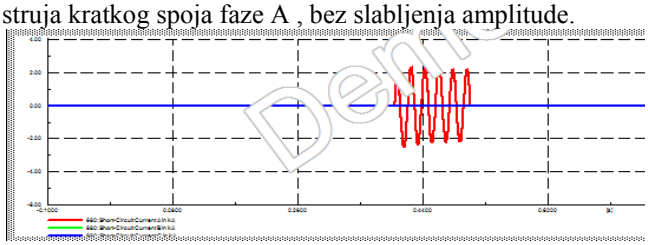
5.2.2. 1PKS na sabirnici 680 u fazi A

Na slikama 14 i 15 prikazane su simulacije struja 1pks u fazi A , sa i bez povezanog vetrogeneratora.



Sl.14. *Struje kratkog spoja u slučaju 1pks bez povezanog vetrogeneratora u fazi A*

Na slici 14 vidi se poremećaj u sve tri faze , i nakon toga stupa simulacija u trajanju od 200ms u kojoj se jasno vidi struja kratkog spoja faze A , bez slabljenja amplitude.



Sl.15. Struje *kratkog spoja u slučaju 1pks sa povezanim vetrogeneratorom u fazi A*

Na slici 15 se vidi struja kratkog spoja faze A bez prethodnog poremećaja .
Isto ponašanje sistema je i za fazu B i za fazu C.

6. ZAKLJUČAK

Analizom razlčitih slučajeva za jednopolni i tropolni kratak spoj sa i bez priključenog vetrogeneratora u mreži , zaključuje se da njegovo priključenje u mreži doprinosi povećanju vrednosti propada napona i faznim strujama, naročito kod tropolnog kratkog spoja. Ako je u mreži priključeno više DG-a dobija se veća snaga iz njih i povećanje deformacija napona i struja će biti veće .

U slučaju IEEE 13-bus test mreže rađene su simulacije na sabirnici 680 u slučaju 3pks i 1pks. Rezultati deformacije napona su očigledni pogotovo u slučaju sa povezanim DG-om.
Što se tiče struja kratkog spoja promene koje nastaju povezivanjem DG-a dovode do smanjenja amplitude struje kratkog spoja i njenog neiščezavanja.
Pored mesta kvara i pogonskog stanja sistema na propade napona u EES utiče i vrsta kvara.

7. LITERATURA

[1] V. Mijailović, *„Distribuirani izvori energije – principi rada i eksploatacioni aspekti“*, Akademska misao, Beograd, 2011.

[2] A. Stanisavljević, V.A. Katić, B. Dumnić, B. Popadić, „A Brief Overview of the Distribution Test Grids with a Distributed Generation Inclusion Case Study“, *Serbian Journal of Electrical Engineering*, Vol.15, No.1, Feb.2018, pp. 115 – 129,

[3] https://www.digsilent.de/en/

[4] V.A. Katić, A. Tokić, T. Konjić: *„Kvalitet električne energije“*, TEMPUS-CEFES, Fakultet tehničkih nauka, Novi Sad, 2007.

Kratka biografija:

Pavle Perge rođen je u Sremskoj Mitrovici 1991. god. Osnovne studije završio je na Fakultetu tehničkih nauka 2019., a master 2020.Iz oblasti Elektrotehnike i računarstva.

Vladimir A. Katić rođen je 1954. god. u Novom Sadu. Doktorirao je na Univerzitetu u Beogradu 1991. god. Od 2002. god. je redovni profesor Univerziteta u Novom Sadu. Oblasti interesovanja su mu energetska elektronika, kvalitet električne energije, obnovljivi izvori električne energije i električna vozila.

Aleksandar M. Stanisavljević, rođen je u Beogradu 1988. god. Doktorirao ne na Univerzitetu u Novom Sadu 2019. god. gde je trenutno u zvanju docenta. Oblast interesovanja su mu integracija obnovljivih izvora energije na mrežu i kvalitet električne energije.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE24Markovic

INTEGRACIJA NEWWAVE WORKFLOW ENGINE-A I OPENPONK ALATA

NEWWAVE WORKFLOW ENGINE AND OPENPONK TOOL INTEGRATION

Milica Marković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je predstavljena integracija NewWave workflow engine i OpenPonk alata. Model je nacrtan u OpenPonk editoru za meta-modelovanje. Plugin je zadužen za parsiranje modela u jeziku specifičnom za domen i mapiranje na NewWave elemente.*

Ključne reči: *Workflow engine, Pharo, NewWave, OpenPonk*

Abstract – *This paper presents integration of NewWave workflow engine and OpenPonk tool. The model is provided using OpenPonk tool for metamodeling. Plugin is in charge for reading domain specific language of model and for mapping in NewWave elements.*

Keywords: *Workflow engine, Pharo, NewWave, OpenPonk*

1. UVOD

Razvojem tehnologije, pre same izrade poslovnog procesa, neophodno je modelovanje elemenata poslovnog procesa. Modelovanje poslovnih procesa - *Business Process Modeling,* (BPM) je aktivnost u kojoj se predstavljaju (specificiraju) poslovni procesi nekog preduzeća na način da se mogu analizirati, poboljšavati i automatizovati. Odnose se na postojeće ili buduće (poboljšane) poslovne procese. Modeli tipično definišu:

- ko su korisnici (spoljni akteri),
- šta su ulazi i izlazi,
- koje su aktivnosti,
- način odvijanja poslova (*workflow* - tok izvršavanja),
- ko ih obavlja (unutrašnji akteri) [1].

Kao proizvod ove ideje, nastala je specifikacija takozvanog Workflow Engine. Standardizacijom i implementacijom Workflow engine-a, dobija se prednost u mogućnosti jednostavnog modelovanja aktivnosti poslovnih procesa. Workflow engine je softverska aplikacija koja upravlja poslovnim procesima [2]. Takođe, upravlja i nadgleda stanje aktivnosti u poslovnom toku. Akcije u samom toku mogu biti bilo šta, od čuvanja forme za apliciranje u *Document management system* do slanja kružnog mejla korisnicima. Workflow engine-i olakšavaju upravljanje tokom informacija, zadacima i događajima.
Pharo, objektno-orijentisan programski jezik, poseduje veliki potencijal za implementaciju i podršku mnogim poslovno-orijentisanim aplikacijama [3].

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević.

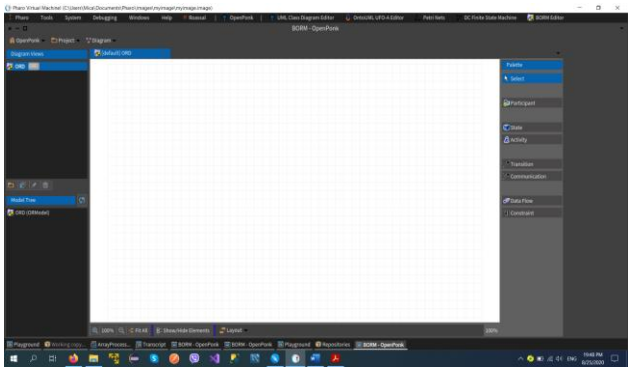
Pharo obezbeđuje jednostavnu, ali moćnu sintaksu i koncentrisan je na jednostavnosti i trenutnoj povratnoj informaciji.
Mnogi projekti zasnovani na Pharo programskom jeziku podržavaju pristup zasnovan na modelima (*Model-Driven approach)* i dinamički pristup svim fazama razvoja softvera: specifikaciji, dizajnu i implementaciji. U nedostatku široko prihvaćene podrške za modelovanje i implementaciju toka rada, nastao je NewWave workflow engine, sa idejom da uvede podršku za workflow rad u Pharo-u i omogući jendostavno upravljanje procesima [4].
OpenPonk je besplatna, otvorenog koda platforma za konceptualno modelovanje dijagrama, jezika specifičnih za domen i algoritama operacija nad modelima i dijagramima, kao što su transformacije i validacije modela, automatski raspored elemenata modela [5]. OpenPonk je implementiran u Pharo okruženju i razvijen pod MIT licencom. Razvijeno je okruženje koje omogućava grafičku vizualizaciju, interakciju grafičkih objekata, perzistenciju, raspored i grafički korisnički interfejs. OpenPonk platforma nudi dizajnerima modela alat koji rešava modelovanje notacije, specifične algoritme, transformacije modela, simulacije, itd. [6].
NewWave je workflow engine otvorenog koda napisan u Pharo-u. NewWave omogućava korisnicima da specificiraju kompoziciju workflow-a. To je omogućeno kroz procese i aktivnosti kao različiti tipovi zadataka i redosled tih aktivnosti. Svaki tok je vizualizovan korišćenjem Roassal biblioteke. NewWave primenjuje TaskIt biblioteku koja omogućava korišćenje zadataka u Pharo, sa apstrakcijom da izvršava i sinhronizuje konkurentne zadatke [7].
Cilj ovog rada je da pokaže kako na osnovu specifikacije meta-modela u alatu OpenPonk koji korisnik sam napiše, pozvati NewWave workflow engine da izvrši preces nacrtan u editor.

2. SPECIFIKACIJA SISTEMA

Motivacija za izradu ovog rada je da se na osnovu modela procesa specificiranog u alatu OpenPonk izvrši NewWave proces sa svojim elementima BPMN notacije.

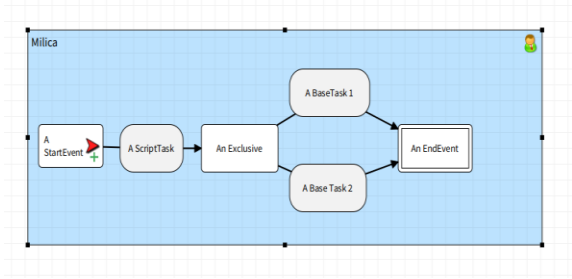
Iako korisnik kreira modele primarno kroz interfejs za dijagrame, postoje još načina za kreiranje modela: importovanje iz drugih alata (preko formata za razmenu, kao što je XMI), transformacija iz drugačije prezentacije (re-inženjerstvo programskog koda), ručno kreiranje potrebnih klasa (preko prorgamabilnog API-a), korišćenjem jezika specifičnih za domen, itd. Izbor odgovarajućeg načina zavisan je od konteksta, stoga će mnogi alati ponuditi različite opcije korisnicima. Kako bi ovo bilo omogućeno korisnicima, u OpenPonk je uveden

jezik specifičan za domen za meta-modele – BORM. Na slici 1 prikazan je BORM editor.



Slika 1. BORM okruženje

BORM model ima vizuelnu notaciju, a moguće je i koristiti tekst za opis modela procesa. Za potrebe izmena modela preko jezika specifičnih za domen, u OpenPonk je uveden DSL editor. Ovaj editor je direktno povezan sa trenutno otvorenim Workbench editorom i korisnik može vršiti izmene u modelu kroz ovaj editor; kada se tekst sačuva u editoru, model se promeni i kada se editor osveži na osnovu promena u modelu, promeni se i tekstualna prezentacija modela. Na slici 2 prikazan je primer modela procesa nacrtan u BORM OpenPonk Activity radnoj površini.



Slika 2. Model NewWave procesa

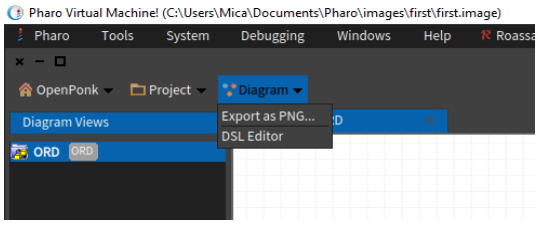
U primeru iz ovog poglavlja, proces započinje StartEvent događajem, koji je u BORM modelu označen sa *initial state* (znak plus sa slike 2). Nakon StartEvent-a sledi jedan ScriptTask u kome je definisana skripta koju on izvršava, i u ovom konkretnom primeru to je „10 atRandom“. Nakon ovog zadatka, sledi An Exclusive gateway-grananje, nakon koga slede dva BaseTask-a, BaseTask 1 i BaseTask 2. U Exclusive gateway, definisani su uslovi – *condition*, koje zadatak pre grananja mora da ispuni kako bi se nastavio proces u jednom od dva BaseTask-a. U ovom konkretnom primeru je da je za izvršavanje prvog BaseTask neophodno da broj generisan u ScriptTask-u bude veći ili jednak 5, a uslov za BaseTask 2 je da generisani broj bude manji od 5. Nakon ova dva BaseTask-a, sledi An EndEvent kojim se završava proces. EndEvent ima oznaku *final*, po čemu se kao i StartEvent-*initial*, razlikuje od ostalih zadataka.

3. IMPLEMENTACIJA REŠENJA

Kada je proces kreiran, potrebno je otvoriti *Domain Specific Language* (DSL) editor procesa u meniju OpenPonk-a, kao što je prikazano na slici 3).

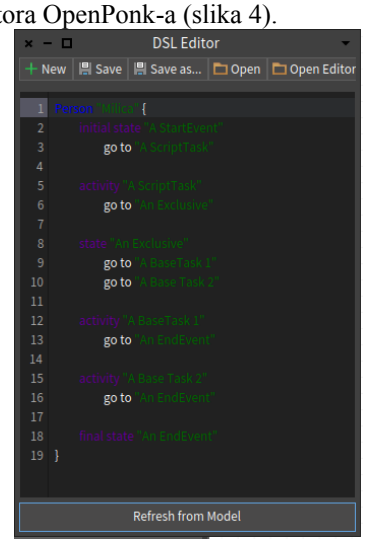
3.1 Kreiranje modela

Kada su nacrtani elementi procesa u Workbench editoru OpenPonk-a, potrebno je otvoriti DSL editor kako bi se napravio tekstualni model procesa, koji je potreban za dalji rad.



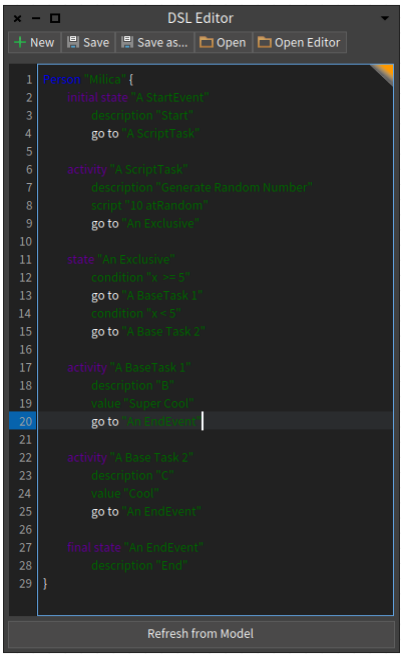
Slika 3. DSL editor

Kada se DSL editor otvori, neophodno je da se klikne na dugme *Refresh from model* na dnu prozora editora da bi se učitao model napisan u jeziku specifičnom za domen BORM editora OpenPonk-a (slika 4).



Slika 4. DSL editor sa modelom procesa

Pošto u modelu definisanom na ovaj način nema dovoljno podataka specifičnih za elemente procesa, potrebno je dodatno definisati atribute koji su usko vezani za elemente procesa. Da bi se engine pokrenuo, za minimalno izvršavanje potrebno je kreirati *Start* i *End Node* i sekvencu koja ih povezuje. Na slici 5 prikazan je tekstualni model procesa sa dodatnim obeležjima.



Slika 5. Model NewWave procesa

3.2 Obrada modela procesa

Nakon što je fajl sačuvan, potrebno ga je učitati, prepoznati elemente procesa i napraviti tok izvršavanja NewWave procesa. Za učitavanje fajla zadužena je klasa *LoadingFile* i njena metoda *loadDslFile*, koja za argument prima string putanju do mesta gde se nalazi fajl, prikazani na listingu 1.

```
loadDslFile: path

        | anArray |
        anArray   :=   path   asFileReference
readStreamDo: [ :in |
                Array streamContents: [ :out |
                [ in atEnd ] whileFalse: [ out
nextPut: in nextLine ] ] ].
        ^ anArray.
```

Listing 1. Metoda za učitavanje fajla

Zatim, potrebno je razgraničiti model po elementima koji se u njemu nalaze. Da bi se to uradilo, definisana je klasa klasa i njena metoda *numberOfElementsInDsl* za određivanje broja elemenata u procesu na osnovu broja praznih linija, prikazana na listingu 2.

```
numberOfElementsInDsl: stringArray

          | numberOfElements |
          numberOfElements := 0.
          2 to: (stringArray size-1) do: [ :i
|
                    (stringArray at: i) = ''
                              ifTrue: [
numberOfElements := numberOfElements + 1. ]
].
          numberOfElements := numberOfElements
+ 1.
        ^numberOfElements.
```

Listing 2. Metoda za računanje broja elemenata

Na osnovu dobijenog broja elemenata u modelu procesa, jedan primer implementacije je metoda koja kreira dvodimenzionalni niz elemenata čiji redovi će biti rezervisani za jedan element iz procesa, a kolone za obeležja tih elemenata, npr. za *description*, *script*, *value*, *condition*, *go to*, itd.

Metoda *splitArray* klase *ModelElement* prikazana je na listingu 3.

```
splitArray: stringArray numberOflementsDsl:
aNumberOflementsDsl
          | arrayOfArrays j k red |
          arrayOfArrays := Array2D new:
aNumberOflementsDsl.
          j := 1.   "row"
          k := 1.   "column"
          2 to: stringArray size - 1 do: [ :i
|
                    row := stringArray at: i.
                    row = ''
                              ifFalse: [
arrayOfArrays at: j at: k put: row.
                                        k := k
+ 1. ]
                              ifTrue: [ j := j
+ 1.
                                        k := 1.
] ].
        ^arrayOfArrays.
```

Listing 3. Metoda za računanje broja elemenata

Kada je model podeljen po redovim i kolonama matrice, sledeći korak bi bio dobijanje niza elemenata u kome bi bili nazivi elemenata iz modela. Na listingu 4 prikazana je metoda klase ArrayProcessing *getNamesOfElements*, čiji su parametri dvodimenzionalni niz dobijen u prethodnom koraku i broj elemenata u procesu dobijen u koraku ranije.

```
getNamesOfElements: array2d numberOfElements:
aNumber

          | arrayProcessed processElementName
processElementNameTokeni processElement |
          arrayProcessed := Array new:
aNumber.
          1 to: aNumber do: [ :i |
                    processElement := array2d
at: i at: 1.
                    processElementNameTokeni
:= processElement findTokens: '"'.
                    processElementName :=
processElementNameTokeni at: 2 .
                    arrayProcessed at:i put:
processElementName . ].
        ^arrayProcessed.
```

Listing 4. Metoda za dobijanje naziva elemenata

Kada je niz podeljen po redovima i kolonama, određen broj elemenata u procesu i dobijeni nazivi elemenata procesa, mogu se pozvati instance klasa NewWave procesa i napraviti tok procesa-workflow. Na listingu 5 prikazan je deo metode *processArray* klase *ArrayProcessing*. Rezultujući niz NewWave elemenata realizovan je kao OrderedDictionary kolekcija.

```
processArray: array2d numberOfElements:
aNumber array: anArray
          | k r se sc ee exclusive bt1 bt2
receiver1 message1 result1 condition1
condition2 keyword1 argument1 receiver2
result2 receiver3 keyword2 argument2 result3
resultArray me numberOfElementsInExclusive
bt3 receiver4 condition3 keyword3 argument3
result4 |
          k := 1.
          r := 1.
          resultArray := OrderedDictionary
new.
          me:=ModelElement new.
        numberOfElementsInExclusive:=me
numberOfElementsInExclusive: array2d
aNumber: aNumber .
```

Listing 5. Deo metode za određivanje NW elemenata

Na listingu 6 prikazano je prepoznavanje *NWStartEvent* na osnovu naziva elementa iz tekstualnog DSL fajla.

```
1 to: aNumber do: [ :e |
(anArray at: e) = 'A StartEvent'
ifTrue: [ se := NWStartEvent new.
          se description: (((array2d at: r at:
k + 1) findTokens: '"') at: 2).
          resultArray at: 1 put: se.
          k := 1.
        r := r + 1 ].
```

Listing 6. Prepoznavanje NWStartEvent elementa

Na osnovu naziva elementa procesa, poziva se instance klasa NewWave procesa. Atributi NewWave elementa se dobijaju iz tačno definisanih pozicija u dvodimenzionalnom nizu. Pravljenje sekvence elemenata je implementirano kroz kolekciju *OrderedDictionary* i tako što se

trenutni element u procesu obrade dodaje kao sledeći element prethodnom elementu u rezultujućoj sekvenci. Ukoliko su prisutna grananja, u primeru procesa iz ovog rada to je *Exclusive gateway*, potrebno je izračunati broj elemenata nakon grananja i u tim elementima ispitati koji su sledeći i prethodni elementi u procesu.
U *ScriptTask*-u, blok koda u script atributu obrađen je preko obrazaca za unarne i *keyword messages*-poruke. U grananju je definisana provera uslova, a sami uslovi u task-ovima. Kada su prepoznati NewWave elementi i napravljena sekvenca elemenata, NewWave proces je spreman za izvršavanje. Važan element za izvršavanje procesa je *StartEvent*.

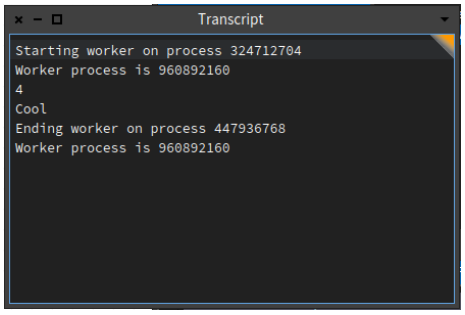
3.3 Izvršavanje procesa

Za samo izvršavanje NewWave procesa definisana je klasa *ProcessFlow* i metoda klase *makeFlow* koja ima parametar niz NewWave elemenata koji je dobijen izvršavanjem metode *processArray* klase *ArrayProcessing*, prikazano na listingu 7.

```
makeFlow: anArray

          | process engine |
          process := NWProcess
                    id: '1'
                    name: 'process1'
                    initialFlowElement:
(anArray at:1).
          engine := WaveEngine new.
          engine addProcess: process name:
'process1'.
          engine startProcess: 'process1'.
        ^anArray.
```

Listing 7. Metoda za izvršavanje procesa

ProcessHandler je zadužen za procese i kreira se kada je proces prosleđen engine-u – korak: *engine addProcess:*. WaveExecutor je sam izvršilac procesa i neophodan mu je element od koga će započeti izvršavanje procesa. U primeru u radu *initialNode* je StartEvent. Jedan proces može imati više izvršilaca, jedan je glavni, a drugi se kreiraju po potrebi. Rezultat izvršavanja procesa prikazan je u Transcript prozoru Pharo okruženja, prikazano na slici 6.



Slika 6. Rezultat izvršavanja NewWave procesa

4. ZAKLJUČAK

U ovom radu detaljno je opisan proces integracije alata OpenPonk i NewWave procesa, kao i svi problemi i njihova rešenja nastala tokom implementacije. Prvobitno je diskutovano na temu specifikacije modela procesa u BORM editoru OpenPonk-a, a zatim i dodavanje dodatnih atributa modelu radi lakšeg prepoznavanja NewWave elemenata. Pored toga, predstavljen je način izvršavanja NewWave procesa na osnovu prepoznatih NewWave elemenata. Opisani postupak donosi povezivanje NewWave workflow engine-a koji je implementiran od strane tima profesora i asistenata sa Fakulteta tehničkih nauka u Novom Sadu i OpenPonk alata implementiranog od strane i Centra za konceptualno modelovanje i implementaciju sa Češkog tehničkog univerziteta iz Praga.

Nedostatak implementacije parsera sa sobom donosi dodatne obaveze u radu sa modelom procesa. Potrebno je voditi računa o memorisanju tekstualnog modela procesa iz DSL editora BORM dijagrama. Drugi nedostatak ovog pristupa rada jeste veliki broj ispitivanja i određivanja elemenata u procesu. Implementacija rešenja na ovaj način nije najbrža niti optimalna, ali je predložen jedan pristup implementaciji u ovom radu.

5. LITERATURA

[1] „Modelovanje poslovnih procesa“, http://mf.unibl.org/upload/documents/Dokumenti/Predmeti/Informacioni%20sistemi/3%20-%20Modelovanje%20poslovnih%20procesa%20(DFD%2C%20IDEF0%2C%20UML).pdf, pristupljeno: 8. jun 2020.

[2] „Workflow Engine“, https://en.wikipedia.org/wiki/Workflow_engine, pristupljeno: 8. juna 2020.

[3] „Pharo“, https://pharo.org/, pristupljeno: 23. jun 2020.

[4] „NewWave“, https://github.com/skaplar/NewWave, pristupljeno: 23. jun 2020.

[5] „OpenPonk“, https://openponk.org/, pristupljeno: 23. jun 2020.

[6] Peter Uhnak, Robert Pergl, „The OpenPonk modeling platform“, IWST’16, Prague, Czech Republic.

[7] Sebastijan Kablar, Miroslav Zarić, Gordana Milosavljević, „NewWave Workflow Engine“, IWST’19, Cologne, Germany, avgust 2019.

Biografija:

Milica Marković rođena je 21. septembra 1994. godine u Novom Sadu. Osnovnu školu „Ivo Lola Ribar” završila je 2009. godine. Gimnaziju „Jovan Jovanović Zmaj” u Novom Sadu završila je 2013. godine. Iste godine upisala je Univerzitet odbrane, Vojnu akademiju, smer Vojnoelektronsko inženjerstvo, modul Informacioni sistemi. 2017. godine završava pomenute osnovne akademske studije i iste godine upisuje master akademske studije na Fakultetu tehničkih nauka u Novom Sadu.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE25Deket

POREĐENJE DOCKER SWARM I KUBERNETES

COMPARISON OF DOCKER SWARM AND KUBERNETES

Milan Deket, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu su opisani osnovni pojmovi i načini korišćenja Docker, Docker Swarm i Kubernetes alata. Objašnjeno je šta je Docker i šta su kontejneri, kao i kako se koriste. U poglavljima Docker Swarm i kubernetes se opisuje kako upravljati kontejnerima u kompleksnijim rešenjima koja mogu da se nalaze na više servera kao i kako pratiti stanje celog sistema i time se braniti od neočekivanih grešaka, odnosno neočekivanog gašenja sistema.*

Ključne reči: *Virtuelizacija, kontejneri, skaliranje aplikacija, docker, kubernetes, docker swarm*

Abstract – *The basic terms and ways of using Docker, Docker Swarm and Kubernetes tools are described in this paper. It explains what the Docker is and what the containers are, and how they are used. The Docker Swarm and Kubernetes chapters describe how to manage containers in more complex solutions that can be located on multiple servers, as well as how to monitor the state of the entire system and thus to defend themselves from unexpected errors, or unexpected crashing of the system.*

Keywords: *Virtualization, containers, application scaling, docker, docker swarm, kubernetes*

1. UVOD

Docker je nastao 2013. godine kao o*pen source* projekat od strane kompanije *dotCloud*. U toku prve godine razvoja projekat je porastao do veličine da je kompanije zatvorena i otvorena nova pod nazivom *Docker*, *Inc*. Pola decenije kasnije *Docker* je postao standard za serviranje *web* aplikacija.

Sve je počelo sa virtuelizacijom, jer su serveri postajali sve jači hardverski i veliku količinu vremena su bili u stanju mirovanja odnosno bez zadataka. Tako su programeri počeli da postavljaju više operativnih sistema na jedan računar i nastala je virtuelizacija i virtuelne mašine.

Sledeći veliki talas je bio takozvani *cloud*. Kompanija Amazon je predstavila svoje *cloud* rešenje pod nazivom *Amazon Web Services* ili skraćeno *AWS*. To je bilo jeftino i jednostavno rešenje za ljude koji su hteli da brzo dobiju veliku kompjutersku moć i mogućnost skaliranja aplikacija, ali da je isto tako brzo i ugase ukoliko im nije potrebna koristeći konzolu na *AWS*-u.

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

Nakon *cloud*-a stižu *Docker* kontejneri. To je bila još jedna velika promena u razvoja i serviranja aplikacija. Ova promena je kreirana da pomogne i programerima, dok su prethodne najviše uticale na sistem administratore. Ovu promenu su sa istim naporom mogli da nauče i programeri i administratori, a pomagala je i jednima i drugima u njihovim poslovima.

2. DOCKER

U današnje vreme se sve vrti oko brzine. Brzina razvoja aplikacija, brzina razvoja biznisa i svih ostalih stvari koje jednoj kompaniji mogu omogućiti veći profit za što kraće vreme. Koristeći *Docker* mogu da se osete benefiti, ali isto tako ako niste upoznati sa *Docker*-om i ostalim alatima neophodnim za održavanje kontejnera, oni mogu da vaš proizvod, odnosno aplikaciju odvedu u propast ili da je ugroze.

Docker pokriva ceo životni ciklus razvoja aplikacija. Aplikacija koja sadrži *frontend*, *backend*, *worker*-e i druge slojeve ili pod-aplikacije ima veliki broj zavisnosti, tj. oslanja se na funkcionalnost nekih drugih manjih aplikacija, biblioteka ili tehnologija, a onda može da se desi da te manje aplikacije ili biblioteke takođe imaju svoje zavisnosti što sve zajedno stvara jedan začarani krug. Ukoliko delovi glavne aplikacije rade na različitim operativnim sistemima, to značajno otežava razvoj i održavanje. Postoje rešenja za ovaj problem, kao što su na primer korišćenje virtuelnih mašina, instaliranje više operativnih sistema na jedan računar ili korišćenje više računara za razvoj jedne aplikacije, što je najgore rešenje. Nakon što uspešno postavite sve što vam je neophodno za razvoj aplikacije, potrebno je isto postaviti i na test, kao i na produkcionom okruženju, a zatim i na mašinu svakog novog programera koji se pridruži na razvoju aplikacije. Tu dolazimo do gore pomenute stvari, tj. brzine. Vreme potrebno da se sve ovo namesti će koštati kompaniju. Tu svoju primenu pronalaze Docker i kontejneri.

Docker će omogućiti da na isti način pokrenete i testirate aplikacije nezavisno od operativnog sistema. U današnje vreme 80 odsto vremena programera se svodi na održavanje aplikacije. To znači da malo vremena ostaje za dodavanje novih funkcionalnosti. Korišćenjem *Docker*-a možemo smanjiti procenat koji se odnosi na održavanje i pretvoriti to vreme u vreme za dodavanje novih mogućnosti na postojećoj aplikaciji ili kreiranju nove aplikacije.

2.1. *Docker image* i kontejneri

Image predstavlja sve zavisnosti, biblioteke, kod i operativni sistem koji su neophodni da pokrenu aplikaciju. Kontejner je *image* koji je pokrenut, od koje je napravljen proces. Da bismo počeli da koristimo *image* i

kontejnere najlakše je uzeti *image* koji je spreman za pokretanje. Najveća kolekcija za *Docker* se nalazi na *Docker hub*-u. *Web* adresa za *Docker hub* je hub.docker.com.

"*Docker image* je datoteka koja se sastoji iz više slojeva. Ona se koristi da bi se pokrenuo kontejner koji je opisan u datoteci. Datoteka se zapravo sastoji iz setova komandi i definisanih verzija aplikacija koje treba da se pokrenu, a zavisi od računara na kojem se pokreću. Takođe može da sadrži razne sistemske biblioteke koje će biti neophodne za pokretanje kontejnera, kao i alate i druge datoteke. Jedan *Docker image* može da se pokrene u obliku više kontejnera." [1].

Svaki *Docker image* počinje sa praznim slojem poznatijim kao *scratch*. Nakon toga svaka komanda koja se izvrši predstavlja novi sloj. *Docker image* može da ima jedan ili više slojeva. Kada pravimo novi *docker image*, počinjemo sa praznim slojem, tj *scratch*-om. Svaki novi sloj ima svoju jedinstvenu *hash* vrednost koja je kreirana *SHA* algoritmom, odnosno funkcijom. Jedinstvena *hash* vrednost pomaže sistemu da uporedi dva sloja i proveri da li su dva sloja ista ili različita.

U trenutku kada u *dockerfile*-u imamo definisane druge *image*-e i možda dva ili tri druga *image*-a imaju definisan Ubuntu kao prvi sloj, tad *hash* vrednost i keširani *image*-i pomažu. *Docker* neće dva puta skinuti Ubuntu, nego će preko *hash* vrednosti prepoznati da je to isti sloj i samo ga iskoristiti onoliko puta koliko je neophodno, što znači da ukoliko odlučimo da Ubuntu bude početni *docker image* za više slojeva, tada čuvamo samo jednu kopiju Ubuntu *image*-a. Od ove funckionalnosti najviše benefitujemo tako što čuvamo prostor na *hard* disku, smanjujemo protok saobraćaja koji treba da ide preko mreže na internet i povećavamo brzinu pokretanja kontejnera.

Docker image se identifikuje preko *image id*-a, repozitorijuma i taga, odnosno oznake. Tagovi su slični kao i *git* tagovi. Sa *docker* tagom zapravo možemo da skinemo *docker image* sa odredjenog *commit*-a.

2.2. Poređenje kontejnera i virtuelne mašine

Često na internetu možemo naići na poređenja kontejnera i virtuelnih mašina. Sličnosti postoje, ali u suštini to su dve različite stvari. Zapravo ih je teško i porediti, jer se poprilično razlikuju. Kontejner je samo proces. Proces koji radi na postojećem operativnom sistemu. To je proces koji ima ograničene resurse unutar operativnog sistema i samim tim se značajno razlikuje od virtuelne mašine.

2.3. *Docker* mreža

Docker svojim unapred podešenim opcijama omogućava da se kontejneri lako startuju za ljude koji tek počinju sa učenjem i upoznavanjem *Docker* kontejnera, ali postoji veliki broj opcija koji može dodatno da se konfiguriše, kao što je na primer mreža unutar kontejnera.

Kada se pokrene kontejner, čak i bez navođenja kojoj mreži da pripada, on će biti uključen u mrežu koju je *Docker* za korisnika unapred kreirao pod nazivom *bridge* mreža. *Bridge* je privatna virtuelna mreža kreirana od strane *Docker*-a. Svaka mreža, koju je *Docker* unapred kreirao ili je korisnik naknadno napravio za sopstvene potrebe, a koristeći *Docker*, je priključena na *NAT Firewall* - operativnom sistemu računara na kojem je *Docker* pokrenut. *Docker* će uraditi svu potrebnu konfiguraciju da bi kreirani kontejneri mogli da se povežu na internet.

U prethodnih nekoliko primera sa *Nginx*-om prilikom pokretanja smo koristili opciju *-p* da bismo otvorili *port*, odnosno saobraćaj sa operativnog sistema na određenom portu preusmerili na određen *port* unutar kontejnera. Ukoliko želimo da kontejneri međusobno komuniciraju, nema potrebe da otvaramo *port*-ove i time stvorimo bezbednosne probleme. Dovoljno je samo da dva kontejnera koja treba da komuniciraju budu unutar iste mreže. Broj mreža koje korisnik može da kreira nije ograničen, tako da može da ima jednu mrežu po aplikaciji odnosno kontejneru, a isto tako kontejner ne mora biti priključen ni na jednu mrežu.

2.4. *DNS* u *Docker*-u

Veoma je važno znati da se korisnik ne sme osloniti na *IP* adrese kontejnera u mreži, obzirom da su one dinamičke tako da je krucijalno da se koriste nazivi kontejnera, jer se kontejneri konstantno pokreću, uništavaju, premeštaju i slično. To znači da se i *IP* adrese kontejnera menjaju. Ne možemo se čak ni osloniti da će iz minuta u minut *IP* adresa biti ista, jer može da se desi da je kontejner uništen ili da je greška izazvala kontejner da se stopira i onda će *Docker* da se pobrine da pokrene novi kontejner, ali ne možemo znati na kojoj *IP* adresi.

Zato *Docker* koristi *DNS* (*Domain Name System*). "*DNS* je, u osnovi, sistem koji pretvara imena računara (*hostname*) u *IP* adrese. DNS takodje obezbeđuje podatke i o serverima elektronske pošte na domenu (MX), početnom *DNS* serveru (*SOA*) i druge. *DNS* je zasnovan na hijerarhijskom principu i jedna je od osnovnih komponenti interneta." [2]. U unapred kreiranoj *bridge* mreži kontejneri mogu da pronađu jedni druge samo preko *IP* adresa, dok u mreži koju korisnici kreiraju mreža ima automatski podržanu *DNS* opciju, tako da se kontejneri mogu pronaći i preko imena, a ne samo preko *IP* adresa. Ukoliko korisnik ima dva kontejnera, jedan pod nazivom *mysql* i drugi pod nazivom *web-app*, *web-app* kontejner će moći da pristupi *mysql*-u pozivajući *mysql* sa *port*-om 3306, tačnije *mysql*:3306, ako je *mysql* pokrenut na svom podrazumevanom *port*-u.

2.5. *Dockerfile*

Dockerfile je datoteka koja u sebi sadrži navedene korake za kreiranje određenog *docker image*-a. *Dockerfile* podseća na *shell* skriptu, ali nije, to je potpuno drugačiji jezik koji je jedinstven za *docker*. Kada se kreira *dockerfile* podrazumeva se da tako i nazovemo datoteku, ali nije pogrešno ni da je nazovemo drugačije. Prilikom pokretanja se mora eksplicitno naglasiti naziv datoteke.

Prva komanda koju treba navesti je *FROM* komanda i ona se nalazi u svakom *dockerfile*-u i obavezna je. Koristi da se opiše od kojeg *docker image*-a će početi da se gradi *Docker* kontejner.

ENV komanda se koristi za podešavanje *environment* varijabli. Ova komanda je veoma bitna za kreiranje kontejnera jer je ona glavni način da se proslede *key-value* vrednosti za kreiranje kontejnera ili za već pokrenute kontejnere. Jedan od razloga zašto se *environment*

varijable preferiraju je zato što su podržane na svakom sistemu.

Red navođenja komandi u *dockerfile*-u je bitan s obzirom da ih *Docker* izvršava *top-down* pristupom, što znači da kreće od prve navedene komande i redom prelazi na sledeću navedenu komandu. U pozadini *Docker* izvršava *shell* komande unutar kontejnera.

RUN komanda može da pokreće *shell* komande čije izvršenje nam je neophodno unutar kontejnera. To može biti *updatepackage manager-a* operativnog sistema, instaliranje neke aplikacije ili otpakivanje određene datoteke.

2.6. Perzistencija podataka

Kontejneri su *immutable* i privremeni što znači da oni nikad ne menjaju stanje, nego kreiraju novo i nisu namenjeni da dugo rade dok ne budu uništeni. Ideja je da se kontejneri mogu u bilo kom trenutku odbaciti ili uništiti, dok se uvek može kreirati novi iz *docker image*-a. Ovaj koncept kontejnera da se ne menjaju i da budu privremeni nas tera da ne menjamo stvari kada se pokrenu.

Ako treba da se desi promena u konfiguraciji ili hoćemo da pokrenemo noviju verziju kontejnera onda ćemo stopirati i uništiti stare kontejnere i pokrenuti nove. Ovakav koncept ima svoje prednosti i mane. Šta se dešava ukoliko kontejner sadrži bazu podataka koju treba da sačuvamo? Nju ne možemo uništiti jer ćemo time izgubiti podatke važne za korisnike aplikacije.

U idealnom scenariju kontejneri ne bi trebali da sadrže te osetljive podatke koje bismo snimili u bazu u kombinaciji sa datotekama koje pokreću samu aplikaciju. Ovakav koncept je poznat kao *separation of concerns*.

Kontejneri kao što su *nginx* i *mysql* su već bili konfigurisani da čuvaju podatke sve dok kontejner nije uništen i samo restartovanje ili stopiranje ove dve vrste kontejnera ne bi uništilo podatke koji su kreirani za vreme njihovog rada. Pre 10 godina ovaj problem nije postojao jer su svi podaci bili sačuvani direktno na disk računara i nije bilo kontejnera, tako da sistem administratori nisu morali da brinu o ovom problemu.

Sve je samo po sebi bilo perzistentno. U doba kontejnera i automatskog skaliranja aplikacija ovo predstavlja jedinstven i velik problem. *Docker* nudi dva rešenja za ovaj problem:

1. data volumes
2. bind mounts

Prva opcija, *data volumes*, kreira specijalnu lokaciju izvan kontejnera na kojoj će se čuvati podaci koji zahtevaju perzistenciju. Ovo će sačuvati podatke iako obrišemo kontejner i omogućiti nam da posle tu lokaciju dodamo bilo kom drugom kontejneru. Kontejner to vidi kao putanju do datoteka.

Druga opcija nam omogućava da delimo direktorijum između *host*-a i *Docker*-a, odnosno između računara na kojem je pokrenut *Docker* kontejner i samog kontejnera. Za kontejner ova putanja će biti kao i bilo koja druga lokalna putanja, jer neće biti svestan da se zapravo taj direktorijum ili datoteka nalazi na *host*-u.

3. *DOCKER SWARM*

Velika prednost kontejnera je ta što aplikacije možemo da pokrenemo na bilo kojoj platformi i na bilo kojem *hardware*-u. Aplikacije se pokreću i na različitim *cloud* provajderima - od *AWS*-a, preko *Azure*-a, *Digital Ocean*-a, do *Google Cloud*-a. Problem nastaje kada imamo na stotine kontejnera koji su rasprostranjeni po raznim serverima. To stvara veliki problem za administratore. Ukoliko ste velika organizacija kao što je *Netflix* koji imaju jako veliki broj inženjera koji održavaju na hiljade kontejnera onda problem i nije toliko velik jer imate veliki broj ljudi koji će da prate stanja tih kontejnera i servera, ali ukoliko ste mala organizacija i imate samo jednog čoveka zaduženog za sve to, onda je problem znatno veći i odgovori na pitanja kako pokrenuti sve kontejnere, kako ugasiti samo neke, kako ih pokrenuti ponovo, kako ih obrisati i update-ovati postaju znatno komplikovaniji. Upravo kao odgovor na ovo pitanje 2016. godine kreiran je *Swarm* režim. To je *Swarm* koji danas poznajemo. Swarm je postojao i pre, za verzije koje su bile starije od 1.12, ali nije obavljao sav potreban posao. 2016. godine na konferenciji u vezi sa *Docker*-om koja se zove *Dockercon* objavljen je *SwarmKit*. *SwarmKit* je set biblioteka kojim je omogućen veliki broj novih funkcionalnosti za *Swarm* režim.

Swarm je zadužen za upravljanje grupom kontejnera (*cluster management*). Sastoji se od više *host*-ova koji koriste *Docker,* koji je pokrenut u *Swarm* režimu.

Nakon instalacije *Docker*-a, *Swarm* režim nije uključen. Potrebno je pozovati dodatnu komandu da bi se *Swarm* uključio. To je urađeno kako se ne bi uticalo na postojeći *Docker* sistem ili na alate koji su se oslanjali na *Docker*.

3.1. Vrste *node*-ova u *Swarm*-u

Node je instanca *Docker*-a koja je uključena u *Swarm*. Jedan ili više *node*-ova može biti pokrenuto na jednom *host*-u, odnosno računaru ili *cloud*-u. Za produkciju je uglavnom preporučljivo da se *node*-ovi nalaze na što više fizički odvojenih mašina. Za pokretanje *Swarm*-a zadužen je menadžer *node*. Menadžeri imaju tri zadatka:

1. Da održavaju stanje klastera
2. Da planiraju pokretanje servisa
3. Serviraju *HTTP API endpoint*-e za *Swarm*

Koristeći Raft, menadžeri uspevaju da održe konzistentno stanje celog *swarm*-a i svih servisa koji su pokrenuti u njemu. Za testiranje je u redu da imamo samo jednog menadžera, ali za produkciju je ipak potrebno više u slučaju da taj jedan menadžer prestane da radi, potrebno je imati druge koji će da preuzmu posao. Menadžeri mogu da rade posao kao *worker*-i. Menadžeri su zapravo *worker*-i sa dozvolom da kontrolišu *Swarm*. Menadžere možemo učiniti *worker*-ima, ali isto tako i *worker*-e možemo promovisati u menadžere.

3.2. Arhitektura menadžera *node*-a

Menadžer *node* se sastoji iz 5 glavnih delova, a to su: *API*, *Orchestrator*, *Allocator*, *Scheduler*, *Dispatcher.*

API je zadužen za prihvatanje komandi od klijenta, u našem slučaju terminala.

Orchestrator služi za evaluaciju stanja klastera. On upoređuje stanje koje je korisnik deklarisao da želi u odnosu na ono stanje u kojem je klaster trenutno. Ukoliko korisnik želi da doda jednu instancu nekog kontejnera, *orchestrator* će uporediti željeno stanje klastera, a to je da treba da ima jednu instancu pokrenutu, sa trenutnim stanjem u kojem na primer nema nijednu pokrenutu instancu. Njegov odgovor će biti da treba kreirati jednu instancu željenog kontejnera, tj. kreiraće novi zadatak koji će biti neophodno izvršiti. Zadatak u ovom trenutku još uvek nije dodeljen niti jednom *node*-u.

Allocator služi da dodeli resurse novom zadatku. On će uzeti komandu koju je korisnik uneo kroz *API*, na primer kreiranje novog servisa i novi zadatak koji je kreirao *orchestrator* i za njih će alocirati *IP* adresu.

Scheduler je zadužen za dodeljivanje zadataka *worker node*-ovima. *Scheduler* konstantno prati da li je kreiran neki novi zadatak. On će preuzeti zadatak, proveriti koji su *node*-ovi dostupni i imaju mogućnost da izvrše taj zadatak. Mogućnost izvršavanja nekog zadatka može da zavisi od resursa određene mašine, tj. da li ima dovoljno memorije, zauzetosti procesora, količine već pokrenutih kontejnera, itd. Nakon toga, ukoliko ima više *node*-ova koji mogu da izvrše zadatak, on će ih sortirati i uzeti onaj za koji misli da je najbolji za izvršenje tog zadatka.

Dispacther je komponenta na koju se svi *worker node*-ovi povežu i odgovaraju. Svaki *worker* kada želi da se poveže na *dispatcher* će prijaviti koliko resursa ima, koliko kontejnera je pokrenuto, itd. Takođe, konstantna komunikacija se dešava izmedju *worker*-a i *dispatcher*-a na menadžer *node*-u, takozvani *heartbeat*, da bi menadžer znao da li je *worker node* još uvek tu ili se ugasio. Njegova glavna uloga je prosleđivanje zadataka *worker node*-ovima. On će da proveri da li postoji neki novi zadatak koji je kreiran i za koji je određen koji *worker* treba da ga izvrši, ukoliko ima on će proslediti taj zadatak određenom *worker*-u.

4. KUBERNETES

Kubernetes je *open-source orchestrator* sistem za *Docker* kontejnere, što znači da kontroliše životni ciklus kontejnera na klasteru koji može da ima više fizičkih mašina. Može da pokrene više kontejnera na jednoj mašini, dok više mašina onda može da čini klaster. *Kubernetes* kao i *Docker Swarm* može da pokrene razne vrste aplikacija, ali postoje razlike u odnosu na S*warm*. Kontejnere možemo pokretati na određenom *node*-u zadajući parametre koje *node* treba da ispuni da bi se na njemu pokrenuo kontejner. Isto tako, kontejnere lako možemo prebacivati sa *node*-a na *node* ukoliko koristeći *Kubernetes*, na primer, treba da uradimo neku vrstu održavanja na jednoj od mašina.

Kubernetes projekat je podržan od strane *Google*-a i projekat se može naći na *Github*-u. *Google* koristi kontejnere duže od jedne decenije i svo iskustvo korišćenja i održavanja kontejnerizovanih aplikacija je iskorišćeno prilikom kreiranja *Kubernetes* projekta.

Kubernetes kao i *Docker Swarm* se može instalirati i pokrenuti na mnogim platformama. Na nekim platformama je podržan veći broj funckionalnosti. Platforme sa najboljom podrškom su *AWS* i *Google Cloud*. Na primer, na nekim platformama nisu podržani *volume*-i i eksterni *load balancer*-i, ali to ne znači da se *Kubernetes* ne može koristiti, nego da samo ove dve navedene funckionalnosti ne mogu.

5. ZAKLJUČAK

Kubernetes i *Swarm* nisu konkurentski alati. Oni omogućavaju isto krajnje rešenje, ali u zavisnosti od raznih faktora korisnik bi trebao da zna koji alat da koristi. *Kubernetes* je bolje koristiti u sledećim slučajevima:

1. Starije i stabilnije rešenje sa boljim *monitoring* opcijama
2. Za razvoj kompleksnijih aplikacija
3. Veliki broj sistema u klasteru

Sa druge strane, S*warm* je bolji za:

1. Korisnike koji imaju samo osnovni nivo znanja o *Docker*-u i ne žele previše vremena da potroše na instalaciju i konfiguraciju klastera
2. Razvoj manje kompleksnih aplikacija

Swarm pruža jednostavno rešenje koje omogućava korisniku da brzo kreira i pokrene svoj klaster, dok *Kubernetes* podržava veće i kompleksnije zahteve. *Swarm* je popularniji među programerima koji preferiraju brzo startovanje aplikacije na klasteru, bez puno konfiguracije, dok je *Kubernetes* korišćeniji u produkcionim rešenjima od strane visokoprofilisanih interenet kompanija koje su napravile popularne servise koje koriste veliki broj ljudi.

6. LITERATURA

[1] Docker image, https://searchitoperations.techtarget-.com/definition/Docker-image.

[2] Wikipedia, https://sr.wikipedia.org/wiki/DNS.

Kratka biografija:

Milan Deket rođen je u Subotici 1992. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Veštačka inteligencija odbranio je 2020.godine.
Kontakt: milandeket@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE26Sekulic

AUTOMATSKO TESTIRANJE ADMS FUNKCIONALNOSTI ZA UČITAVANJE SNIMKA DINAMIČKIH PODATAKA

AUTOMATED TESTING OF ADMS LOAD SNAPSHOT FUNCTIONALITY

Dragiša Sekulić, Savo Đukić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je izvršena analiza automatskog testiranja ADMS funkcionalnosti za učitavanje snimka dinamičkih podataka. Predstavljene su osnovne vrste testiranja softvera, kao i razlike između manuelnog i automatskog testiranja. Opisani su alati i okruženje korišćeni za pisanje automatskih testova. Na kraju rada dati su rezultati izvršavanja napisanih automatskih testova, kao i zaključci do kojih se došlo izvršenom analizom.*

Ključne riječi: *Automatski test, AMDS softver, funkcionalnost za učitavanje snimka dinamičkih podataka.*

Abstract – *In this paper the analysis of automated testing of ADMS Load Snapshot functionality is performed. The basic types of software testing, as well as the differences between manual and automated testing, are presented. Tools and environment used for writing automated tests are also described. At the end, results of automated tests are provided, as well as the conclusions derived based on performed analysis.*

Keywords: *Automated test, ADMS software, Load snapshot functionality.*

1. UVOD

Testiranje je veoma značajna aktivnost u procesu životnog ciklusa razvoja softvera. U izradi softvera, testiranje predstavlja pokušaj da se pronađu greške u softveru koji je napravljen. Greške u softveru mogu prouzrokovati ogromne novčane štete, tako da se moraju uočiti i otkloniti što je moguće ranije [1].
Kroz ovaj rad realizovano je automatsko testiranje funkcionalnosti za učitavanje snimka dinamičkih podataka (eng. Load Snapshot) koja je implementirana u naprednom softveru za menadžment distributivnih elektroenergetskih sistema (ADMS). Pošto se automatsko testiranje softvera sve više koristi radi uštede vremena, javila se ideja da se automatizuje testiranje ove funkcionalnosti.

2. TESTIRANJE SOFTVERA

Testiranje, u užem smislu, predstavlja provjeru da li određeni softver u potpunosti odgovara originalnim korisničkim zahtjevima, tj. da li je u potpunosti implementiran prema korisničkim zahtjevima. Testiranje, u širem smislu, predstavlja sistem kontrole kvaliteta, što znači da se osim provjere softvera, provjeravaju i sve njegove prateće komponente i karakteristike [1].
Testiranje softvera još možemo definisati kao proces koji se koristi da bi se utvrdila ispravnost, potpunost i kvalitet razvijenog softvera, a sastoji se od nekoliko aktivnosti: planiranje testiranja, dizajn testova, izvršavanja testova, evaluacija testova [1].

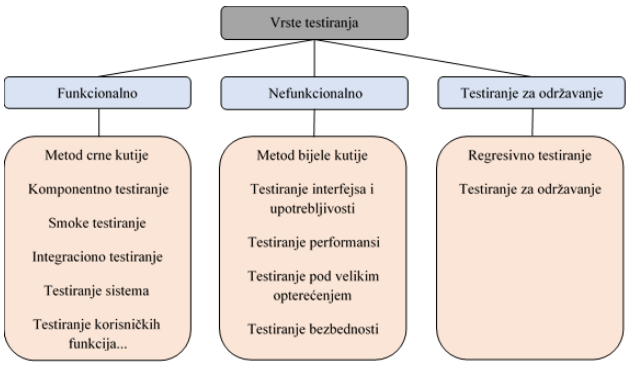
Testiranje se može vršiti manuelno i automatski. U tabeli 2.1 su date neke od razlika između manuelnog i automatskog testiranja [2].

Tabela 2.1. Razlike između manuelnog i automatskog testiranja

Parametar	Manuelno testiranje	Automatsko testiranje
Vrijeme obrade	Zahtjeva mnogo vremena i značajne ljudske resurse.	Znatno brže od manuelnog.
Pouzdanost	Manje pouzdano zbog mogućnosti ljudskih grešaka.	Pouzdanije je jer se izvodi pomoću alata i skripti.
Promjene korisničkog interfejsa	Ne utiču na manuelno testiranje.	Zahtevaju modifikacije automatskih testova.
Isplativost	Neisplativo za obimno regresivno testiranje.	Neisplativo za regresivno testiranje malog obima.
Programersko znanje	Nije potrebno.	Potrebno.
Idealan pristup	Kada je test slučaj potrebno izvršiti samo jednom.	Kada se često izvršavaju isti skupovi test slučajeva.

Testiranje softvera se najčešće dijeli na funkcionalno, nefunkcionalno i testiranje za održavanje, u zavisnosti od toga da li se kontrolišu samo krajnje funkcionalnosti ili i sam programski kod. Slika 2.1 prikazuje vrste testiranja [3].

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



Slika 2.1. Vrste testiranja

3. ALATI KORIŠĆENI ZA RAZVOJ AUTOMATSKIH TESTOVA

Za potrebe razvoja automatskih testova korišćeno je PyCharm okruženje, koje se koristi za programiranje u Python-u i koje rruža različite mogućnosti kao što su pomoć pri pisanju koda, grafički debager i integrisanu jedinicu tastera. PyCharm okruženje podržava rad na više različitih platformi (Windows, Linux), kao i rad sa JSON fajlovima, koji se koriste za čuvanje i razmjenu različitih tipova podataka [4].

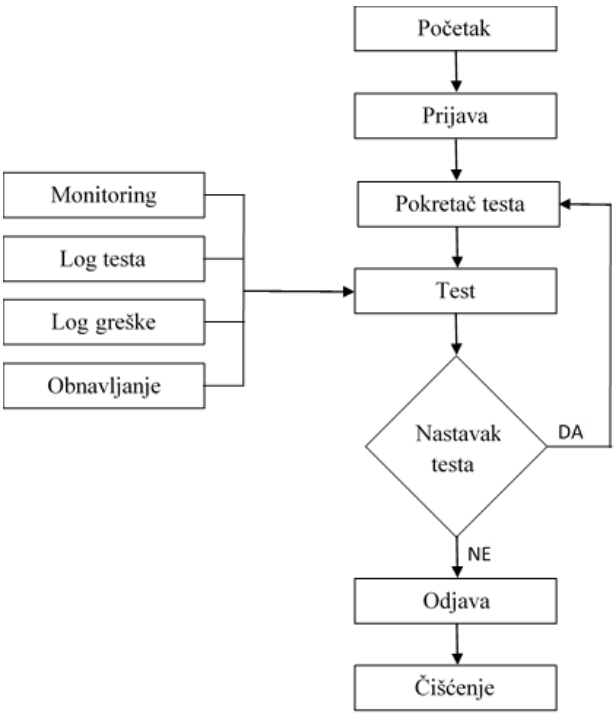
Kao osnova za razvoj automatskih testova korišćeni su Microsoft SQL Server i SQL Server Management Studio. SQL Server predstavlja integrisano riješenje za upravljanje bazama podataka. SQL Server Management Studio je alat koji pojednostavljuje upravljanje relacionom bazom podataka u sistemu SQL Server-a [5].

4. OKRUŽENJE AUTOMATSKOG TESTIRANJA

Uloga okruženja u kojem se razvijaju automatski testovi je ista kao arhitektura softvera u procesu njegovog razvoja. Okruženje automatskog testiranja definiše zajedničke funkcije, standardne testove, oslikava cjelokupnu strukturu u kojoj će se realizovati automatski testovi i definiše pravila prilikom imenovanja, dokumentovanja i upravljanja testovima. Dakle, okruženje daje mogućnost razvijanja održivih automatskih testova [6].

Zajedničke funkcije su funkcije koje svaki automatski test koristi prilikom svog razvoja i izvršavanja. Funkcije koje mogu biti sadržane u svim automatskim testovima su npr. ispisivanje loga testa, loga greške, rezultata izvršavanja, itd. Navedene funkcije se mogu pozivati iz bilo kog dijela automatskog testa, jer su realizovane kroz potprograme. Na slici 4.1 je prikazan algoritam zajedničkih funkcija [6].

Početak je funkcija koja priprema okruženje u kojem će se izvršavati automatski test. **Prijava** je funkcija koja pokreće okruženje ADMS softvera, provjerava da li je aplikacija dostupna za automatsko testiranje i provjerava validnost korisničkog imena i šifre za pokretanje aplikacije. **Pokretatač testa** je potprogram koji pokreće automatske testove. **Monitoring** je funkcija koja vrši nadgledanje izvršavanja svakog potprograma automatskog testa i provjerava stanje softvera nakon svakog izvršenog koraka. **Log testa** u posebnu datoteku upisuje rezultate izvršavanja svakog koraka i vrijeme izvršavanja. **Log greške** u posebnu datoteku upisuje rezultate nakon svakog neuspiješnog izvršavanja koraka automatskog testa. **Odjava** je obrnuta funkcija od učitavanja – vraća softver u stanje prije izvršavanja automatskog testa. **Čišćenje** je funkcija koja je slična funkciji početak, sa razlikom što ova funkcija uklanja posledice izvršavanja potprograma automatskog testa.



Slika 4.1. Algoritam zajedničkih funkcija

5. OPIS FUNKCIONALNOSTI ZA UČITAVANJE SNIMKA DINAMIČKIH PODATAKA

Funkcionalnost za učitavanje snimka dinamičkih podataka omogućava učitavanje snimljenih dinamičkih podataka (stanja prekidača, vrijednosti mjerenja, itd.) koju su upisani u bazu podataka. Ova funkcionalnost pruža inženjerima mogućnost oflajn analize radnih stanja mreže iz prošlosti, učitavanjem dinamičkih podataka i pokretanjem elektroenergetskih funkcija, u cilju pronalaženja uzroka problema koji su nastali u mreži (npr. kvar na dijelu mreže ili prekid napajanja potrošača).

Korisnici funkciju za učitavanje snimka dinamičkih podataka mogu da koriste kao alat za rekreiranje stvarnih stanja iz prošlosti i kao alat za podršku „šta ako“ analizama. Ova funkcionalnost korisniku omogućava da nakon učitavanja snimka dinamičkih podataka mijenja konfiguraciju mreže i vidi uticaj koji bi ta promijena imala u određenom trenutku.

Takođe, funkcionalnost pruža mogućnost provjere uklopnog stanja prekidača, kao i vrijednosti mjerenja, u željenom trenutku u prošlosti. Validnost učitanih dinamičkih podataka zavisi isključivo od toga da li su podaci upisani u bazu podataka; ukoliko podaci nisu upisani, korisnik neće imati validno stanje. Naravno, rad sa ovom funkcionalnošću je moguć samo u simulacionom modu ADMS-a (učitavanje nije moguće u realnom vremenu).

6. AUTOMATSKI TESTOVI ZA VERIFIKACIJU FUNKCIONALNOSTI UČITAVANJA SNIMKA DINAMIČKIH PODATAKA

U nastavku su dati primjeri realizovanih automatskih testova. Prva tri testa verifikuju uklopna stanja prekidača, a druga tri vrijednosti merenja. U nastavku je za svaki test data svrha, podaci od interesa i očekivani rezultat:

- Test 1 – Verifikacija uklopnog stanja prekidača na osnovu istorijskih podataka isčitanih iz baze podataka. Podaci od interesa su trenutak promjene stanja i vrijednost stanja prekidača. Očekivani rezultat testa je da se vrijednost stanja prekidača isčitana iz baze podataka poklapa sa vrijednošću iz kontrolnog prozora za učitani snimak iz prošlosti.
- Test 2 – Verifikacija uklopnog stanja prekidača na osnovu istorijskih podataka isčitanih iz baze podataka za željeni vremenski period. Podaci od interesa su trenutak promjene stanja i vrijednost stanja prekidača. Očekivani rezultat testa je da se vrijednost stanja prekidača isčitana iz baze podataka poklapa sa vrijednošću iz kontrolnog prozora za učitani snimak iz prošlosti.
- Test 3 – Verifikacija uklopnog stanja prekidača na osnovu istorijskih podataka isčitanih iz baze podataka. Podaci od interesa su trenutak promjene stanja i vrijednost stanja prekidača. Očekivani rezultat testa je da se vrijednost stanja prekidača isčitana iz baze podataka poklapa sa vrijednošću iz kontrolnog prozora za učitani snimak dinamičkih podataka iz prošlosti.
- Test 4 – Verifikacija vrijednosti mjerenja na osnovu istorijskih podataka isčitanih iz baze podataka. Podaci od interesa su trenutak promjene vrijednosti i vrijednost mjerenja. Očekivani rezultat testa je da se vrijednost mjerenja isčitana iz baze podataka poklapa sa vrijednošću iz kontrolnog prozora za učitani snimak dinamičkih podataka iz prošlosti.
- Test 5 – Verifikacija vrijednosti mjerenja na osnovu istorijskih podataka isčitanih iz baze podataka za željeni vremenski period. Podaci od interesa su trenutak promjene vrijednosti i vrijednost mjerenja. Očekivani rezultat testa je da se vrijednost mjerenja iščitana iz baze podataka poklapa sa vrijednošću iz kontrolnog prozora za učitani snimak dinamičkih podataka iz prošlosti.
- Test 6 – Verifikacija vrijednosti mjerenja na osnovu istorijskih podataka isčitanih iz baze podataka. Podaci od interesa su trenutak promjene vrijednosti i vrijednost mjerenja. Očekivani rezultat testa je da se vrijednost mjerenja isčitana iz baze podataka poklapa sa vrijednošću iz kontrolnog prozora za učitani snimak dinamičkih podataka iz prošlosti.

6.1. Rezultati izvršavanja automatskih testova

Testiranje predmetne funkcionalnosti je izvršeno na testnom sistemu ADMS softvera u četiri ciklusa automatskog testiranja. U tabeli 6.1.1 i 6.1.2 su prikazani rezultati izvršavanja ciklusa.

Iz tabele 6.1.1 se vidi da je Test 3 u drugom ciklusu neuspiješno izvršen. Iz logova je zaključeno da je došlo do usporenja sistema, koje je prouzrokovalo da prozor sa signalima ne bude otvoren u predviđenom vremenskom periodu (greška 031). Ovaj ciklus automatskog testiranja je pokazao da postoji problem u performansama sistema.

Kao što se vidi u tabel1 6.1.2, i u trećem ciklusu je pao jedan test, dok je jedan test preskočen. Analizom logova Testa 4 je uočeno da se vrijednost mjerenja iščitana iz baze podataka ne poklapa sa vrijednošću u kontrolnom prozoru (greška 041). Što se tiče Testa 2, u zadatom vremenskom periodu nije bilo promijena uklopnog stanja prekidača, tako da je test preskočen.

Iz tabele 6.1.2 se vidi da su dva testa neuspiješno izvršena, Test 1 i Test 6. Analizom logova je utvrđeno zašto je došlo do pada navedenih testova. Problem koji je doveo do pada Testa 1 je to što se uklopno stanje prekidača iščitano iz baze podataka nije poklopilo sa uklopnim stanjem u kontrolnom prozoru prekidača (greška 011). Analizom logova Testa 6 je zaključeno da se istorijski podaci iz baze podataka nisu mogli iščitati za odabrano mjerenje, tj. pozvana metoda je vratila praznu listu, kao da nema upisanih podataka (greška 061).

Tabela 6.1.1. Rezultati izvršavanja prvog i drugog ciklusa

Broj testa	Prvi ciklus		Drugi ciklus	
	Status izvršavanja	Broj greške	Status izvršavanja	Broj greške
Test 1	Prošao	/	Prošao	/
Test 2	Prošao	/	Prošao	/
Test 3	Prošao	/	Pao	031
Test 4	Prošao	/	Prošao	/
Test 5	Prošao	/	Prošao	/
Test 6	Prošao	/	Prošao	/

Tabela 6.1.2. Rezultati izvršavanja trećeg i četvrtog ciklusa

Broj testa	Treći ciklus		Četvrti ciklus	
	Status izvršavanja	Broj greške	Status izvršavanja	Broj greške
Test 1	Prošao	/	Pao	011
Test 2	Preskočen	/	Prošao	/
Test 3	Prošao	/	Prošao	/
Test 4	Pao	041	Prošao	/
Test 5	Prošao	/	Prošao	/
Test 6	Prošao	/	Pao	061

U tabeli 6.1.3 su prikazana vremena trajanja izvršavanja testova, manuelno i automatski. Iz tabele se vidi da su vremena izvršavanja automatskih testova mnogo manja od vremena manuelnog izvršavanja istih. Kao što se iz ove tabele vidi, ukupno vrijeme trajanja manuelnog izvršavanja je 1489 sekundi, odnosno 24 minuta i 49 sekundi, dok je vrijeme trajanja automatskog izvršavanja 484 sekunde, odnosno 8 minuta i 4 sekunde. Dakle, vrijeme trajanja automatskog izvršavanja testova je oko 3 puta manje od manuelnog. Prikazana tabela pokazuje jednu od prednosti automatskog testiranja u odnosu na manuelno, a to je ušteda vremena potrebnog za izvršavanje testova.

Tabela 6.1.3. Poređenje vremena trajanja izvršavanja testova manuelno i automatski

Broj testa	Vrijeme trajanja manuelnog izvršavanja testa [s]	Vrijeme trajanja automatskog izvršavanja testa [s]
Test 1	277	107
Test 2	235	35
Test 3	261	136
Test 4	250	33
Test 5	206	36
Test 6	260	137
Ukupno vrijeme trajanja izvršavanja testova	1489	484

6.2. Detektovane prednosti i mane automatskog testiranja

Prednosti automatskog testiranja, detektovane kroz rad, su:

- Izvršavanje automatskih testova na novijim verzijama softvera; potrebne su minimalne modifikacije samo u slučaju da je došlo do unapređenja testirane funkcionalnosti.
- Potreban je manji broj test inženjera i manje vrijemena za testiranje, čime je postignuta mnogo bolja iskorišćenost resursa.
- Izvršavanje automatskih testova je moguće u bilo kom vremenskom periodu (24/7).
- Automatskim testiranjem se postiže smanjenje vremena potrebnog za isporuku softvera.

Mane automatskog testiranja, detektovane kroz rad, su:

- Test inženjer mora proći obuku za pisanje automatskih testova, odnosno mora da poznaje programski jezik u kom se realizuju automatski testovi. Na to se utroši određeno vrijeme, koje se kasnije isplati zbog uštede na vremenu izvršavanja testova.
- Modifikacije automatskog testa ili loša implementacija mogu dovesti do neuspiješnog izvršavanja.
- Ista greška u softveru može biti registrovana kroz više automatskih testova (ista greška može dovesti do pada niza automatskih testova).
- Automatski test može biti lažno neuspiješno izvršen. Nepravilna konfiguracija ulaznih parametara i nevalidno testno okruženje mogu biti razlozi lažnog neuspiješnog izvršavanja testa, koje nije posledica greške u softveru, niti loše napisanog automatskog testa.
- Automatski test može biti lažno uspiješno izvršen. Loš dizajn automatskog testa, nepravilno provjeravanje moguće greške u softveru i nepravilno navedeni očekivani rezultati mogu biti razlozi lažnog uspiješnog izvršavanja testa.

7. ZAKLJUČAK

Tema ovog rada je automatsko testiranje funkcionalnosti za učitavanje snimka dinamičkih podataka implementirane u ADMS softver. U prvom poglavlju rada je definisan proces, vrste i metoda testiranja. Nakon toga su opisani korišćeni alati i okruženje za automatsko testiranje, kao i sama funkcionalnost koja je predmet testiranja. Na kraju rada je dat opis napisanih automatskih testova i rezultati njihovog izvršavanja.

Automatizacija testova je realizovana upotrebom programskog jezika Python u razvojnom okruženju PyCharm. Realizovani automatski testovi verifikuju uklopna stanja prekidača i vrijednosti mjerenja. Istorijski podaci iščitani iz baze podataka su poređeni sa podacima prikazanim u ADMS klijentskim aplikacijama.

Automatskim testiranjem realizovanim kroz ovaj rad je postignuta ušteda vremena i potrebnih ljudskih resursa. Takođe je postignuta mogućnost testiranja 24/7, jer automatski testovi mogu da se izvršavaju u bilo koje doba dana i noći. Zaključak je da je automatsko testiranje pouzdanije, efikasnije i ekonomičnije u odnosu na manuelno, iako obe metode testiranja imaju svoje prednosti i mane.

8. LITERATURA

[1] Popović Jovan, *Testiranje softvera u praksi*, Računarski fakultet, Beograd, 2012.

[2] https://www.guru99.com/difference-automated-vs-manual-testing.html

[3] Glenford J. Myers, *The art of software testing*, John Wiley and Sons, New Jersey, 2004.

[4] Pedro Kroger, *Modern Python Development With PyCharm*, 2015.

[5] Paul Atkinson, Robert Vieira, *Microsoft SQL Server 2012 programiranje: Od početka*, CET, Beograd, 2013.

[6] Linda G. Hayes, *Automated Testing Handbook*, Software Testing Institute, Richardson, 2004.

Kratka biografija:

Dragiša Sekulić rođen je u Bijeljini 1990. godine. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Energetska elektronika i električne mašine odbranio je 2016. godine.

Savo Đukić rođen je u Novom Sadu 1983. godine. Doktorsku disertaciju odbranio je 2014. godine na Fakultetu tehničkih nauka iz oblasti Elektroenergetski sistemi.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE30Djuka

PRIMENA ALGORITAMA ZA REPROJEKCIJU RASTERSKIH SLIKA PRILIKOM PRIKAZA VIŠESLOJNIH MAPA

APPLICATION OF RASTER IMAGE REPROJECTION ALGORITHMS WHEN DISPLAYING MULTILAYER MAPS

Viktor Đuka, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj –*U ovom radu vrši se upoznavanje rada sa standardnim rasterskim map servisima, integracijom ovih servisa sa klijentskim aplikacijama i prikazom tile-ova. Takođe, glavna stavka zadatka jeste opis geografskih podataka u različitim projekcijama, način kako reprojektovati tile-ove kao i implementacija algoritma za reprojekciju tile-ova. Analizirana su tri algoritma, od kojih su dva implementirana, a jedan je opisan kroz tekst kako se može implementirati.*

Ključne reči: *rasterske mape, projekcije mapa, reprojekcija tile-ova, algoritam*

Abstract – *This paper presents introduction with standard raster map services, integration this services with client applications and reprezentation tiles. Also, the main problem in this paper is resolving to description of geographical data in different projections, method how to reprojection tiles from one to another projection and implementation algorithm for this reprojection. The methods were analyzed for three algoritms, two of which were implemented and one was decribed throughout the text how it can be implemented.*

Keywords: *raster maps, map projections, reprojection tiles, algorithm*

1. UVOD

Na samom početku važno je upoznati se sa geoprostornim podacima kao i pristupu istim preko računarskih mreža. Za korisnike ove aplikacije, to može biti zamena za razne papirne mape, karte i atlase kao i za pristup informacijama koje su nekad bile teško dostupne kao na primer satelitski snimci.
Pored opisa zadatka, biće opisani osnovni pojmovi koji su vezani za sam rad. Jedan od njih jeste geografski informacioni sistem (GIS) koji predstavlja informacioni sistem koji je dizajniran tako da radi sa podacima koji su referencirani prostornim ili geografskim koordinatama.
On se može smatrati nekom vrstom baze podataka sa specifičnim atributima za skladištenje prostorno referenciranih podataka, kao i skupom operacija za rad sa tim podacima [1]. Upotreba GIS-a u aplikacijama podrazumeva prikaz, analizu i pretragu satelitskih snimaka, ulica, puteva, raznih objekata itd. U ovom radu će biti prikazan klijentski deo aplikacije (WPF) koji će vršiti prikaz rasterskih podataka, tj, satelitskih snimaka željenih reprojektovanih tile-ova.

Serverski deo ovog zadatka jeste prikupljanje tile-ova sa eksternih servisa kojima se može pristupati. Na tim servisima se mogu čitati i prikazivati geoprostorni podaci, ali i objavljivati. To znači da je moguće da neko napravi svoje mape sa raznim podacima i zatim ih „objavi“ kao servis.

Što se tiče objavljivanja geoprostornih podataka i pristup njima, oni su regulisani sa OGC standardima. Takođe, u radu su opisani i WMS (Web Map Service) stardardi koji se koriste za rasterske podatke, tj. za georeferencirane slike. WMTS (Web Map Tile Service) standard je opisan za rasterske podatke, ali kod njih je akcenat na optimizaciji jer predviđa da se originalna georeferencirana slika deli na više delova (*tile*) za određene nivoe zumiranja. O ovim stvarima ima više u poglavlju 3.

Poglavlje 4 obuhvata opis projekcija mapa. One predstavljaju način na koji globus predstavljen u ravni. To zahteva transformaciju geografskih širina i dužina lokacija sa zemljine kugle u lokacije ravni.

Dalje je opisan klijentski deo aplikacije, odnosno koje sve feature-e poseduje aplikacija za reprojektovanje tile-ova. Klijentska aplikacija je WPF aplikacija koja ima razne mogućnosti poput prikaza tile-ove zadatih u nekom BBOX-u, iscrtavanje tačaka, linija, i kao glavnu karakteristiku, reprojektovanje.

Što se tiče 7. poglavlja, tamo je opisan algoritam reprojekcije korišćen u ovoj aplikaciji što predstavlja glavnu karakteristiku aplikacije. Sama aplikacija sadrži i serverski deo koji komunicira sa jedne strane sa klijentom, a sa druge strane sa eksternim servisima za dobavljanje geoprostornih podataka. U ovom poglavlju, takođe je prikazan pregled dosadašnjih rešenja, tj srodnih istraživanja, gde su navedeni srodni radovi, problemi koje su oni rešavali i po čemu se ovaj rad razlikuje u odnosu na njih.

Na kraju je opisan je zaključak gde su opisani predlozi za dalje širenje aplikacije.

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

2. SERVISI ZA MAPE

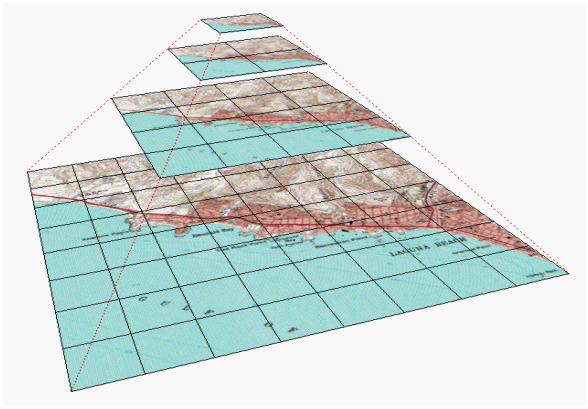
2.1 Tipovi servisa i osnovni standardi

Dobavljanje geografskih podataka se veoma često obavlja putem veb servisa. Veb servisi mogu biti i nestandardni, odnosno, specifični za nekog provajdera. Jedan od takvih tipova je ArcGIS servis. Takođe, Google Maps i Yahoo! Maps predstavljaju nestandardne servise koji se uveliko koriste u mnogim sistemima (npr. android uređaji). Standarde za ove veb servise objavljuje OGC (Open Geospatial Consortium).

Jedan od standarda koji podržava podelu mape na polja je WMTS (Web Map Tile Service). Pored standardnih i nestandardnih veb servisa, postoji i mogućnost da se geografski podaci dobavljaju i sa diska, što nije uobičajna praksa. Tu se može javiti problem sa prostorom na disku, ažuriranjem mapa, sinhronizacijom, itd.
Ono što je korišćeno u ovom radu, jesu standardni servis, odnosno WMTS. Ono što je prednost kod standardnih servisa jeste što se njihovo korišćenje smatra dobrom praksom jer poseduju veliku količinu geografskih podataka uskladištenih u raznim formatima. Zato je i poželjno imati neki standardizovan interfejs za pristup istim. Takođe, još jedan razlog zbog čega je dobro imati pristup standardizovan servisu, jeste taj da ko poznaje standard po kome se pravi neki servis za mape lako može da koristi servise različitih provajdera mapa.

2.2 Mapa podeljena na tile-ove i piramida polja

Na slici 2.1 možemo videti sliku Kalifornije u više nivoa. Svaki od tih nivoa ima svoj kvalitet slike. Tako, na primer, nivo koji je predstavljen samo jednim poljem, ima najniži kvalitet slike. Kako se nivo povećava, odnosno zoom level, tako raste i broj polja a samim tim i kvalitet slike se poboljšava. Na nivou sa jendim poljem se ne mogu videti neki detalji, na primer ulice, dok ukoliko broj polja ode na nekih deset hiljada, videće se lepo ulice. Polja po pravilu imaju fiksnu veličinu, npr 256x256 piksela. Ovakvim načinom podele mape na polja se dobija struktura koje se zove piramida polja (tile pyramid) ili piramida slika(image pyramid) (slika 2.1).



Slika 2.1 *Podela slike na mape polja Kalifornije i njen piramidni prikaz*

2.2 Prednosi i nedostaci *tiled* mapa

Podela mape na polja, odnosno, tile-ove, donosi neke prednosti i nedostatke. Ono što donosi prednost jeste zumiranje mape po svojim nivoima. Na taj način možemo da razdvojimo „bitne“ od „nebitnih“ stvari. Tako u nekim slučajevima, gde nam nije potreban prikaz svih detalja, već želimo da vidimo neki region samo gde se nalazi, dovoljan nam je niži zoom level. Time ne moramo sva polja da renderujemo i prikazujemo korisniku. Sa druge strane, ukoliko nam je potreban detaljan prikaz ulica, izrederovaćemo veći skup polja koji obuhvata taj region.
Primer za odsustvo podele na polja je **WMS standard** [3], gde je smanjena mogućnost keširanja, zato što klijent u tom slučaju zahteva sliku određene veličine, koja pokriva određeno geografsko područje, obuhvata određene slojeve, itd. Ovde je mala verovatnoća da će klijent više puta napraviti isti zahtev. Zbog toga, svaki put kad korisnik napravi zahtev, slike se generišu na serveru, što je proces koji je vremenski i računarski zahtevan i negativno utiče na skalabilnost servisa i smanjuje kvalitet servisa (QoS) za korisnike [2].
Dobavljanje polja preko mreže je popularan mehanizam preko koga klijenti za mape dobijaju podatke. Kod veb klijenata je to neophodno, a kod desktop klijenata smanjuje kompleksnost i veličinu instalacije [4].

3. OPEN GEOSPATION CONSORTIUM (OGC) STANDARDI

OGC je konzorcijum kompanija, nevladinih organizacija, agencija i univerziteta sa zajedničkom vizijom u kojoj svi imaju korist od geografskih informacija i servisa dostupnih preko različitih mreža, aplikacija i platformi. OGC nastoji da postigne globalni konsenzus pri razvoju standarda za interfejse i enkodiranje koji omogućavaju interoperabilnost među različitim geoprostornim bazama podataka, servisima i aplikacijama [3].

3.1 Web Map Service (WMS)

OGC standard definiše Web Map Service kao standard koji obezbeđuje HTTP interfejs koji služi za dobavljanje georeferenciranih slika iz jedne ili više distribuiranih geoprostornih baza podataka. U WMS zahtevu su definisani geografski slojevi i područje koje se želi prikazati. Odgovor na zahtev je jedna ili više georeferenciranih slika mapa (JPEG, PNG, itd) [3].

3.2 Web Map Tile Service (WMTS)

WMTS standard podrazumeva već pomenutu podelu mape na polja, i sve prednosti koje su navedene za takav pristup važe i ovde, što uključuje i ono najbitnije, a to je generisanje polja unapred, tj. serverski *tile cache*. Najbitniji resursi kojima se mogu pristupiti kod WMTS servisa su ServiceMetadata (metapodaci servisa) i Tile (polje). Njima se može pristupiti pomoću Get metoda [4].

4. PROJEKCIJE MAPA

Projekcije mapa predstavljaju način na koji globus predstavljen u ravni. To zahteva transformaciju geografskih širina i dužina lokacija sa zemljine kugle u lokacije ravni.

Transformacija projekcije mape [5] je jedno istraživačko polje projekcije mape koje proučava teorije i metode transformacije koordinate tačaka od jedne vrste projekcije do druge.
Postoje tri vrste metoda transformacije projekcije mapa [6]:

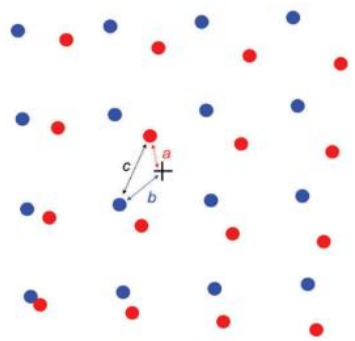
- Inverzna metoda transformacije.
- Direktni metod transformacije i kombinovani metod.
- Vrednost metoda transfomracije i vrednosno-analitička metoda transformacije.

U ovom konkretnom radu korišćene su dve projekcije. Jedna od njih je Web Mercator projekcija (EPSG:3857) koju ću malo detaljnije opisati, a druga je manje poznata EPSG:31256 projekcija.

5. REPROJEKCIJA TILE-OVA

Većina softvera danas koji koriste neke map servise imaju svoju predstavu u određenoj projekciji. Kako bi došlo do validnog prikazivanja geografskih podataka, potrebno je da se nekako usklade te projekcije. Tu se javlja pojam reprojekcije.
Pojam reprojekcije ima više značenja. U ovom slučaju, pojam reprojekcije predstavlja konvertovanje koordinata iz jednog koordinatnog sistema u željeni koordinatni sistem.
Primera radi, neki klijent radi sa jednom vrstom koordinatnog sistema (npr Web Mercator), i želi da predstavi neki region u drugoj projekciji. On će poslati podatke (npr Bbox uglova koordinata) serveru koji će mu obraditi te podatke, pronaći tile-ove, konvertovati u željenu projekciju i vratiti koordinate u novoj projekciji.

U literaturi se uglavnom opisuju drugačiji sistemi i nije ista tematika ali je suština ista.
U nastavku sledi pregled najsrodnijih radova.
U radu *Best practices for the reprojection and resampling of Sentinel-2 Multi Spectral Instrument Level 1C data* [8] je opisano da na svakih 10 metara reprojektovan piksel nekog tile-a, a distance na slici 2.1 možemo primetiti distance *a*, *b* i *c* koje su kvantifikovane tamo gde se preklapaju pločice.
Rastojanja *a* i *b* definišu greške u ovom algoritmu „najbližeg suseda“ , dok *c* kvantifikuje razliku u položaju izazvanu razmatranjem jednog, a ne drugog tile-a. Reprojektovani piksel predstavlja krstić na slici 7.8.



Slika 7.8 *Opis algoritma najbližeg suseda iz jednog od radova [8]*

6. ALGORITAM REPROJEKCIJE

Algoritam reprojekcije predstavlja način na koji ćemo jednu fotografiju (u našem slučaju jedan tile) u jednoj projekciji dobiti istu tu fotografiju u drugoj projekciji. Kako smo već gore naveli, zbog konverzije koordinata i njhove predstave u GIS-u neke koordinate jedne projekcije nisu iste u drugoj projekciji. Zato je potrebno njihovo reprojektovanje iz jednog u drugi sistem. U ovom poglavlju biće opisano nekoliko načina za ovaj poduhvat kao i načini koji su implementirani u ovom radu.

6.1 Opis prvog algoritma reprojekcije

Za početak, tile u EPSG:31256 je veličine 256x256 piksela. To je ukupno 65536 piksela koje sadrži ta jedna fotografija. U ovom algoritmu uzeti su u obzir apsolutno svi pikseli sa tog tile-a i svaki piksel je imao svoju poziciju na slici. Budući da je tile kvadrat, odnosno, dvodimenzijalni je tile, on u sebi ima svoje pozicije svih piksela. Pa tako za prvi piksel, pozicija je (0,0), drugi (0,1) itd.. Ovaj algoritam pamti svaku poziciju svih 65536 piksela. Za reprojekciju nam nisu dovoljni samo pikseli i pozicije već i koordinate tih piksela. Koordinate jednog tile-a dobijamo upotrebom WorldFile-ova [7].
Na osnovu WorldFile-a, imamo informaciju o gornjem levom uglu i njegovim koordinatama. Na osnovu pozicije tile-a izračunavamo i ostale uglove. Zatim na osnovu naših rubnih koordinata i na osnovu toga što znamo da po širini i po visini imamo 256 piksela, izračunavamo na koliko metara je sledeća koordinata. Taj podatak će nam biti ključan u ovom algoritmu. Zatim se iterira kroz svaku poziciju piksela i na poziciju gornjeg levog ugla se dodaje naš podatak koji smo izračunali u prethodnom koraku a to je na koliko metara je sledeća koordinata i pomnoži se sa pozicijom piksela. Ovom iteracijom dobijamo Dictionary sa 65536 podataka koji u sebi kao ključ sadrži koordinatu, a kao vrednost poziciju njegovu. Kako smo ovo izračunali prelazimo na konverziju ovih koordinata u EPSG:3857.
Za ovu konverziju koristili smo eksterni wrapper (proj_api) koji koristi eksternu biblioteku za konverziju koordinata. Svaku koordinatu smo konvertovali i smestili je u novi Dictionary koji sada u sebi sadrži koordinate u EPSG:3857 i pozicije na kojima treba da se nalazi.
Prilikom ove konverzije, odredili smo i rubne koordinate i njih sačuvali u posebne varijable kako bismo taj bbox poslali na naš server koji će nam dati tile-ove u EPSG:3857. U ovom slučaju, dobili smo 6 tile-ova za gore prikazanu fotografiju.
Nakon preuzimanja tile-ova, treba pronaći gde se koja koordinata nalazi u tim tile-ovima. Na osnovu WorldFIle-ova pronalazimo traženi tile i iz tog tile-a izvlačimo traženi piksel koordinate koju smo poslali da se pronađe. Tako radimo za svih 65536 koordinata. Svaki piksel koji izvučemo, smestimo ga u poseban Dictionary koji u sebi ima kao ključ poziciju piksela, a kao vrednost konkretan piksel. Na osnovu ovog Dictionary-a znamo na kojoj poziciji treba da se nalazi određeni piksel i tako ga iscrtavamo na našem canvas-u.
Ovaj algoritam se ispostavio da je efikasan sa stanovišta preciznosti, ali sa stanovišta vremena obrade ove reprojekcije, nije baš efikasan. Svakim pristupom eksternoj biblioteci za konverziju fajlova izgubi se puno vremena i to je veoma zahtevan pristup. Treba 65536 puta

pristupiti proj_api biblioteci, odnosno, 65536 puta je pozivati a to je veoma skupa operacija.
Zbog toga, implementiran je još jedan algoritam koji je brži.

6.2 Opis drugog algoritma reprojekcije

Drugi algoritam reprojekcije koji je implementiran u ovom radu je baziran na istom principu. Veoma je sličan prethodnom algoritmu pa će u ovom delu teksta biti opisano samo koja je razlika između njih.
Algoritam funkcioniše isto sve do onog momenta kada treba da se pristupa proj_api biblioteci. Kada se dođe do tog dela, uzima se svaki deseti piksel iz Dictionary-ja u kojem se nalaze sve koordinate iz EPSG:31256 projekcije i njihove pozicije. Pa tako kao krajnji rezultat imamo deset puta manje piksela za obradu (6554). Dalji tok ovog algoritma je isti kao što je prethodni opisen. Razlika je samo broj piksela za pristup proj_api biblioteci.
Pikseli koji nisu svaki deseti bili, njih smo računali na osnovu toga da se uzme vrednost poslednje koordinate i doda razlika dva piksela koja smo izračunali u metrima.
Tako ponovo imamo koordinate svih piksela koje treba kasnije da obradimo na isti način kao u prethodnom algoritmu samo sa 10 puta manjim pristupom proj_api-ju.

7. ZAKLJUČAK

U ovom radu analizirana su razni algoritmi za reprojektovanje piksela rasterskih tile-ova.
Implementirana su dva algoritma gde i jedan i drugi imaju svoje prednosti i mane.

Prvi algoritam je sa stanovišta efikasnosti veoma dobar, ali je spor. Drugi algoritam je dosta brži ali mu efikasnost nije jača strana.

U sistemima koji imaju naprednu tehnologiju i dobro razvijen informacioni sistem mogu koristiti prvi algoritam jer se ne bi osetio nedostatak brzine, a preciznost bi bila na najvišem mogućem nivou. Takođe, sistemi kojima je od bitnog značaja da reprojekcija bude veoma efikasna i da se detalji ispolje na najvišem mogućem nivou, oni bi svakako koristili ovaj algoritam.

Što se tiče drugog algoritma, on bi najefikasniji bio u sistemima u kojima bi se vršilo testiranje reprojekcija kako bi se na što brži način videlo da li odgovara dobijena projekcija onom što se očekivalo.

Takođe, implementirana je i klijentska aplikacija koja koristi ove algoritme za reprojektovanje rasterskih tile-ova.

Proširenje aplikacija moguće je u više smerova.

Jedan od smerova jeste taj da kao serverski deo, moguće da bilo koja druga klijentska aplikacija koristi server kako bi se izvršila reprojekcija tile-ova. Na samom serveru ostavljena je mogućnost za proširivanje broja projekcija koje bi se obrađivale.

Takođe, aplikacija može dalje implementirati nove algoritme, i to lakim ubacivanjem novih metoda. Isto tako, klijent može da se izmeni i upotrebi za neku svrhu testiranja raznih mogućnosti pre nego što se neka ozbiljna aplikacija ne odluči za korišćenje određene projekcije.

Iako su se neki algoritmi koji su implementirani dobro pokazali, ipak treba obratiti pažnju za budući razvoj aplikacije da se opterećenje i zauzeće radne memorije svede na minimum.

8. LITERATURA

[1] Jeffrey Star and John Estes, "*Geographic Information Systems: An Introduction*" Englewood Cliffs, NJ: Prentice-Hall, 1990, page 2-3

[2] Ricardo García, Juan Pablo de Castro, Elena Verdú, María Jesús Verdú and Luisa María Regueras, "Web Map Tile Services for Spatial Data Infrastructures: Management and Optimization", *Cartography – A Tool for Spatial Analysis*, Dr. Carlos Bateira (Ed.), InTech, 2012, dostupno na: http://cdn.intechopen.com/pdfs/38302/InTech-Web_map_tile_services_for_spatial_data_infrastructures_management_and_optimization.pdf

[3] Open Geospatial Consortium, dostupno na: http://www.opengeospatial.org/

[4] Open Geospatial Consortium Inc., *"OpenGIS Web Map Tile Service Implementation Standard"*, April 2010, dostupno na: http://portal.opengeospatial.org/files/?artifact_id=35326

[5] WU Z.X., HU Y.J. "The Development of Map Projection in China in the Past 30 Years (in Chinese)", *Map Projection Collected Papers (in Chinese)*, *Survey and Mapping Press*, pp.1-5, Beijing, 1983

[6] WU Z.X., YANG Q.H.: "A Research on the Transformation of Map Projections in Computer-Aided Cartography (in Chinese)", *Map Projection Collected Papers (in Chinese). Survey and Mapping Press*, Beijing: 287--322 (1983)

[7] WorldFile, dostupno na: https://www.usna.edu/Users/oceano/pguth/md_help/html/tbme6h0z.htm

[8] David P. Roy, Jian Li, Hankui K. Zhang & Lin Yan, „Best practices for the reprojection and resampling of Sentinel-2 Multi Spectral Instrument Level 1C data*"*, *Remote Sensing Letters*, Vol. 7, No. 11, pp. 1023-1032, DOI: 10.1080/2150704X.2016.1212419, 27 Jul 2016.

Svim linkovima je pristupljeno 16.10.2020.

Kratka biografija:

Viktor Đuka rođen je 09.02.1996. u Novom Sadu. Smer Računarstvo i automatiku na Fakultetu tehničkih nauka u Novom Sadu upisuje 2015. godine, a 2017. godine se opredeljuje za Primenjene računarske nauke i informatiku kao usmerenje. Školske 2019. Godine usmerio se na podusmerenje Softversko iznženjerstvo. Osnovne studije je završio 2019. godine i stekao zvanje diplomiranog inženjera elektrotehnike i računarstva. Master studije je upisao na istom usmerenju, takođe 2019. godine.
kontakt: viktordjuka10@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE31Marin

RAZVOJ APLIKACIJE ZA TESTIRANJE BEZBEDNOSTI WEB APLIKACIJA

DEVELOPMENT OF APPLICATION FOR TESTING WEB APPLICATIONS SECURITY

Ivana Marin, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu predstavljen je razvoj aplikacije za testiranje bezbednosti web aplikacija u Python programskom jeziku, kroz prethodno analizirane koncepte bezbednosti i ranjivosti web aplikacija i penetracionog testiranja.*

Ključne reči: *Bezbednost web aplikacija; Penetraciono testiranje; Ranjivosti web aplikacija*

Abstract – *This paper presents the development of application for testing Web applications security in Python programming language, through previously analyzed concepts of web application security and vulnerabilities and penetration testing.*

Keywords *Web application security; Penetration testing; Web application vulnerabilities*

1. UVOD

Danas, u 21 veku, milioni ljudi širom sveta stupaju u interakciju sa web aplikacijama svakog dana, bilo da su to društvene mreže, kupovina, bankarstvo, pošta ili potraga za informacijama.

Kompanije i organizacije kao proizvođači žele da svojim korisnicima omoguće najkvalitetniji softver i odgovornost koju imaju prema njima, a i prema tržištu je velika. Jedan od ključnih faktora jeste bezbednost web aplikacije. Bezbednost se već dugi niz godina definiše kao krucijalan faktor u izgradnji web aplikacija. Softveri, kao što su web aplikacije, svakodnevno su izložene hakerskim napadima, pokušajima neovlašćenog pristupa, krađi osetljivih informacija…

To su samo neke od kriminalnih aktivnosti koje se nalaze u okruženju web aplikacija. Stoga je neophodno obezbediti najveći nivo zaštite i omogućiti da aplikacija uspešno funkcioniše, da zaštiti sebe i svoje podatke u svakom trenutku, a naročito ukoliko dođe do malicioznih napada. Danas, više nego pre, kompanije ulažu u zaštitu svojih aplikacija i korisnika. Jedan od najboljih izbora za to predstavlja penetraciono testiranje web aplikacija.

2. ISTRAŽIVANJE O RANJIVOSTIMA WEB APLIKACIJA

Sa sigurnosnog aspekta, ranjivost predstavlja manu ili slabost u dizajnu, implementaciji, operaciji, koja može biti iskorišćena za kompromitovanje aplikacije.

Pretnja predstavlja bilo šta što može naškoditi aplikaciji, koristeći njenu ranjivost. To može biti maliciozni napadač koji dolazi u pristup podacima iz baze podataka.

Jedan od najvećih problema u bezbednosti predstavlja reaktivan pristup umesto proaktivnog. Reaktivnost znači da se problem već dogodio i da posledice treba sanirati. Proaktivnost treba da bude pravi pristup i kao rezultat toga, bezbednosni problemi biće otkriveni u ranijim fazama razvoja, a ne kada se desi napad [1]. Stoga je veoma važno integrisati bezbednost kroz sve faze razvoja softvera, od samih definisanja zahteva, preko dizajna, implementacije, testiranja, pa sve do puštanja u produkciju i održavanja softvera [1]. Međutim, čak i uz ispraćene sigurnosne prakse, velika je verovatnoća da će ostati sigurnosnih propusta.

Sa povećanjem broja organizacija koje posluju na mreži, web aplikacije i web serveri predstavljaju odličnu priliku za napadače. Za interakciju sa web aplikacijom, potreban je samo pregledač. Tokom godina, web aplikacije čuvaju ključne podatke kao što su lični podaci i finansijske evidencije. Kada se web aplikacija ne testira na ranjivosti i napadač dobije pristup podacima korisnika, to može ozbiljno uticati na vrednost brenda kompanije, ako korisnik pokrene postupak protiv kompanije zbog nedovoljnog rada na zaštiti njegovih podataka. To takođe može dovesti do gubitka prihoda, jer će se mnogi korisnici "preseliti" kod konkurenata koji bi obezbedili bolju sigurnost [2].

Stručnjaci iz OWASP zajednice sastavili su ono što se smatra konačnom standardnom listom glavnih ranjivosti web aplikacija. Neke od najpoznatijih i najčešćih ranjivosti sa ove liste su: *SQL Injection, Broken Authentication, Broken Access Control, Cross-site Scripting* (XSS) [3].

3. ISTRAŽIVANJE O PENETRACIONOM TESTIRANJU WEB APLIKACIJA

Penetraciono testiranje se definiše kao proaktivni način testiranja web aplikacija simulacijom napada, slično stvarnom napadu, postavljajući testera u ulogu napadača (hakera) [2]. Ovakvo testiranje vezano je ugovorom između testera i vlasnika aplikacije (klijenta) koja će se testirati [2].

Penetracioni testovi se pokazuju korisnim u pronalaženju ranjivosti u organizaciji i proveravanju da li će ih napadač moći iskoristiti da bi stekao neovlašćen pristup. Iz

---

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

izveštaja o testiranju mogu se proceniti potencijalni uticaji na organizaciju i predložiti kontramere za smanjenje rizika [4].

Pristupi penetracionom testiranju uključuju sledeće tipove: crna, bela i siva kutija [5].

Penetracioni test crna kutija ne zahteva prethodne informacije o ciljnoj mreži ili aplikaciji i izvodi se kao stvarni napad hakera [6]. U ovom slučaju testeru nisu date informacije o unutrašnjem radu web aplikacije, niti o izvornom kodu ili softverskoj arhitekturi [6]. U penetracionom testu bela kutija, testeru se pruža pun pristup i informacije o unutrašnjoj strukturi sistema - izvornom kodu [7]. Izvorni kod aplikacije omogućava da se izvrši statička/dinamička „analiza izvornog koda". Penetraciono testiranje siva kutija, nalazi se između testiranja crne i bele kutije, gde su neke informacije pružene, a neke skrivene [8]. Testerima se pruža dovoljno informacija za početak,

Kada je reč o kategoriji testiranja, važno je osvrnuti se i povući paralelu na tipove hakera koji postoje: crni, beli i sivi šešir.

Haker sa crnim šeširom svoje znanje koristi u negativne svrhe. Vođen je ličnom dobiti, uključujući zaradu iznudom ili drugim obmanjujućim metodama prikupljanja novca [9][10]. Haker sa belim šeširom se često naziva stručnjakom za bezbednost [10]. Takve hakere zapošljava organizacija i dozvoljeno im je da napadaju organizaciju kako bi pronašli ranjivosti koje bi napadač mogao iskoristiti [10]. Haker sa sivim šeširom predstavlja sredinu između belog i crnog šešira [9]. Na primer, haker sivog šešira radio bi kao stručnjak za bezbednost u organizaciji [9]. Međutim, može ostaviti ulaz da bi mu kasnije mogao pristupiti.

Postoji više metodologija penetracionog testiranja, ali je princip testiranja u svima veoma sličan. U ovom radu analiziran je pristup sa četiri faze:

Prva faza je prikupljanje informacija ili izviđanje i njen cilj je naučiti što više o klijentima [11]. Nakon svih prikupljenih informacija o meti, sledi faza skeniranja ranjivosti, koja obuhvata razumevanje kako će ciljna aplikacija odgovoriti na razne pokušaje upada [10]. Kada se završi faza skeniranja ranjivosti sledi eksploatacija ili iskorišćavanje ranjivosti. Ova faza predstavlja proces sticanja kontrole nad sistemom [10].

Za fazu eksploatacije veoma je važno da prethodne faze budu kompletirane, jer se uspeh ekploatacije zasniva na prethodno prikupljenim informacijama. Nakon završene eksploatacije neophodno je napraviti rezime u obliku izveštaja o izvršenom penetracionom testiranju [10].

Kad god je to moguće, prilikom pisanja detaljnog izveštaja o penetracionom testiranju, treba uključiti predloge za rešavanje problema koji su otkriveni. Pružanje rešenja za svaki problem koji se otkrije je vitalni deo svakog detaljnog izveštaja [10].

Uz temeljno analiziran teorijski deo, u nastavku je data njegova primena, kroz razvoj aplikacije kao alata koji će testirati bezbednost nekoliko web aplikacija.

4. RAZVOJ APLIKACIJE ZA TESTIRANJE BEZBEDNOSTI WEB APLIKACIJA KROZ PISANJE PENETRACIONIH TESTOVA

Aplikacija za testiranje implementirana je kao skup ručno pisanih skripti u *Python* programskom jeziku, unutar *Visual Studio Code* okruženja i namenjena je za sledeće vrste napada:

- *SQL Injection* (na prvom mestu OWASP TOP 10 liste)
- *Broken Authentication* (na drugom mestu OWASP TOP 10 liste)
- *Broken Access Control* (na petom mestu OWASP TOP 10 liste)

Testovi su implementirani uz pomoć *requests* i *BeautifulSoup Python* biblioteka.

Aplikacije koje se testiraju, u okviru pen test kućne laboratorije, pokrenute su na lokalu i sa aspekta bezbednosti predstavljaju aplikacije sa različitim nivoima zaštite, date sa ciljem testiranja bezbednosti i otkrivanja ranjivosti:

- *OWASP Juice Shop* – aplikacija sa predviđenim ranjivostima i izazovima različitih težina, namenjena vežbanju bezbednosti na web aplikacijama
- *Damn Vulnerable Web App* (DVWA) - namerno ranjiva web aplikacija, sa različitim nivoima težine i ciljem da pomogne stručnjacima za bezbednost da testiraju svoje veštine i alate, a programerima da bolje razumeju proces zaštite web aplikacije. Za potrebe testiranja odabran je visok nivo zaštite.
- *Mega Travel Booking* – web aplikacija za rezervaciju smeštaja, rađena kao studentski projekat

Pored manuelnih testova analizirani su i automatizovani testovi, uz pomoć poznatih, postojećih alata: *SqlMap* (za *SQL Injection*), *Wfuzz* (za *Broken Authentication*) i *Burp Suite Authz* (za *Broken Access Control*)

Kako bi se skupili svi neophodni podaci pre napada, konfiguriše se *proxy* u pregledaču i u *Burp Suite Proxy*-ju se prati filtriran saobraćaj između dve strane.

4.1 Test za *SQL Injection*

SQL Injection se sastoji od ubacivanja malicioznog SQL upita putem forme u aplikaciji. Uspešan napad rezultuje pristupom osetljivim podacima unutar baze podataka, kao i potencijalnom modifikacijom istih.

Ideja za testiranje *SQL Injection*-a kroz aplikaciju u *Python*-u pomenute tri aplikacije je sledeća: Sastaviti listu najpoznatijih malicioznih upita/fraza i za svaku od njih testirati jedno ili više unosnih polja u aplikaciji, kako bi se pokazalo da li je aplikacija ranjiva na ovaj napad, i ako jeste pokušati doći do podataka o korisnicima.

4.2 Test za *Broken Authentication*

Zaobiđena autentifikacija predstavlja ranjivost koja omogućava napadaču da prođe ili zaobiđe autentifikacione kontrole. On ima priliku da kompromituje lozinke, token sesije i da koristi implementacione mane kako bi preuzeo identitet korisnika. Do kompromitovanja sistema usled ove ranjivosti dolazi usled više faktora: korisnikovi kredencijali nisu bezbedno sačuvani ili su lako predvidivi, sesije se ne invalidiraju nakon određenog perioda…

Testiranje autentifikacione kontrole kroz test aplikacije zasniva se na sprovođenju *brute-force* napada na predviđene tri aplikacije. Polazna tačka za ovaj tip automatizovanog napada biće top 10000 najslabijih lozinki.

4.3 Test za *Broken Access Control*

Zaobiđena kontrola pristupa predstavlja ranjivost koja omogućava napadaču da pristupa neovlašćenim ili neautorizovanim funkcijama ili podacima kao što su pristup tuđem nalogu, pregled osetljivih podataka, izmena korisnikovih podataka, menjanje prava pristupa… Kontrole pristupa mogu se podeliti u tri široke kategorije: vertikalne, horizontalne i zavisne od konteksta:

- Vertikalne kontrole pristupa omogućavaju različitim vrstama korisnika pristup različitim delovima funkcionalnosti aplikacije [12]. U najjednostavnijem slučaju, ovo obično uključuje podelu između običnih korisnika i administratora
- Horizontalne kontrole pristupa omogućavaju korisnicima pristup određenom podskupu resursa iste vrste korisnika [12].
- Kontrole pristupa zavisne od konteksta osiguravaju da je pristup korisnika ograničen na ono što je dozvoljeno obzirom na trenutno stanje aplikacije [12].

Prilikom testiranja kontrole pristupa fokus će biti na eskalaciji horizontalnih privilegija i pristupu tuđim podacima.

5. REZULTATI TESTIRANJA

5.1 *SQL Injection*

Rezultati sa testiranja *SQL Injection*-a za polje pretrage *Owasp Juice Shop* aplikacije pokazuju da je moguće doći do podataka o korisnicima uz pomoć *union* naredbe, u kombinaciji sa podacima koji su dobijeni iz izuzetaka na *backend*-u. *SqlMap* alat uz odgovarajuće parametre u komandnoj liniji dolazi do tabela korisnika, ali bez mogućeg pristupa samim podacima.

Kada je u pitanju DVWA aplikacija, *Python* skripta, kreirana uz pomoć dva zahteva, s obzirom da se napad odigrava u dva različita prozora, rezultuje uspešnim napadom i dolaskom do podataka o korisnicima (imena i prezimena). *SqlMap* alat takođe dolazi do podataka o korisnicima, a u ovom slučaju i do tabela cele baze podataka, kao i korisnikovih lozinki.

Treća aplikacija, *Mega Travel Booking,* za polje pretrage smeštaja, pokazuje otpornost na maliciozne SQL upite iz manuelnog testa i za svaki zahtev vraća odgovor 422 – *Unprocessable entity.* Takođe i za *SqlMap* alat.

Kako bi se izbegle posledice *SQL Injection*-a, neophodno je implementirati adekvatan nivo zaštite uz sledeći koncept [3]:

- Validacija i sanitizacija ulaznih polja: definisanje odgovarajuće sintakse za ulazno polje forme i eskejpovanje (transformisanje) specijalnih karaketera poput ‘ / \ ” u sigurnu formu.
- Korišćenje alata/okruženja za objektno-relaciono mapiranje podataka. Obuhvata korišćenje parametrizovanih upita, pa pisanje upita za bazu u kodu nije neophodno.

5.2 *Broken Authentication*

Kod testiranja autentifikacije za *Owasp Juice Shop* i DVWA aplikaciju iskorišćena su dobijena korisnička imena iz testiranja *SQL Injection*-a, a za lozinku se koristi lista top 10000 najslabijih lozinki.

Test za *Owasp Juice Shop* aplikaciju rezultuje pronađenom lozinkom za administratorski mejl i uspešnim prijavljivanjem na sistem pod identitetom administratora. Do istih rezultata dolazi i *Wfuzz* alat.

Testiranje autentifikacije za DVWA aplikaciju nailazi na malo teži pristup, obzirom na prisustvo ANTI-CSRF tokena kod logovanja na sistem. Međutim, test pokazuje da je uz pomoć *BeautifulSoup* biblioteke moguće zaobići ovaj mehanizam zaštite i doći do pogođene lozinke iz liste, što ne važi i za *Wfuzz* alat, koji rezultuje neupešnim napadom.

Kada je u pitanju *Mega Travel Booking* aplikacija u testiranju *SQL Injection*-a nisu otkriveni podaci koji bi se mogli iskoristiti za ovaj napad, kao ni oni koji bi ukazali na *email* adrese korisnika prilikom prikupljanja informacija, pa pristup sa ovim testom nije moguć, kao ni sa testom *Wfuzz* alata.

Kako bi se izbegle evidentne posledice zaobiđene autentifikacije neophodno je implementirati adekvatan nivo zaštite uz sledeći koncept [3]:

- Lozinke ne čuvati u otvorenom tekstu, već koristiti više mehanizama. Čest oblik: *hash* funkcije primenjene na tekst lozinke kome se dodaje *salt* (niz *random* znakova) kako bi otežao *brute-force* napad.
- Implementirati provere za slabe lozinke
- Definisati minimalnu i maksimalnu dužinu i kompleksnost lozinke.
- Ograničiti ili odložiti neuspešne pokušaje logovanja. Vršiti *logging* svih neuspeha i obavestiti administratore kada su detektovane sumnjive aktivnosti koje mogu da nagoveste *brute-force* napade, a u tom slučaju ih stopirati.

5.3 *Broken Access Control*

Testiranje kontrole pristupa *Owasp Juice Shop* aplikacije svodi se na pokušaj pristupa tuđoj kupovnoj korpi. Preko ulogovanog korisnika i menjanjem parametra u URL-u test pokazuje da se može pristupiti podacima tuđe kupovne korpe. Podaci dobijeni u rezultatima testa poslužili su za još jedan tip testa – slanje žalbe u ime drugog korisnika, koji rezultuje ponovno narušenom

kontrolom pristupa i kreiranjem žalbe pod drugim identitetom. Alat *Burp Suite Authz* je takođe pokazao zaobilaženje kontrole pristupa, kreiranjem žalbe u ime drugog korisnika.

Kada je reč DVWA aplikaciji, ona ne nudi mogućnost direktnog napada na kontrolu pristupa u vidu eskalacije horizontalnih privilegija i stoga testiranje kontrole pristupa na ovu aplikaciju nije vršeno.

Za *Mega Travel Booking* aplikaciju test je vršen za kreiranje rezervacije, sa idejom umetanja tuđeg tokena prilikom slanja zahteva. Rezultati su pokazali izuzetak na *backend*-u, međutim na osnovu njih ne može se utvrditi da li ispitivana ranjivost postoji u aplikaciji. Kada je reč o *Burp Suite Authz* alatu, test je rezultovao zabranom pristupa traženim resursima (403 – *Forbidden*).

Kako bi se izbegle evidentne posledice zaobiđene kontrole pristupa neophodno je implementirati adekvatan nivo zaštite uz sledeći koncept [3]:

- Odbiti pristup za sve što nije javno dostupno (npr. neregistrovanim korisnicima).
- Podesiti da svi korisnici, programi, procesi imaju minimalno privilegija koliko je neophodno.
- Primeniti mehanizme kontrole pristupa koji će sprečiti neovlašćen pristup. Jedan od njih je RBAC model, često primenjiv.
- Vršiti *logging* svih neuspešnih kontrola pristupa i obavestiti administratore kad je to potrebno (npr. kad se isti neuspešni zahtevi ponavljaju).

6. ZAKLJUČAK

U ovom radu predstavljen je razvoj aplikacije za testiranje bezbednosti web aplikacija kroz prethodno analizirane koncepte bezbednosti web aplikacija i penetracionog testiranja istih. Testovi kroz koje se razvija aplikacija pisani su u *Python* programskom jeziku. Za testiranje su odabrane jedne od glavnih ranjivosti zastupljene u web aplikacijama danas: *SQL Injection*, *Broken Authentication*, *Broken Access Control*. Testirane su tri web aplikacije u tri scenarija.

Kada je reč o SQL *Injection*-u, testovi su pokazali uspešnu eksploataciju ranjivosti za prvu i drugu aplikaciju, a odabrani *SqlMap* alat kao paralela manuelnim testovima, to pokazuje za takođe iste aplikacije.

Testovi za autentifikaciju su izvedeni kao *brute-force* napadi i rezultuju uspešnim zaobilaženjem autentifikacije za prve dve aplikacije, dok *Wfuzz* alat namenjen *brute-force* napadu to čini samo za prvu. Treći tip testa namenjen kontroli pristupa, pokazao je da je samo prva aplikacija ranjiva za eskalaciju horizontalnih privilegija, i kroz manuelni i kroz *Burp Suite Authz* automatizovani alat.

Ono što se primećuje skoro u svim uspešnim napadima jeste nedostatak evidentiranja događaja, koje bi ukoliko je adekvatno implementirano upozorilo na sumnjive aktivosti.

Na kraju su izložene mere zaštite za testirane ranjivosti.

Može se reći da je za dati scenario i odabrane automatizovane alate, aplikacija u razvoju za testiranje, kao manuelni alat, pokazala bolju efikasnost pri otkrivanju ranjivosti. Ovakav način testiranja pruža testeru mogućnost da bolje analizira situaciju, razmišlja kako i gde bi haker mogao da izvrši napad.

Dalje istraživanje u razvoju aplikacije obuhvata proširenje trenutnih penetracionih testova, poput eskalacije vertikalnih privilegija kod zaobiđene kontrole pristupa i *Credential stuffing* napada kod zaobiđene autentifikacije. Takođe, budući rad uključuje i testiranje drugih ranjivosti poput *Cross-site scripting* napada. Preporučuje se i uvođenje drugih alata, radi poređenja rezultata sa trenutnim stanjem i dolaska do što preciznijih zaključaka.

7. LITERATURA

[1] Ahmed, S., "Secure Software Development: Identification of Security Activities and Their Integration in Software Development Lifecycle", *School of Engineering Blekinge Institute of Technology, Ronneby, Sweden*, 2007

[2] Wolf Halton, Bo Weaver, Juned Ahmed Ansari, Srinivasa Rao Kotipalli, Mohammed A. Imran, *"Penetration Testing: A Survival guide"*, Packt Publishing Ltd, 2016

[3] Owasp TOP 10 – 2017, OWASP Foundation, 2017

[4] Mansour Alharbi, "Writing a Penetration Testing Report", *SANS Institute – Information Security Reading Room*, 2010

[5] https://purplesec.us/types-penetration-testing/#Involve (pristupljeno u septembru 2020.)

[6] https://resources.infosecinstitute.com/the-types-of-penetration-testing/#gref (pristupljeno u septembru 2020.)

[7] Kassem A. Salech, *"Software Engineering"*, J. Ross Publishing, 2009

[8] Joseph Muniz, Aamir Lakhani, *"Web Penetration Testing with Kali Linux"*, Packt Publishing Ltd, 2013

[9] Rafay Baloch, *"Ethical Hacking and Penetration Testing Guide"*, CRC Press, 2017

[10] Patrick Engebretson, *"The Basics of Hacking and Penetration Testig - Ethical Hacking and Penetration Testing Made Easy"*, Elsevier, 2011

[11] Georgia Weidman, *"Penetration Testing – A Hands-On introduction to Hacking"*, William Pollock, 2014

[12] Dafydd Stuttard, Marcus Pinto, *"The Web Application Hacker's Handbook – Finding and Exploiting Security Flaw, Second Edition"*, John Wiley & Sons, Inc, 2011

Kratka biografija:

Ivana Marin rođena je u Novom Sadu 1996. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranila je 2020.god.
kontakt: ivanamarin67@gmail.com

Зборник радова Факултет техничких наука, Нови Сад

UDK: 004.9
DOI: https://doi.org/10.24867/12BE32Blagojevic

НАМЈЕНСКИ ЈЕЗИК И ОКРУЖЕЊЕ ЗА МОДЕЛОВАЊЕ НОТНОГ ЗАПИСА И ГЕНЕРИСАЊЕ СПЕЦИФИКАЦИЈА ЗА МУЗИЧКИ СОФТВЕР

A DOMAIN SPECIFIC LANGUAGE AND A FRAMEWORK FOR MUSICAL NOTATION MODELING AND GENERATING MUSIC SOFTWARE SPECIFICATION

Оливера Благојевић, *Факултет техничких наука, Нови Сад*

Област – ПРИМЈЕЊЕНЕ РАЧУНАРСКЕ НАУКЕ И ИНФОРМАТИКА

Кратак садржај – *У овом раду је описан намјенски језик и окружење за моделовање нотног записа, као и израда генератора и генерисање спецификација за музички софтвер. За креирање језика је коришћена текстуална конкретна синтакса. Осим језика, креирани су и генератори, који имају за циљ генерисање фајлова формата који одговарају софтверима за креирање и читање нотног система. Креирани намјенски језик поједностављује рад са софтверима за компоновање музике и чини писање нотних система знатно лакшим. За развој метамодела коришћено је окружење Eclipse Modeling Framework, а за дефинисање ограничења модела коришћен је OCL језик опште намјене. За креирање текстуалне конкретне синтаксе коришћен је радни оквир Xtext, а за генераторе програмски језик Xtend.*

Кључне ријечи: *музика, музичка нотација, нотни запис, намјенски језици, развој софтвера базиран на моделу*

Abstract – *The paper describes domain-specific language and framework for music score modeling, as well as generator design and generating specifications for music software. Textual concrete syntax was used for creating the language. Besides the language, generators were created, which goal is to generate files that have formats that fit sheet music creation and reading softwares. Created domain-specific language simplifies the work with softwares for music composing and makes writing sheet music much easier. Eclipse Modeling Framework was used for metamodel development as well as general purpose OCL language to define model constraints. Xtext framework was used to create textual concrete syntax, and Xtend programing language was used for generators.*

Keywords: *music, music notation, sheet music, Domain-Specific Languages, Model-Driven Software Development*

1. УВОД

Нотни запис је ручно записана или одштампана форма музичке нотације, која користећи музичке симболе означава тонове, ритмове или акорде пјесме или инструменталног музичког дјела. На енглеском језику нотни запис музике је „*sheet music*“, јер је употребом термина „*sheet*“ (срп. лист) намјера била направити разлику између писаних или штампаних музичких форми и звучних записа. Нотни запис је основни облик у коме се класична музика биљежи тако да је могу учити и изводити соло пјевачи, инструменталисти, или музички ансамбли.

Ручно писање нотних записа углавном одузима много времена, подложно је грешкама и понекад непрегледно. Да би се потешкоће приликом ручног писања нотног записа избјегле, данас на интернету постоје различити музички софтвери. Међутим, уколико корисник музичког софтвера није унапред упознат са софтвером или није довољно вјешт приликом коришћења савремених технологија, ови софтвери могу бити веома компликовани за коришћење и нејасни.

Истраживањем и анализом музичких софтвера је утврђено да сви музички софтвери омогућавају увоз и извоз фајлова који садрже записану музичку нотацију у одређеном, изабраном, формату. Могуће је покренути музички софтвер, учитати фајл формата који одређени софтвер подржава и добити исписан нотни систем, без коришћења софтвера за исписивање нотног записа. Софтвер се након увоза фајла може искористити за додатне опције и за репродукцију композиције, а измјењени фајл се потом поново може сачувати у жељеном формату. На овај начин би се уштедило вријеме за исписивање музичких нотација, а предности музичких софтвера би се и даље могле искористити. Циљ овог рада је проналажење начина за генерисање фајлова који садрже жељени нотни запис и који могу бити увезени у музичке софтвере и тиме рјешење могућих проблема приликом употребе музичких софтвера. Да би се ово остварило, креирани су намјенски језик за моделовање нотног записа, назван *MusicDSL*, и њему одговарајући генератори, који у себи садрже спецификације за музички софтвер.

Поред уводног поглавља и закључка, овај рад садржи још шест поглавља. Поглавље *Преглед постојећег стања у области* представља преглед доступних музичких софтвера и преглед формата које фајлови ових програма користе, а које *MusicDSL* генерише. У

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Милан Челиковић, доцент.

оквиру поглавља *Теоријске основе намјенског језика и трансформација модела* су дати основни појмови и концепти за спецификацију намјенских језика и теоријске основе трансформација модела. Поглавље *Опис коришћених технологија* даје опис технологија које су коришћене за креирање *MusicDSL* намјенског језика и генератора. У *Архитектури система* су приказани дијелови описаног система. У поглављу *Намјенски језик за моделовање нотног система* је дат детаљан опис концепата и креирања *MusicDSL* намјенског језика. Поглавље *Генерисање фајлова* представља изглед генератора, опис тока генерисања и коначни резултат.

2. ПРЕГЛЕД ПОСТОЈЕЋЕГ СТАЊА У ОБЛАСТИ

Брзи развој десктоп рачунара 1980-их година је допринио стварању много софтвера за писање нотних записа. То је тада био много јефтинији начин креирања нотних записа и дијелова за оркестралну музику, који су користили млади композитори, музички педагози и студенти композиције. Нотни записи у тадашњим музичким софтверима су били читљиви али нису изгледали професионално и због тога их комерцијални издавачи музике нису користили. Током 1990-их година већина ових софтвера је изашла из употребе, јер су се појавили нови, лакши за коришћење и бољег квалитета записа. Развојем *Windows* оперативног система појављивало се све више музичких софтвера, са опцијама увоза, измјене и штампања музичке нотације, који могу да испишу једноставну композицију, комад за одређен инструмент, или комплексну оркестралну музику, високог квалитета записа. Скоро сви музички софтвери користе своје формате фајлова за чување креираних композиција. Да би се креиране композиције могле користити у различитим софтверима, већина музичких софтвера дозвољава увоз и извоз фајлова у стандардним форматима за размјену фајлова.

2.1. Формати за размјену дигиталних нотних записа

Као што је претходно наведено, сви софтвери за креирање и читање нотих записа имају опцију увоза и извоза музичких нотација у фајлове одређених формата, који се касније могу користити у неком другом софтверу, чиме је обезбијеђена једноставна размјена дигиталних нотних записа. Најчешће омогућен формат дигиталног нотног записа је *MusicXML*. Скоро сви софтвери подржавају увоз овог формата, али не и извоз. Неки софтвери дозвољавају чување нотних записа само у *XML* формату фајла (*Noteflight*) или *MSCX* (*MuseScore*).

Extensible Markup Language (*XML*) или проширив мета језик за означавање текстуалних докумената, је стандардни скуп правила за дефинисање формата података у електронској форми прописан од стране *World Wide Web* групе (*W3C*). Слиједећи правила *XML* стандарда корисници дефинишу сопствене формате података, које могу користити за њихово складиштење, обраду и размјену. *XML* документи имају хијерархијску структуру и концептуално се могу тумачити као структура стабла, која се назива *XML* стабло. *XML* документи морају да садрже основни елемент („родитељ“ свих осталих елемената). Сви елементи у *XML* документу могу да садрже поделементе, текст и атрибуте. Стабло представљено *XML* документом почиње од основног, коријенског, елемента и грана се до најнижег нивоа елемената [1].

MusicXML је стандардни отворени формат за размјену дигиталних нотних записа, који се може бесплатно користити уз *W3C* спецификацију. Дизајниран је за дијељење нотних записа између апликација и за архивирање таквих фајлова за будућу употребу. Од датума издавања овај формат подржава преко 200 апликација [2].

MSCX је формат дигиталног нотног записа креиран за *MuseScore.* Уведен је тек у верзију *MuseScore* 0.9.5, јер је прије тога било тешко ријешити конфликт са повезивањем фајла у Windows-у. *MSCX* фајлови чувају некомпресовану верзију нотних записа. Компресована верзија се чува у *MSCZ* фајловима, што је стандардни формат за *MuseScore* композиције. *MSCX* фајлови могу бити преведени у двадесет језика у *MuseScore*-у. [3] Због тога што је *MSCX* некомпресовани текстуални формат, може једноставно ручно бити уређиван у било ком текстуалном едитору.

2.2. Преглед доступних софтвера за креирање нотних записа

Данас су софтвери за креирање нотних записа потпуно доступни и постали су неопходни скоро као што су софтвери за унос и обраду текста. На тржишту их има много, а овдје ће бити подијељени у три групе: *offline* софтвери, *online* софтвери и апликације. Биће наведени они који су бесплатни и који дозвољавају увоз и извоз фајлова претходно поменутих формата.

Међу *offline* софтверима најпознатији су: *MuseScore*, *ImproVisor*, *Sibelius*, *Denemo*, *Anvil Studio* и *IBOS MusicXml Reader*.

У групи *online* софтвера најпознатији су: *Noteflight* и *Flat*. Од апликација тренутно су доступне: *forScore* и *Orpheus Sheet Music Pro.*

3. ТЕОРИЈСКЕ ОСНОВЕ НАМЈЕНСКОГ ЈЕЗИКА И ТРАНСФОРМАЦИЈЕ МОДЕЛА

Језик представљен у раду је намјенски, а имплементиран је захваљујући трансформацијама из модела у текст. Како би поступак израде рјешења за проблем описан у раду био јаснији, а самим тим и циљ рада, потребно је дати теоријске основе ових појмова.

Језици специфични за домен (енг. *Domain-Specific Modeling Languages, DSL*) су дизајнирани посебно за одређен домен, контекст, или компанију, како би олакшали задатке људима који треба да опишу ствари у том домену. Ако је језик усмјерен на моделовање назива се још и доменски специфичним језиком за моделовање (енг. *Domain-Specific Modeling Language, DSML*). *DSL* се користио у рачунарској науци много прије него што је сам акроним настао. Примјери оваквих језика су: *HTML* језик за израду *web* страница, *Logo* за једноставно цртање, *VHDL* за описивање хардвера, *Mathematica* и *MatLab* за математику, *SQL* за приступ базама

података, итд. [4] Језик специфичан за домен је намијењен за употребу у оквиру конкретно дефинисаног домена. Сваки намјенски језик је дефинисан кроз три кључна елемента:

- Апстрактна синтакса описује структуру језика и начин на који се различити примарни елементи могу међусобно комбиновати, независно од било које конкретне репрезентације или кодирања.
- Конкретна синтакса описује одређену репрезентацију језика моделовања, укључујући кодирање и/или проблеме визуелне презентације. Конкретна синтакса може бити: текстуална – подаци се кодирају низом карактера, као код програмских језика и графичка конкретна синтакса – подаци се кодирају пропсторним распоредом графичких и текстуалних елемената.
- Семантика – описује значење елемената дефинисаних у језику и значење различитих начина комбиновања ових елемената.

Модели нису и изоловани ни статички ентитети. Као дио процеса инжењеринга заснованог на моделу, модели су: обједињени, поравнати, преправљени и пречишћени. Све ове операције над моделима су имплементиране као трансформације модела. Могу бити трансформације из модела у модел (енг. *Model-To-Model*, *M2M*) или трансформације из модела у текст (енг. *Model-To-Text*, *M2T*). У M2M трансформацијама улазни и излазни параметри трансформације су модели, док је у M2T трансформацијама улазни параметар модел, а излазни текст.

4. ОПИС КОРИШЋЕНИХ ТЕХНОЛОГИЈА

Изабрани начин креирања намјенског језика за моделовање нотног система је креирање намјенске текстуалне синтаксе на основу метамодела. Помоћу *M2T* трансформација се аутоматизује генерисање музичких фајлова креирањем шаблона. У наставку овог поглавља су описане технологије којима је постигнуто наведено.

За креирање метамодела, његове динамичке инстанце и генерисање кôда из модела коришћен је *Eclipse*-ов радни оквир за моделовање (енг. *Eclipse Modeling Framework*, *EMF*). *EMF* нуди могућности генерисања кôда за креирање алата и других апликација базираних на структурисаном моделу података. Из спецификације модела описане у *XMI* формату, *EMF* нуди и алате и *runtime* подршку за израду скупа *Java* класа из модела, заједно са скупом додатних класа које омогућавају читање и уређивање модела на основу команде, и основни едитор.

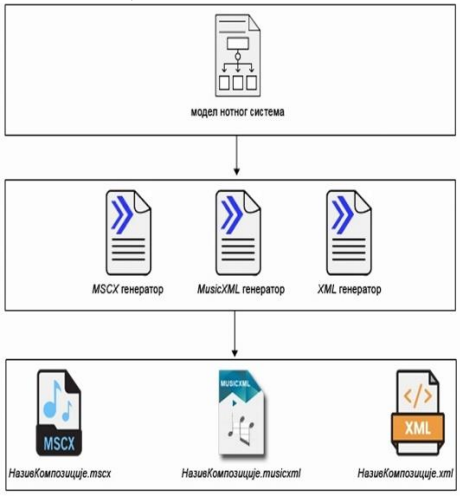
У метамоделу рађеном у EMF *Ecore*-у није било могуће представити сва ограничења намјенског језика, нека су требала бити додатно дефинисана, и за то је коришћен језик за ограничења објеката (енг. *Object Constraint Language*). Правила за ограничења су писана на нивоу апстрактне синтаксе, како би сама имплементација ограничења била једноставнија.

Текстуална граматика намјенског језика је аутоматски генерисана, а затим прилагођена и генерисани су њени артефакти у *Xtext* радном оквиру. *Xtext* аутоматизује креирање текстуалних едитора намјенских језика у виду *Eclipse plugin*-а. Коришћена је опција креирања пројекта на основу постојећег модела (eng. *Xtext Project from Existing Ecore Models*), како би се могле користити инстанце креираног *Ecore* метамодела.

За реализацију генератора који креирају .xml, musicxml и .mscx фајлове коришћен је програмски језик *Xtend*. *Xtend* је програмски језик вишег нивоа генералне намјене за *Java* виртуелну машину. Синтактички и семантички има коријене у *Java* програмском језику али се фокусира на сажетију синтаксу и неке додатне функционалности као што су: детекција типа података, методе проширивања (енг. *extension method*) и *operator overloading* (различити оператори имају различите имплементације, у зависности од њихових аргумената).

5. АРХИТЕКТУРА СИСТЕМА

Први корак израде софтверског рјешења описаног у раду је дефинисање архитектуре жељеног система, а затим креирање система који одговара дефинисаној архитектури. Почетна ставка архитектуре система је модел нотног система, који се састоји од свих елемената нотног система који су потребни за генерисање жељених фајлова. На основу модела је креиран и намјенски језик *MusicDsl*, а потом су направљени генератори помоћу којих добијамо фајлове жељених формата. Архитектура система је приказана на слици 1.



Слика 1: Архитектура система

Систем се састоји од: модела нотног система, на основу кога су креирани генератори (*MSCX*, *MusicXML* и *XML*), који генеришу фајлове истоимених формата.

6. НАМЈЕНСКИ ЈЕЗИК ЗА МОДЕЛОВАЊЕ НОТНОГ ЗАПИСА

Коријенски концепт апстрактне синтаксе је композиција (*Composition*), гдје корисник касније уноси уопштене информације о композицији коју креира (назив, аутор, инструмент за који је

предвиђена, итд). Композиција у себи садржи линијске системе (*Staff*), може имати највише два и то је ограничење које је било могуће намјестити у *EMF*-у. Сваки линијски систем садржи кључ (*Clef*), тактну вриједност (*TimeSignature*) и тоналитет (*Key*). Код неких софтвера за креирање нотног система (нпр. *Flat.io*) је могуће само једном унијети тактну вриједност и тоналитет и то се не мијења, остаје исто за оба кључа.

Код других софтвера за креирање нотног система за сваки линијски систем се те вриједности поново уписују (нпр. *Sibelius*), тако да су приликом генерисања шаблони прилагођавани томе. Осим ова три елемента, које линијски систем мора имати, он такође садржи и апстрактну класу са својим осталим елементима (*StaffElement*) коју насљеђују такт (*Measure*), пауза (*Rest*), понављање (*Repetition*) и нота (*Note*). Типови ових елемената су наведени у енумерацији. Нота сама за себе садржи украсе, своју артикулацију и везе, који се налазе унутар концепта нотација (*Notation*).

Текстуална конкретна синтакса је креирана тако што је прво *Xtext* генерисао граматику на основу коријенске класе *Composition*, а затим је граматика прилагођена жељеном изгледу језика. Приликом прилагођавања, промијењено је сљедеће:

- већина витичастих заграда је замијењена угластим;
- почетак такта се означава именом и редним бројем такта након којег слиједи ознака *begin*, а крај се означава ознаком *end*;
- вриједности енумерација се уносе без испред унијетог назива;
- називи веза су уклоњени, јер нема потребе да их корисник уноси;
- промијењене су неке кључне ријечи, највише када је концепт *Note* у питању.

```
Composition "Yesterday" [
          author: "John Lennon and Paul McCartney"
          instrument: "Piano"
          tonality: "F major"
          soundTempo: 80
          Staff 1 [
                    clef: violin
                    key: -1
                    time: beat 4, beatType 4
                    Measure 1 begin
                    note G :
                              type eighth
                              duration 1
                              octave 4
                              voice 1
                              lyrics "Yes"
end,
```

Листинг 1: Примјер почетног дијела модела нотног система композиције Yesterday

7. ГЕНЕРИСАЊЕ ФАЈЛОВА

За имплементацију генератора потребно је било направити *Xtend* шаблоне који одговарају спецификацији музичких софтвера и структури фајлова формата за размјену дигиталне музичке нотације. То значи да су осим шаблона који одговарају спецификацији музичких софтвера генерисане и функције за претраживање и препознавање линијских система, тактова, типова музичких елемената, итд.

Из *MusicDslGenerator* генератора се позивају и покрећу *MusicXML*, *XML* и *MSCX* генератори. Генерисан је аутоматски приликом генерисања *Xtext* артефакта и касније прилагођен. Основни елемент по коме се филтрира и врши претрага осталих елемената је *Composition*. Елемент *Composition* је коријенски елемент и због тога једини из кога се може приступити свим осталим елементима, јер су унутар њега.

8. ЗАКЉУЧАК

У раду је описан намјенски језик *MusicDsl* који омогућава креирање фајлова који се могу учитати у музичке софтвере, без претходног програмерског предзнања корисника, довољно је само да познаје синтаксу језика. Језик посједује концепте којима је детаљно описан цијели нотни систем, тј. сви елементи који су неопходни за писање једне композиције. Из њега су даље креирани генератори, који након корисниковог уноса нотног записа путем намјенског језика генеришу фајлове жељених формата.

Намјенски језик *MusicDsl* би могао бити још унапређен, у зависности од тога у ком смјеру ће се мијењати музички софтвери. Иако ови софтвери упозоравају ако је у учитаном фајлу превише или премало нота у једном такту у односу на тактну вриједност, у будућности би се могла додати валидација унутар конкретне синтаксе, тако да корисник у старту не би могао унијети погрешан број тактних елемената. Ово би било веома корисно, али је тешко реализовати јер је подложно грешкама уколико се не наведу све комбинације. Апстрактна синтакса би вјероватно могла бити још проширена, уколико софтвери почну додавати још неке музичке елементе у своје опције. Ако корисник не уноси различите ноте за другу врсту кључа, онда би нека врста преводиоца из кључа у кључ такође била добра идеја за унапређење језика. Када би се појавили нови софтвери на тржишту, са новим форматима представљања музичких елемената у фајлу, израда још једног генератора би била неопходна.

ЛИТЕРАТУРА

[1] *XML, Wikipedia* - https://en.wikipedia.org/wiki/

[2] *MusicXML*, документација - https://www.w3.org/2017/12/musicxml31

[3] *MuseScore*, документација - https://musescore.org/en/handbook

[4] M. Brambilla, J. Cabot, and M. Wimmer, *Model-driven software engineering in practice*, Second edition. San Rafael, Calif.: Morgan & Claypool Publishers, 2017.

Кратка биографија:

Оливера Благојевић рођена је 1994. године у Добоју, у Босни и Херцеговини. Дипломски рад из области информационих технологија одбранила је 2017. године на Слобомир П Универзитету. Мастер студије на Факултету техничких наука је уписала 2019. године.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE33Suvajdzic

ЕВАЛУАЦИЈА СОФТВЕРСКИХ АЛАТА ЗА ОБРАДУ И ПРЕПОЗНАВАЊЕ ГОВОРА

EVALUATION OF SOFTWARE TOOLS FOR SPEECH PROCESSING AND RECOGNITION

Милан Сувајџић, *Факултет техничких наука, Нови Сад*

Област – Рачунарство и аутоматика

Кратак садржај – *У овом раду представљене су неке од грана вештачке интелигенције, као што су машинско учење, неуронске мреже и обрада природног језика, које су повезане с областшу препознавања говора. Дат је детаљан опис структуре и начина функционисања система за препознавање говора. Представљена је SpeechRecognition библиотека преко које је извршена примена три тренутно најпопуларнија сервиса за препознавање говора. Урађена је њихова детаљна анализа и евалуација по задатим критеријумима, на основу које су изведени закључци о томе колико је сваки од њих погодан за коришћење.*

Кључне речи: *Софтверски алатал, обрада говора, препознавање говора*

Abstract – *This paper presents some of the branches of artificial intelligence, such as machine learning, neural networks and natural language processing, which are related to the field of speech recognition. A detailed description of the structure and functioning of the speech recognition system is given. The SpeechRecognition library was presented, through which the application of 3 currently most popular speech recognition services was performed. Their detailed analysis and evaluation according to the given criteria was done, on the basis of which conclusions were made about how suitable each of them is for use.*

Keywords: *Software tool, speech processing, speech recognition*

1 УВОД

Човек, као друштвено биће, живи и ради у заједници, у оквиру које комуницира са другим људима. Језик је човеково најважније средство комуникације, а говор примарни медиј. Говорна интеракција својствена је људским бићима, те се по томе разликује од осталих бића на земљи.

Како није увек у ситуацији да директно оствари вербални контакт са другим људима, човек се у данашње време служи средствима модерне технологије (рачунаром, мобилним телефоном и многим другим апаратима). Интеракција коју човек остварује посредно, путем тастатуре, миша, управљачких конзола и других приступа, наилази на разне потешкоће, као што су: губитак информација, грешке и време које се троши на пренос и интерпретацију. Стога се неизбежно намеће потреба имплементације говорне интеракције и у случају интеракције човека - рачунар.

Рад обухвата говорне технологије које се данас најчешће користе и претварање говора у аналогни и дигитални таласни облик разумљив рачунарима. Препознавање говора или конверзија говора у текст укључује хватање и дигитализацију звучних таласа, претварање у основне језичне јединице или фонеме, изградњу речи од фонема, и контекстуалне анализе речи.

Препознавање говора је способност рачунара да препозна опште, природне изразе од стране различитих корисника.

Нагласак је на моделирању говорних јединица и граматике на основи скривеног Марковљевог модела, као и на примени вештачких неуронских мрежа у процесу конверзије говора.

Препознавање говора омогућава унос улазних података користећи се гласом. Примене су небројене, од медицине, авио индустрије до роботике итд.

Иако постоји још много простора за унапређења, данашњи системи дају задовољавајуће резултате, те потврђују да се ова технологија развија у правом смеру.

2 ОСНОВНИ ТЕОРИЈСКИ КОНЦЕПТИ

Ово поглавље даје преглед основних теоријских појмова неопходних за разумевање методологије коришћене у експерименту, као и метода за евалуацију добијених резултата.

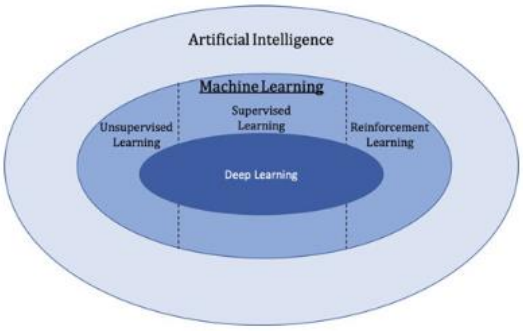
2.1 Машинско учење

Машинско учење (eng. ***Machine Learning***) је поткатегорија вештачке интелигенције (eng. ***Artificial Intelligence*** – AI), која представља способност дигиталног рачунара или рачунара контролисаног робота да извршава задатке уобичајено повезане са интелигентним бићима. Машинско учење представља проучавање рачунарских алгоритама који се аутоматски побољшавају кроз искуство. Категорије машинског учења су: **учење под надзором** (*Supervised Learning*) - ослања се на учење из скупа података са ознакама, **учење без надзора** (*Unsupervised Learning*) - одређује кластере, на основу података на без ознака и **учење ојачавањем** (*Reinforcement Learning*) - фокусира се на максимизирање награде за дату радњу или низ предузетих радњи [1].

НАПОМЕНА:
Овај рад је проистекао из мастер рада чији ментор је био др Александар Купусинац, ванр. проф.

2.2 Дубоко учење

Дубоко учење (eng. ***Deep Learning***) је поље машинског учења које се бави алгоритмима инспирисаним структуром и функцијом мозга званом вештачке неуронске мреже. Подручје дубоког учења покрива све три категорије машинског учења, а заједно са машинском учењем упада у област вештачке интелигенције. (Слика 1)



Слика 1. *Однос дубоког учења, машинског учења и вештачке интелигенције [1]*

2.2.1 Вештачке неуронске мреже

Вештачка неуронска мрежа (eng. ***Artificial Neuron Network*** - ANN) је систем за обраду информација састављен од огромниг броја међусобно повезаних процесних чворова званих неурони. Неурони се групишу у слојеве, а веза између два неурона се назива ивица. Неуронске мреже које се користе у дубоком учењу имају више од три слоја и називају се дубоке неуронске мреже (*Deep Neural Network* – DNN). Постоје два типа неуронских мрежа: неуронске мреже без повратног преноса (eng. *Feedforward Neural Network*) и неуронске мреже са повратним преносом (eng. *Feedback Neural Network* или *Recurrent Neural Network* – RNN).

2.3 Обрада природног језика

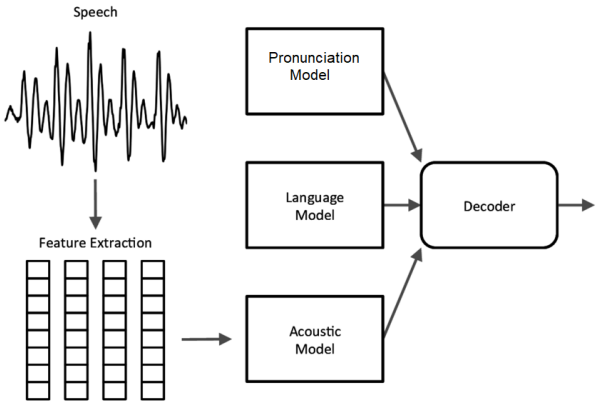
Обрада природног језика (eng. ***Natural Language Processing*** - NLP) је грана вештачке интелигенције која помаже рачунарима да разумеју, тумаче и манипулишу људским језиком. NLP црпи из многих дисциплина, укључујући рачунарску науку и рачунарску лингвистику, у својој потрази за попуњавањем јаза између људске комуникације и разумевања рачунара [1].

2.4 Препознавање говора

Препознавање говора (eng. ***Speech Recognition***) је интердисциплинарно потпоље рачунарске науке и рачунарске лингвистике које развија методологије и технологије које омогућавају препознавање и превођење говорног језика у текст помоћу рачунара. Такође је познато као аутоматско препознавање говора (eng. ***Automatic Speech Recognition*** - ASR), рачунарско препознавање говора или говор у текст (eng. *Speech-to-Text*).

2.5 Структура ASR система

Дизајн и развој система за препознавање говора зависе од различитих компоненти, као што су представљање и пред-обрада говора, класе говора, различите врсте техника издвајања карактеристика, коришћени класификатори, база података и перформансе система. Слика 2 репрезентује типичну структуру аутоматског система за препознавање говора.



Слика 2. *Структура ASR система [2]*

Улаз у систем за аутоматско препознавање говора је говорни **аудио сигнал**. Аудио сигнали су електронски прикази звучних таласа, настали као резултира преноса енергије са једног молекула на други. Постоје два типа аудио сигнала: аналогни (континуирани приказ сигнала током одређеног временског периода) и дигитални (дискретни приказ сигнала током одређеног временског периода).

Издвајање карактеристика звука је прва фаза ASR система. Главни циљ екстракције карактеристика у ASR систему је пронаћи неки скуп вектора или карактеристика које могу дати тачан приказ улазног аудио сигнала.

Акустични модел је следећа фаза, где се помоћу издвојених вектора карактеристика проналази статистички модел. Најмањи разликовни или препознатљиви део језика назива се фонема. Однос између аудио сигнала и фонема представљен је **акустичним моделом**. Креира се база података говора и на њу се примењују алгоритми обуке како би се створио статистички приказ сваке фонеме, који се називају скривени Марков модел (eng. ***Hidden Markov Model*** – HMM).

Модел изговора је једини модел у ASR систему који не пролази кроз фазу учења. Он помаже у организовању фонема да би се створила изговорена смислена реч.

Језички модел, с друге стране, помаже у организовању речи да би се створила изговорена смислена реченица. То је у основи статистички модел који показује колико је вероватно да се низ речи може појавити заједно.

Резултати свих претходно описаних модела, улазе у **декодер**. Декодер тада идентификује најтачнију транскрипцију изговорене реченице.

Из перспективе декодера, то је главни проблем претраживања. Комбинујући акустични модел, модел изговора и језички модел у декодеру се формира графикон претраживања [2].

3 ОПИС РЕШАВАНОГ ПРОБЛЕМА

Системи за препознавање говора имају своје апликативне програмске интерфејсе, путем којих се користе у оквиру других већих система. Већина апликативних програмских интерфејса за препознавање говора, доступна је на интернету и бесплатна један пробни период, а касније се наплаћује у зависности од количине потреба корисника.

3.1 Мотивација

Да би се разумео процес препознавања говора, потребно је извршити његову детаљну анализу у склопу система за препознавање говора. Тек након створене комплетне слике процеса препознавања говора, долази се до потребе евалуације система који врше тај процес. Мотивација овог рада јесте да на основу евалуације различитих система за препознавање говора, примењених преко својих апликативних програмских интерфејса, прикаже предности и мане сваког од њих по задатим критеријумима.

3.2 Преглед стања у области

Од 1920-их, систем препознавања говора се постепено развијао. Прва комерцијална играчка која се може сматрати првом машином за препознавање говора појавила се 1922. године по називом "*Radio Rex*" [1].
IBM је развио и демнострирао *Shoebox* шездесетих година прошлог века, што је био пионирски рад у данашњем систему препознавања говора. Овај јединствени уређај могао је да препозна 16 изолованих речи [1].
Седамдесетих година прошлог измишљен *HARPY* напредни систем препознавања говора заснован на вештачкој интелигенцији. Могао је препознати речник који садржи 1000 речи [1].
Статистички модели постали су популарни од периода 1980-их, а најчешће коришћени модел у вези с тим био је скривени Марков модел. Величина речника је постајала све већа и користило се око 10 000 речника [1].
Током 2000-их наставља се фокус на унапређење машинског учења. Мреже дубоког веровања примењене су на препознавање телефона, постижући врхунске перформансе на ТИМИТ корпусу. Касније је представљен хибридни модел дубоких неуронских мрежа са проширењем скривеног Марковог модела.
Активно истраживање препознавања говора довело је до доласка различитих јавних и лиценцираних софтвера алата. Најпопуларнији јавни софтверски алати су Sphinx, HTK, RWTH и Kaldi. Најпопуларнији лиценцирани софтверски алати су Siri, IBM Watson, Google Speech API, Bing Speech API, Amazon Alexa, Cortana [3].

4 ОПИС РЕШЕЊА ПРОБЛЕМА

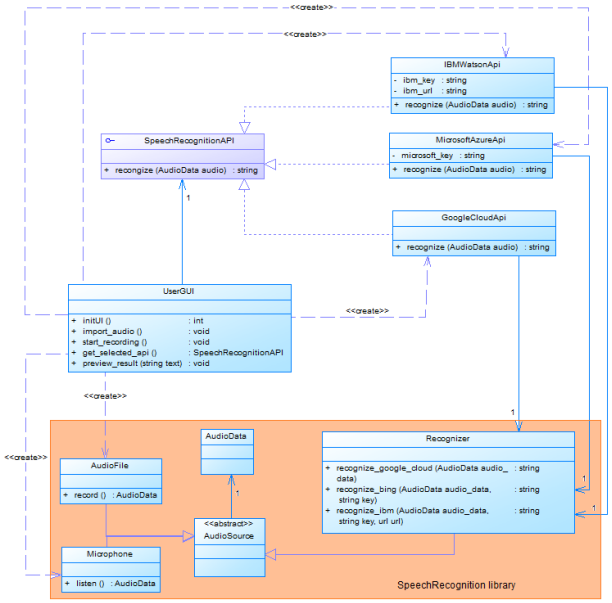
Детаљан опис решења проблема дат је кроз експериментелан пројекат у ком се примењују различити системи за препознавање говора, преко одговарајућих API-ја, захваљујући *SpeechRecognition* библиотеци. Спецификација пројекта дефинисана је кроз две фазе: анализа захтева и спецификација дизајна.

4.1 Анализа захтева

Главна сврха система за препознавање говора се дефинише приликом формулисања захтева у процесу анализе захтева.
Функционални захтеви апликације која представља експериментални пројекат у овом раду су: одабир API-ја за препознавање говора (*Google Cloud Speech to Text*, *Microsoft Azure Speech to Text* или *IBM Watson Speech to Text*), снимање говора путем микрофона, отпремање аудио датотеке, позивање API-ја за препознавање говора и приказивање резултата. Док функционални захтеви дефинишу шта систем ради, нефункционални захтеви одређују како систем то треба да ради. У нефункционалне захтеве спадају: латенција, тачност, прецизност, језичка подршка и цена.

4.2 Спецификација дизајна

Објектни дизајн дефинише решење које би требало да премости јаз између модела анализе и хардверске / софтверске платформе дефинисане током дизајнирања система. То укључује прецизан опис интерфејса објекта и подсистема. Слика 3 детаљно приказује дијаграм класа (eng. ***Class Diagram***) апликације за препознавање говора која се анализира у овом раду.



Слика 3. *Дијаграм класа експерименталне апликације*

4.3 SpeechRecognition библиотека

Једна од најпопуларнијих *Python* библиотека за препознавање говора јесте ***SpeechRecogniton*** библиотека, која се налази на PyPI репозиторијуму. Она олакшава преузимање аудио улаза из ког се врши препознавање говора. Препознавање говора захтева унос аудио звука, а *SpeechRecogniton* олакшава његово преузимање. Уместо прављења скрипте за приступ микрофонима и обраду аудио датотека од нуле, *SpeechRecogniton* обавља те процесе за само неколико минута [4].

4.3.1 Recognizer класа

Све функционалности у *SpeechRecognition* библиотеци врше се преко ***Recognizer*** класе.

Инстанца *Recognizer* класе долази са разним поставкама и функцијама за препознавање говора из аудио извора (аудио датотека или говора преко микрофона) [4].
Свака *Recognizer* инстанца има седам метода за препознавање говора из аудио извора помоћу различитих API-ја [4]:

- recognize_google(): Google Web Speech API
- recognize_google_cloud(): Google Cloud Speech
- recognize_bing(): Microsoft Bing Speech
- recognize_ibm(): IBM Speech to Text
- recognize_houndify(): Houndify by SoundHound
- recognize_sphinx(): CMU Sphinx – захтева инсталацију PocketSphinx
- recognize_wit(): Wit.ai

4.4 Имплементација

Како се сви наведени API-ји, који се користе у експерименталној апликацији, налазе у "облаку", било је потребно имплементирати и аутентификацију за сваки од њих. Приликом аутентификације добијају се додатни параметри за позивање API-ја.
Препознавање говора преко *Google CloudApi*-ја обавља готова ***recognize_google_cloud*** метода класе *Recognizer*. Препознавање говора преко *MicrosofAzureApi*-ја обавља готова ***recognize_bing*** метода класе *Recognizer*. Препознавање говора у оквиру *IBMWatsonApi*-ја обавља готова ***recognize_ibm*** метода класе *Recognizer*.

4.5 Резултати тестирања

Како се подразумева да функционални захтеви система за препознавање говора увек буду испуњени, у оквиру евалуације примењених API-ја у овом експерименталном пројекту, акценат ће бити више дат на нефункционалним захтевимаНа основу њих су формирани критеријуми евалуације, по којима је вршено тестирање.
Узорак говора је добијен из аудио књиге „Emma" од Jane Austin и дат је као улаз у апликацију. Узето је 19. поглавље књиге чији снимак траје 9 минута 7 секунди. Узорак говора снимљен је у тихом окружењу са микрофоном.
Што се тиче перформанси сервиса, Google Cloud STT и Microsoft Azure STT сервиси прелазе невероватних 97% тачности препознавања речи, док IBM Watson STT сервис има нешто мању тачност од 96.9% што је и даље задивљујућ резултат.
Време које је потребно за конверзију говора у текст је најмање код IBM Watson STT сервиса, док је Google Cloud STT бржи од Microsoft Azure STT сервиса за око 1 секунду.
Највећу предност је Google Cloud сервис за препознавање говора остварио у области језичке подршке где подржава скоро дупло више језика од Microsoft Azure STT сервиса. IBM Watson STT сервис подржава неупоредиво мање језика од претходна два.
Различите компаније система за препознавање говора наплаћују различито своје услуге. Ако се цена услуге посматра у форми $/минуту, долази се до закључка да Microsoft компанија има најповољнију цену својих услуга (0.016 $/минуту), док је Google најскупљи (0.036 $/минуту).

5. ЗАКЉУЧАК

У овом раду представљено је широко теоријско сазнање у области препознавања говора, које спада у једно од најактуелнијих области данас. Циљ рада је био да се упореде три најпопуларнија *cloud* сервиса за препознавање говора, развијаних од стране светских гигант компанија (Google, Microsoft и IBM).
Практично решење је била апликација у *Python* програмском језику, унутар које су имплементирани позиви ових сервиса преко њихових API-ја.
Објашњена је примена *SpeechRecognition* библиотеке преко које су извршени API позиви ових сервиса.
На основу резултата евалуације примењених сервиса, долази се до закључка да се не може посебно издвојити ни један сервис за препознавање говора који је најбољи по свим критеријумима.
Све је већа потражња за интерактивним говорним системима који укључују дијалог између људи и рачунара. У будућности ће бити потребно већа природност како у језику које човек користи, тако и у одговорима које генерише систем. Таква природност је вероватно остварива само ако рачунар има добар модел интеракције. Потешкоће се обично сматрају општим проблемима вештачке интелигенције. Иако су достигнућа на овом подручју импресивна, системима за препознавање говора се постављају додатни изазови.

6. ЛИТЕРАТУРА

[1] U. Kamath, J. Liu, J. Whitaker, *Deep Learning for NLP and Speech Recognition*. Switzerland: Springer International Publishing. 2019. [Преузето: 1.09.2020]

[2] S. Sen, A. Dutta,N. Dey, *Audio Processing and Speech Recognition*. Singapore: Springer. 2019. [Преузето: 1.09.2020]

[3] J. Holmes, W. Holmes, *Speech Syntesis and Recognition*. New York: Taylor & Francis. 2003. [Преузето: 1.09.2020]

[4] "The Ultimate Guide To Speech Recognition With Python" https://realpython.com/python-speech-recognition/#recap-and-additional-resources. Приступљено октобра 2020

Кратка биографија:

Милан Сувајџић рођен је у Врбасу 1995. године. Мастер рад на факултету техничких наука из области Електротехнике и рачунарства – Рачунарство и аутоматика одбранио је 2020. год.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE34Gula

EKSPERTSKI SISTEMI I NJIHOVA PRIMENA U CHATBOT APLIKACIJAMA

EXPERT SYSTEMS AND THEIR APPLIANCE IN CHATBOT APPLICATIONS

Mario Gula, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *U ovom radu biće opisani ekspertski sistemi i njihova primena u četbot aplikacijama. Četbot aplikacija ume da odgovara na pitanja vezana samo za određenu temu a u ovom slučaju to je košarka. Aplikacija je napisana u Python programskom jeziku. Pored konkretne implementacije jednog ekspertskog sistema rad diskutuje i o računarskoj tehnici i njenom uticaju na čoveka. Predstavljaju se i različiti četbot agenti kao i njihove prednosti i mane. Takođe se prikazuje i jedan od najaktuelnijih četbotova današnjice.*

Ključne reči: *Ekspertski sistemi, sistemi bazirani na znanju, četbot*

Abstract – *This paper will describe rule-based systems and their application in chatbot applications. The chatbot application is capable to answer only questions from a specific topic and the selected topic is basketball. The application is implemented in Python programming language. Besides the concrete implementation of one expert system this paper also discusses computer technology and its impact on humans. Moreover, the paper shows different types of chatbot agents and their advanages and flaws. It also shows one state-of-the-art chatbot system.*

Keywords: *Expert systems, rule-based systems, chatbot*

1. UVOD

U ovom radu se prolazi kroz proces izgradnje ekspertskog sistema, šta je sve neophodno, ko sve učestvuje u izgradnji, koji su benefiti itd. Objašnjava se šta je četbot i čemu on služi, od čega se sastoji, različiti načini da se implementira četbot sistem kao i različiti agenti koji se mogu koristiti prilikom implementacije. Kao primer najsavremenijeg četbota današnjice uzet je Google-ov Meena četbot koji je se pojavio početkom 2020. godine, i objašnjen je način njegove izrade.Za kraj rada je ostavljena implementacija jednog ekspertskog sistema u četbot aplikaciji. Projekat je rađen u Python programskom jeziku uz pomoć raznih biblioteka. Rešavao se problem četbot aplikacije koja može da odgovara na pitanja vezana za sport košarku. Rešenje je implementirano uz oslonac na neke već postojeće radove. Podaci koji su korišteni za razvoj aplikacije su dobavljeni sa Wikipedia stranice o košarci.

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

Nakon toga, prikazana je implementacija četbot aplikacije u vidu koda i u nekoliko poglavlja je objašnjeno čemu svaki deo tog koga čini. Za kraj je ostavljena analiza dobijenih rezultata i predlog mogućih poboljšanja.

2. EKSPERTSKI SISTEMI

Ekspertski sistem je računarski program koji obezbeđuje rešenja ekspertskog nivoa za važne probleme i takođe je:

1. Heuristističan – rezonuje sa osuđivačkim znanjem kao i sa formalnim znanjem za date probleme
2. Transparentan – obezbeđuje objašnjenje za svoju liniju rezonovanja i odgovara na pitanja o znanju koje poseduje
3. Fleksibilan – integriše novo znanje inkrementalno na svoje već postojeće znanje [1].

To je sistem koji koristi tehnologije veštačke inteligencije da bi simulirao odlučivanje i ponašanje čoveka ili organizacije koja ima ekspertsko znanje za datu oblast. Tipično, ES uključuje bazu znanja koja sadrži akumulirano iskustvo i zaključivanje (***rule engine***) koje predstavlja skup pravila koja primenjuju bazu znanja na svaku situaciju u kojoj se program nađe. Sposobnosti sistema se mogu povećavati tako što se povećava baza znanja ili skup pravila. Najnoviji sistemi se mogu oslanjati na mašinsko učenje koje im omogućava da poboljšavaju svoje performanse sa povećanjem iskustva, baš kao što čovek to čini [2].

Koncept ekspertskog sistema je prvi put razvijen 1970-ih od strane Edward Feigenbauma, profesora i osnivača ***Knowledge Systems Laboratory*** na Stanford Univerzitetu. Feigenbaum je objasnio da se svet kreće od obrade podataka ka „obradi znanja“, a tranzicija je omogućena pomoću novih procesorskih tehnologija i kompjuterskih arhitektura. Ekspertski sistemi su igrali veliku ulogu u mnogim industrijama kao što su finansijske usluge, telekomunikacije, zdravstvo, korisnički servis, transport, video igre, proizvodnja, avijacija i pisana komunikacija. Dva rana ES su prvi prodrli u zdravstvo za medicinske dijagnoze:

- Dendral – koji je pomogao hemičarima da identifikuju organske molekule
- MYCIN – koji je pomogao u identifikaciji bakterija i preporuci antibiotika i doza

Neki skorije razvijen ES, ROSS je „pravnik veštačke inteligencije“ baziran na IBM-ovom ***Watson cognitive computing*** sistemu. ROSS se zasniva na samoučenju koje koristi ***data mining***, prepoznavanje šablona, ***deep learning*** i procesiranje prirodnog jezika (NLP) da bi oponašao rad ljudskog mozga.

3. ČETBOT

Četbot je softverska aplikacija koja komunicira sa korisnikom putem teksta ili konverzijom teksta u govor, umesto obezbeđivanja direktnog kontakta sa živim ljudskim agentom. Dizajnirani da ubedljivo simuliraju način na koji bi se čovek ponašao kao razgovorni partner, četbot sistemi obično zahtevaju kontinuirano podešavanje i testiranje, a mnogi u proizvodnji i dalje nisu u mogućnosti da adekvatno preusmere ili prođu industrijski standardni Turingov test. Četbot je mašinski sistem za konverzaciju koji interaguje sa ljudskim korisnicima pomoću prirodnog ljudskog jezika [3].

Četbotovi se obično koriste u dijaloškim sistemima za različite svrhe, uključujući korisničku službu, usmeravanje zahteva ili za prikupljanje informacija. Dok neke aplikacije za čet koriste opsežne procese klasifikacije reči, procesore prirodnog jezika i sofisticirane softvere veštačke inteligencije, drugi jednostavno pretražuju opšte ključne reči i generišu odgovore koristeći uobičajene izraze dobijene iz pridružene biblioteke ili baze podataka. Četbotovi zamenjuju neke od poslova koji su tradicionalno obavljali ljudi, kao što je npr. online podrška klijentima [4]. Danas se većini četbotova pristupa putem interneta preko pristupačnih mesta na veb lokaciji ili putem virtuelnih asistenata poput Google Assistant-a, Amazon Alexa ili aplikacija za razmenu poruka kao što su Facebook Messenger ili VeChat. Četbotovi su obično klasifikovani u kategorije korišćenja koje uključuju trgovinu (e-trgovina putem chat-a), obrazovanje, zabavu, finansije, zdravlje, vesti, produktivnost.

Ljudi iz marketinga ih koriste za skriptovanje poruka koje su vrlo slične sekvenci automatskog odgovora. Takve sekvence mogu biti pokrenute korisničkim prijavljivanjem ili upotrebom ključnih reči u korisničkim interakcijama. Nakon aktiviranja, niz poruka se isporučuje do sledećeg očekivanog odgovora korisnika. Svaki se odgovor korisnika koristi u stablu odluke kako bi se pomoglo četbotu da se kreće nizom odgovora kako bi isporučio ispravnu poruku odgovora.

Četbotovi bazirani na pravilima se nazivaju botovi bazirani na stablu odluka. Kao što ime govori, oni rade tako što koriste niz unapred definisanih pravila. Ova pravila su osnova za vrste problema sa kojima je četbot upoznat i za koji može da donese rešenja. Poput dijagrama toka, četbotovi zasnovani na pravilima preslikavaju razgovore. To rade u iščekivanju šta bi Četbotovi zasnovani na pravilima mogu koristiti veoma jednostavna ili komplikovana pravila. Međutim, ne mogu da odgovore na bilo koja pitanja izvan definisanih pravila.

Ovi četbotovi ne uče kroz interakcije. Takođe, rade samo sa scenarijima za koje se treniraju. Za poređenje, četbotovi veštačke inteligencije koji koriste mašinsko učenje razumeju kontekst i nameru pitanja pre formulisanja odgovora. Ovi četbotovi generišu vlastite odgovore na složenija pitanja koristeći odgovore na prirodnom jeziku. Što se više koriste i treniraju ovi roboti to oni više uče i to bolje rade sa korisnikom. Iako botovi zasnovani na pravilima imaju manje fleksibilan razgovorni tok, ove zaštitne šine su takođe prednost. Može se bolje garantovati iskustvo koje će pružiti, dok su četbotovi koji se oslanjaju na mašinsko učenje malo manje predvidljivi.

Neke druge prednosti četbota koji se zasniva na pravilima jesu:

- Oni se obično brže obučavaju (jeftinije)
- Lako se integrišu sa nasleđenim sistemima
- Pojednostavljuju primopredaju ljudskom agentu
- Visoko su odgovorni i sigurni
- Mogu uključivati interaktivne elemente
- Mediji nisu ograničeni na tekst interakcije

A neke od mana četbotova baziranih na pravilima su nedostatak emocija – Za razliku od ljudi, četbotovi nemaju emocije iako to nije ključno da bi se konverzacija uspela usmeriti u pravom smeru. Ljudi iz korisničkog servisa bi mogli da razumeju emociju korisnika i da svoj odgovor usklade sa tim ali četbot nema tu mogućnost zbog toga što su pre- programirani. Naredna mana je to što nisu jednostavni za kreiranje – Zahtevno je kreirati četbot aplikaciju od nule, zahteva puno vremena i truda. Takođe je potrebno i puno resursa, kako finansijskih tako i ljudskih, poput eksperta, programera, menadžera itd. Ovi četbotovi rešavaju pitanja „prvog nivoa“ – To je jedna od najvećih mana ovih sistema što mogu da daju odgovore samo na „jednostavnija“ pitanja ili služe da usmere korisnika u pravom smeru, a nisu u mogućnosti da reše kompleksne zahteve. U takvim slučajevima najčešće usmere korisnika ka nekome iz korisničke službe ko će moći da pomogne sa tim problemom. Još jedna mana je to što zahteva održavanje – Četbotovi zahtevaju stalno pregledanje, održavanje, optimizaciju u smislu baze znanja i načina komunikacije sa korisnicima. Treba se dodavati novo, korisno znanje u bazu pošto se znanje vremenom menja.

Da bi se kreirali četbotovi veštačke inteligencije koriste se algoritmi mašinskog učenja da bi se naučio ljudski jezik koji se uglavnom koristi tokom konverzacija. Procesiranje prirodnog jezika (***natural language processing***) se koristi da bi se dodala veštačka inteligencija u četbotove da bi im to omogućilo razumevanje ljudi i njihovog prirodnog razgovora. Kada je četbot razvijan za posebnu industriju, sektor ili kompaniju tada se ključne reči i izjave koje sadrže određene bitne reči uzimaju u obzir u procesu obrade pomoću NLP-a. Na primer, za elektronsku trgovinu, četbot podaci bi trebali da sadrže normalne upite koji se dešavaju prilikom ***online*** poručivanja, uplaćivanja novca, dostave poručenog proizvoda i zamena ili povrat proizvoda. Kada su takvi podaci prikupljeni, anotiraju se tako da mašina zna koje su važne reči i da može da uči iz tih konverzacija i da odgovara prikladno. U suštini četbot predstavlja aplikaciju mašinskog učenja gde developeri koriste ogromne količine podataka i odgovarajućih algoritama da bi se proizveli najbolji odgovori. Ti podaci sadrže setove sličnih pitanja sa najbitnijim odgovorima koji rešavaju korisnikov problem. Četbot tako uči šta su najbolji odgovori za određenu formu pitanja i koje akcije da preduzme ukoliko se neka beznačajna pitanja pojave. Iako su četbotovi veštačke inteligencije napredniji, oni nisu uvek potrebni. Za manje kompanije ili one sa specifičnim ciljevima, četbotovi zasnovani na pravilima su pogodnije rešenje.

3.1. Agenti

Agenti se mogu podeliti u dve jasno odvojene grupe:

- Četbot agenti
- Virtuelni agenti

Obe grupe su proizvod veštačke inteligencije ali se često svrstavaju kao jedno iako to nisu. Četbot agenti su generalno korišćeni za dobavljanje informacija kao što su na primer podaci o nekom proizvodu, dok su virtuelni agenti mogu da asistiraju prilikom obavljanja neki poslova, kao što su podsećanje na sastanak, upravljanje to- do listom, zapisivanje beleški itd. Ako se Četbot agent upita za takvu asistenciju, „zbuniće se“ i verovatno će stalno ponavljati ista pitanja zbog pojašnjenja. Obe se smatraju konvencionalnim interfejsima ali su suštinski različit.

Za komunikaciju, četbotovi sarađuju sa korisnicima samo kada se dogodi unapred određena radnja, poput korisnika koji upisuje u skočni okvir ili razgovara sa uređajem koji „sluša”. Zatim četbot poravnava ključne reči od korisnika, upoređujući ih sa njihovim memorisanim znanjem. Uzimajući „najverovatniji” odgovor, četbot korisniku šalje skriptovane informacije. Neće „ići sa žice” ili „učiti” na osnovu interakcije. Četbot agenti su automatizovani programi korišćeni kao posrednik u komunikaciji sa čovekom putem teksta ili audio zapisa. Ovaj softver unapređen veštačkom inteligencijom je uglavnom korišćen od strane kompanija da bi obogatio korisnički servis. Četbot agenti imaju presudnu ulogu u korisničkom servisu gde su korišćeni kao sredstvo za dobavljanje informacija.

Virtuelni agenti su napredniji od četbotova: mogu više jer imaju više veština - tehnologije. U poslednje vreme su postali popularni zbog napretka u veštačkoj inteligenciji i kognitivnom računarstvu. Virtuelni agenti su takođe poznati kao virtuelni asistenti ili inteligentni virtuelni agenti. U zavisnosti od obuke koja se pruža, virtuelni agenti se mogu koristiti za razumevanje namere korisnika, pružanje personalizovanih, tačnih odgovora i osnovno rešavanje problema, a sve na ljudski „zvučan” način, mogu izvršiti vrlo ponovljive zadatke, poput pomoći u upravljanju osnovnim računima ili bavljenju prodajnim potencijalima kako bi ih kvalifikovali i razmenjuje naprednije interakcije sa ljudskim agentima, glatko, jasno.

Virtuelni agent je lični, digitalni, softverski baziran agent koji asistira čoveku u obavljanju dnevnih aktivnosti kao što su postavljanje budilnika, zakazivanje sastanaka, obavljanje poziva, kucanje poruka i tome slično. Virtuelni agent je sličan personalnom ljudskom asistentu koji npr. zapisuje beleške u toku sastanka, čita poruke i email-ove itd.

3.2. Najsavremeniji četbot

U 2020. godini jedan od najsavremenijih četbot sistema je Google-ov Meena četbot. Istraživači u Google Brain-u su predstavili Meena, četbot koji može da priča skoro o bilo čemu [5]. Ovaj četbot karakteriše model neuronske mreže od čak 2,6 milijardi parametara koji daje bolje performanse od bilo kog postojećeg konverzacionog modela.

Ovaj model je baziran na popularnom Transformer seq2seq arhitekturi, i njegova specifična arhitektura je otkrivena korišćenjem evolutivnih neuronskih arhitektskih pretraga, sa jednim ciljem, unapređivanje zbunjenosti u toku konverzacija. Evolutivni algoritam pretrage je proizveo arhitekturu u kojoj je bio jedan transformer enkoder blok i čak 13 transformer dekoder blokova. Istraživači pretpostavljaju da je Meena doseže bolji kvalitet konverzacije zbog svog moćnog dekoder modula.

Model je treniran korišćenjem konverzacionih niti (thread) koje su bile organizovane u formi stabla i uzorci su sadržali 7 kontekstnih prelaza između učesnika. Podaci korišćeni da se trenira Meena su dolazili iz konverzacija sa socijalnih mreža i ukupno su sadržali 341 GB teksta koji je prethodno filtriran. To znači da je Meena treniran na čak 40 milijardi reči u periodu od 30 dana.

Da bi procenili uspešnost nove metode koju su koristili, naučnici su odlučili da naprave novu metriku koju su nazvali Sensibleness and Specificity Average. Ova metrika je napravljena da zabeleži najbitnije aspekte prirodnog razgovora. Koristeći ovu metriku, istraživači su sračunali rezultate sedam drugih četbotova pored Meena-e, kao i ljudske performanse. Meena pobedio ostale agente, osvojivši čak 79% SSA rezultata, 23 procenta više nego drugoplasirani četbot Mitsuki i Cleverbot, a samo 7% manje od ljudskog nivoa.

4. IMPLEMENTACIJA JEDNOG EKSPERTSKOG SISTEMA

U ovom poglavlju se rešavao problem kreiranja jednog ekspertskog sistema baziranog na pravilima. Konkretan sistem predstavlja četbot aplikaciju koja je namenjena za generalnu upotrebu i može se primeniti na bilo koju temu sa prethodno pripremljenim odgovarajućim podacima. Projekat je napisan u Python programskom jeziku i ne predstavlja rešenje namenjeno za komercijalne svrhe nego služi da se vidi primena ekspertskih sistema na realan problem i tako prikaže njihova primena u stvarnom svetu. Četbot nije univerzalan, ne može da „priča” o bilo kojoj temi, nego se drži jedne konkretne teme što je u ovom slučaju sport košarka.

Da bi se dobile potrebne informacije o odabranoj temi prvo se učitaju (***scrape***) informacije sa veb stranice i tako dobije potreban tekst o toj temi, a za to se koristi Beautifulsoup4 biblioteka. Nakon toga se tekst parsira u rečenice da bi stvorio zbirku podataka a zatim se koristi Pyton-ova ***regex*** biblioteka (re) za neke predprocesorske zadatke nad tekstom. Sve rečenice iz zbirke, kao i korisnički unos se pretvaraju u vektorsku formu i na osnovu toga se upoređuju kosinusne sličnosti (***cosine similarity***) i rečenica sa najvećom sličnošću biva izabrana kao odgovor na upit. Za ovu četbot aplikaciju je odabran ekspertski sistem baziran na pravilima zbog toga što problem koji rešava nije previše kompleksan da bi zahtevao sisteme sa veštačkom inteligencijom, i ponašanje sistema se može jasno definisati skupom pravila i samim tim se nameće ekspertski sistem baziran na pravilima kao optimalno rešenje. Podaci korišćeni u ovom rešenju se dobavljaju sa veb stranice koja govori o konkretnoj temi za koju se pravi četbot. Podaci se dobavljaju prilikom pokretanja aplikacije i čuvaju se samo u memoriji, zbog toga bi usled ažuriranja veb stranice, podaci u aplikaciji mogli biti različiti prilikom ponovnog pokretanja.

Odgovori su generisani na osnovu kosinusne sličnosti vektorizovanih formi unetih rečenica i rečenica u zbirci

omogućavaju ove dve skripte TfidVectorizer i cosine_similarity. Kao što je već napomenuto, koristiće se Wikipedia članak o košarci da bi se kreirala zbirka. Zbirka predstavlja sve podatke koje imamo o konkretnoj temi u formi teksta. Prvo se dobavi članak sa veb stranice i izdvoji sve paragrafe iz teksta članka, nakon toga je tekst konvertovan u mala slova (lowercase) radi lakšeg procesiranja. Treba predobradom ukloniti sve specijalne karaktere i prazna mesta iz teksta. Predobrada teksta se radi zato što bot ne može da rukuje sa neobrađenim podacima, stoga se oni moraju konvertovati u odgovarajuću formu da bi se optimizovao rad četbota.

Uklanjanje karaktera koji nisu iz alfabeta pada pod predobradu teksta. Tekst se izdeliti u reči i rečenice pošto se kosinusna sličnost korisničkog unosa poredi sa svakom rečenicom. To se vrši pomoću tokenizacije koja rečenicu rastavlja na sastavne delove – reči. Kreiraju se pomoćne funkcije koje će ukloniti znake interpunkcije iz korisničkog unosa i koje će lematizirati (lemmatize) tekst. Lematizacija se odnosi na smanjenje reči na korensku formu. Na primer, lematizacija engleske reči „ate" vraća „eat", reč „throwing" će biti „throw" a reč „worse" će se svesti na „bad".

Kada korisnik unese neki od pozdrava, aplikacija pretraži listu pozdrava od korisnika i ako je reč pronađena onda se nasumično bira neki pozdravi iz liste pozdrava za korisnika. Kreirana je metoda koja prima korisnički unos, nalazi kosinusnu sličnost korisničkog unosa i poredi je sa rečenicama iz zbirke. Koristi se funkcija da bi se pronašla kosinusna sličnost između poslednje stavke u listi svih reči (koja zapravo predstavlja korisnički unos jer je dodat na kraj liste) i rečeničnih vektora svih rečenica u zbirci.

Nakon sortiranja pretposlednja stavka u listi ima najveću kosinusnu sličnost sa korisničkim unosom, a poslednja stavka je sam korisnički unos, pa zbog toga nije selektovan. Nakon toga se izravna (flatten) kosinusna sličnost i proveri da li je jednaka nuli ili ne. Ako je sličnost odgovarajućeg vektora nula (0), to znači da nema odgovora za dati korisnički upit, i u tom slučaju se prikazuje odgovarajuća poruka koja će korisnika obavestiti o tome. U slučaju da kosinusna sličnost nije jednaka nuli, znači da je pronađena slična rečenica kao i korisnički unos i u tom slučaju će se vratiti to kao odgovor.

5. ZAKLJUČAK

Ekspertski sistemi osiguravaju univerzalnu i stalnu raspoloživost eksperata, tj. ekspert je uvek dostupan. Ekspertski sistem treba da obavlja postavljene zadatke da bi ekonomski bio opravdan. Danas ekspertski sistemi imaju široku primenu u praksi, i to u oblastima kao što su medicina, ekonomija, avio-saobraćaj, auto industrija, menadžment...

Budućnost ekspertskih sistema je zagarantovana, pogotovo u ovim oblastima. Ekspertski sistemi će u budućnosti obavljati i mnoge upravljačke funkcije. Velika prednost ekspertskih sistema je kraće vreme odlučivanja, i oni su korisni kod problema kod kojih se odluke zasnivaju na analizi i interpelaciji ogromne količine podataka. U ovom radu se diskutovalo o ekspertskim sistemima, od čega se sastoje, koje su njihove karakteristike i njihov razvoj. Istaknuta je upotreba četbotova u komercijalne svrhe i različite opcije za izgradnju jednog četbot sistema.

Poredili su se i različiti agenti koji služe za asistiranje ljudima, četbot agenti i virtuelni agenti, kao i njihove karakteristike i razlike. Za kraj je ostavljen primer implementacije jednog ekspertskog sistema u četbot aplikaciji. Aplikacija je napisana u Python programskom jeziku uz pomoć dodatnih biblioteka. Ova aplikacija predstavlja četbot koji može da priča samo o određenoj temi, a ta tema je košarka. Cilj ovog rada bio je prezentovanje jednog ekspertskog sistema u vidu četbot aplikacije. Zahtevi aplikacije su bili da se priča o određenoj temi i pravila su jasno definisana pa nije bilo potrebe za uključivanjem mašinskog učenja u proces izgradnje četbota.

U toku rada su poređeni različiti tipovi četbot aplikacija i nakon toga se zaključuje da je ekspertski sistem dobar za ovaj četbot. Iako ovaj bot radi posao za koji je napravljen on se može još usavršiti. Proširenje skupa podataka bi za početak doprinelo tome da sistem ima više znanja i da može odgovarati na više pitanja.

6. LITERATURA

[1] Buchanan, Bruce G., and Richard O. Duda. "Principles of rule-based expert systems." Advances in computers. Vol. 22. Elsevier, 1983. 163-216.

[2] Rouse M. (Jun 2016.) Expert System. Search Enterprise AI. Preuzeto sa https://searchenterpriseai.techtarget.com/definition/exp ert-systemAaron Parecki, An Introduction to OAuth 2. O'Reily Webcast, 2012

[3] HAWAR, Bayan Abu; ATWELL, Eric Steven. Using corpora in machine-learning chatbot systems. International journal of corpus linguistics, 2005, 10.4: 489-516.

[4] Io, H. N., & Lee, C. B. (2017, December). Chatbots and conversational agents: A bibliometric analysis. In 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) (pp. 215-219). IEEE.

[5] Adiwardana D. & Luong T. (28. Januar 2020.) Towards a Conversational Agent that Can Chat About…Anything. Google AI Blog. Preuzeto sa https://ai.googleblog.com/2020/01/towards-conversational-agent-that-can.html

Kratka biografija:

Mario Gula rođen je u Subotici 1995. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranio je 2020. godine.
kontakt: gulamario95@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE35Mijatovic

NODE.JS BAZIRANI DISTRIBUIRANI SISTEM ZA DUGOTRAJNO ČUVANJE DIGITALNIH DOKUMENATA

NODE.JS BASED DISTRIBUTED SYSTEM FOR DIGITAL DOCUMENTS LONG-TERM PRESERVATION

Marko Mijatović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je predstavljena arhitektura i implementacije sistema za pravljenje kopija digiralnih dokumenata. Sistem prati promene dokumenata i dodavanje novih. Šalje njihove kopije na čuvanje na više lokacija.*

Ključne reči: *čuvanje dokumenata, bekap*

Abstract – *The paper presents the architecture and implementation of a system for making copies of digital documents. The system monitors changing documents and adding new ones. Sends their copies for keeping to multiple locations.*

Keywords: *document storage, backup*

1. UVOD

Tokom korišćenja računara generiše se veliki broj dokumenata različitog formata. Većina dokumenata korisna je samo na kratak vremenski period, ali za neke će biti potrebno trajno čuvanje. Kako bi se osiguralo da vremenom ostanu pouzdani i korisni, neophodno je pripremiti dobro osmišljen i dokumentovan plan za njihovo čuvanje. Kada se hard disk ili drugi skladišni medijum pokvari, a nije napravljena rezervna kopija, tada je već prekasno za bilo kakvo rešavanje problema. Zato je potrebno razmišljati unapred, jer nije pitanje kada će hard disk prestati sa radov, već kada će se to desiti. Postoji i slučaj kada je fajl oštećen, a korisnik ne zna i misli da je fajl ispravan. Kada se nad takvim fajlom napravi rezervna kopija i ona je za korisnika neupotrebljiva kao i originalni fajl. Kada korisnik sazna da su i fajl i kopija oštećeni, nema načina da povrati fajl. Bitno je proveravati važne fajlove na vreme i utvrditi njihovu ispravnost kako bi se pravila njihova rezervna kopija dok su u ispravnom stanju.

Tema ovog rada jeste kreiranje sistema koji bi periodično obavljao posao provere digitalnih dokumenata i njihovo skladištenje kod drugih izabranih korisnika sistema. Provera digitalnih dokumenata podrazumeva detektovanje da li je došlo do izmene na dokumentu ili do njegovog oštećenja ili gubitka.

Dokumente koji su izmenjeni potrebno je ponovo dostaviti korisnicima na čuvanje kako bi se u sistemu uvek nalazile poslednje verzije dokumenata.

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

2. TEORIJSKE OSNOVE

Sistemi za bekap podataka uglavnom su projektovani kao distribuirani sistemi. Serveri, na koje korisnici postavljaju i na kojima se čuvaju njihovi fajlovi, su distribuirani. Bekap sistemi rade sa digitalnim dokumentima, pa će u ovom poglavlju biti pojašnjen pojam digitalnih dokumenata, uz osljanjanje na knjigu [1].

2.1. Digitalni dokumenti

Preteča digitalnih dokumenata su papirni dokumenti. Kada gledamo iz ugla današnje tehnologije i njenog napretka, oni izgledaju prilično zastarelo. Sa druge strane, oni još uvek imaju svoje prednosti koje se ogledaju u tome što nije potrebno posedovati skup uređaj sa programima, koji se takođe plaćaju, da bi se dokument kreirao i koristio.

Informacije snimljene za dalje korišćenje, pregled i obradu, na način koji zahteva računar ili drugi elektronski uređaj opisane su kao digitalni dokumenti. Razvoj interneta i masovno korišćenje računara, gde su računari međusobno povezani radi lakše komunikacije, dovelo je do situacije gde se na dnevnom nivou kreira ili izmeni velika količina digitalnih dokumenata.

2.2. Bekap podataka

Bekap [2] je mehanizam u sistemima koji su namenjeni za oporavak od katastrofe. Katastrofe koje dovode do gubitka podataka se neretko dešavaju i mi u većini slučajeva ne možemo da utičemo na njih. Možemo samo da sistem i podatke vratimo na stanje pre katastrofe i da ublažimo njene posledice. Bekap mehanizmom vrši se dupliranje, kopiranje podataka na sigurno mesto. Moguće je raditi bekap na dnevnom, nedeljnom ili mesečnom nivou. Potrebno je pravilno izabrati vremenski period uzevši u obzir količinu novih digitalnih dokumenata i izmenu postojećih. Optimalno rešenje je kombinovati više vremenskih perioda, bitne dokumente nad kojima se često vrše izmene bekapovati na dnevnom nivou a ostale na nedeljnom ili mesečnom.

2.3. Dugotrajno čuvanje podataka

Dugotrajno čuvanje podataka [3] predstavlja niz izazova, od tehničkih do društvenih i organizacionih. Tehnički izazvo je osigurati da podaci danas mogu preživeti dugotrajne promene u medijima za skladištenje, uređajima i formatima podataka. Dugotrajno čuvanje podataka ima tri glavna cilja:

- Sačuvati dokument,
- Osigurati pristupačnost,
- Očuvati razumljivost.

2.4. File Fixity

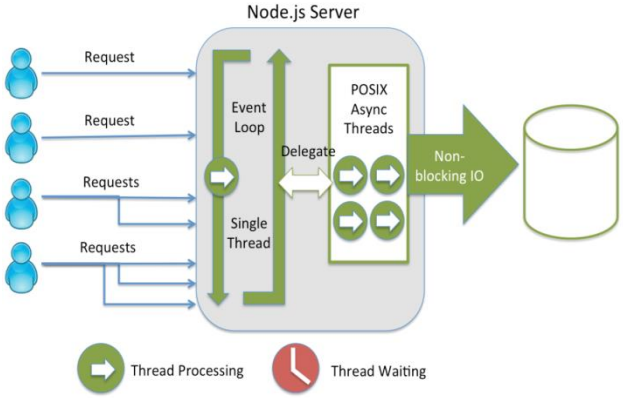
File Fixity [4] predstavlja sigurnost da je digitalni dokument ostao nepromenjen i ispravan. File Fixity se ne odnosi samo na datoteke, već na bilo koji digitalni objekat koji u sebi ima niz bitova, gde je potrebno da je taj niz ostao nepromenjen. Provera ispravnosti datoteka ne osigurava očuvanje digitalne datoteke. File Fixity Omogućava skladištu da identifikuje oštećene datoteke i da pokrene postupak njihove zamene sa ispravnom kopijom.

3. KORIŠĆENE TEHNOLOGIJE

Glavni zahtev sistema bio je da ima mogućnost izvršavanja na različitim operativnim sistemima. Za ispunjenje takvog zahteva bilo je potrebno izabrati tehnologije koje neće biti ograničene izvršavanjem na samo jednoj platformi. Iz tog razloga sistem je implementiran u Node.js i tehnologijama koje su zasnovane na JavaScript jeziku.

3.1. Node.js

Node.js [5] se koristi za razvoj veb aplikacija visokih preformansi. U osnovi je JavaScript okruženje u kojem se izvršavaju programi serverske strane u realnom vremenu. JavaScript je primarno korišćen na klijentskoj strani, gde su skripte napisane u JavaScriptu bile ugrađene u HTML stranice. Node.js omogučava da se JavaScript koristi za pisanje skripti na serverskoj strani. Takva mogućnost je značajna jer nije potrebno poznavanje dodatnih jezika za kompletan razboj veb aplikacije. Node.js ima jednu nit koja obrađuje pristigle zahteve prikazan na slici 1.



Slika 1. *Asinhroni način rada Node.js*

On koristi asinhroni i neblokirajući režim rada. Kod sinhronog izvršavanja, operacija može da krene izvršavanje samo ako je prethodna operacija završila svoj rad. Kod asingronog izvršavanja, operacije ne čekaju jedna drugu nego mogu istovremeno da se izvršavaju. Vreme izvršavanja svake operacije posebno ostaje isto, ali ukupno vreme izvršavanja svih operacija se drastično smanjuje ukoliko svaka može da se izvrši u zasebnoj niti.

3.2. MongoDB

Mongo baza [6] je nerelaciona baza podataka. NoSQL baze podata nisu tabelarne i podatke čuvaju drugačije od relacionih tabela. NoSQL baze imaju drugačije tipove u modelu na osnovu kojih skladište podatke. Glavni tipovi su:

- dokumenti,
- ključ/vrednost
- grafik,
- široke kolone

Pružaju fleksibilne šeme i lako se skaliraju sa velikim količinama podataka i velikim opterećenjem upitima nad bazom. NoSQL baze podataka mogu da čuvaju podatke o odnosima između čuvanih podataka.

Takve baze jednostavno taj podatak čuvaju drugačije u odnosu na relacione baze podataka, povezani podaci ne moraju da se dele u tabele.

NoSQL baze podataka omogućavaju programerima da čuvaju ogromne količine nestrukturiranih podataka, pružajući im veliku fleksibilnost.

4. SPECIFIKACIJA SISTEMA

Čuvanje dokumenata moguće je na različite načine. Većina popularnih sistema za čuvanje korisničkih podataka funkcionišu tako što poseduju velika skladišta podataka na različitim mestima. Korisnici ne znaju gde im se fajlovi nalaze i da li se svi fajlovi skladište na istom mestu. Ovaj sistem zamišljen je tako da nudi zanimljiv način čuvanja korisničkih dokumenata.

Korisnici u sistemu čuvaju kopije svojih dokumenata kod drugih korisnika. Jedna pristupna tačka koja preuzima i skladišti korisničke fajlove dok ih svi korisnici koji su garantovali njihovo čuvanje ne preuzmu. Kada svi korisnici preuzmu fajlove oni se brišu sa pristupne tačke radi oslobađanja njenog ograničenog skladišnog prostora.

Prednosti ovakve arhitekture sistema ogledaju se u tome što korisnici tačno znaju gde i kod koga se kopije njihovih dokumenata čuvaju. Nije potrebna velika količina prostora na jednom mestu.

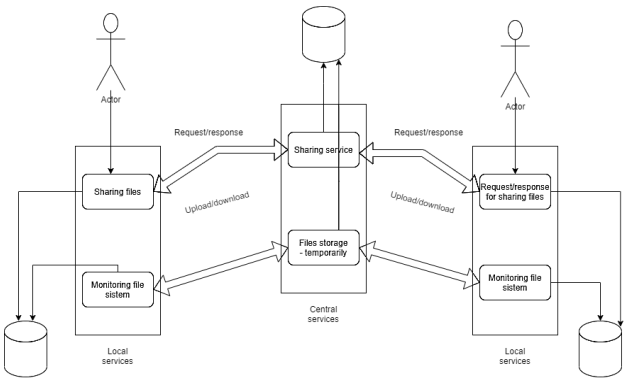
4.1. Arhitektura sistema

Sistem je realizovan kao veb aplikacija, zato je potrebno da su računari, koji će biti korisnici sistema, povezani na internet ili međusobno umreženi sa centralnim serverom. U slučaju da se za centralni server obezbedi javna IP (Internet Protocol) adresa, sistem bi mogao da funkcioniše tako što bi ostali korisnici uz povezivanje na internet mogli da mu pristupe sa bilo kog mesta.

Sistem je zamišljen da posle konfigurisanja može samostalno da radi bez interakcije sa korisnikom. U sistemu na svakoj korisničkoj mašini postoji lokalni servis koji obavlja glavni posao na korisničkim računarima i komunicira sa centralnim serverom. Skladištenje podataka obavlja se na svakom korisničkom računaru i na centralnom serveru. Korisnici imaju i veb aplikaciju pomoću koje je moguć uvid u fajlove koji se bekapuju i slanje zahteva drugim korisnicima za skladištenje kao i prihvatanje i odbijanje zahteva.

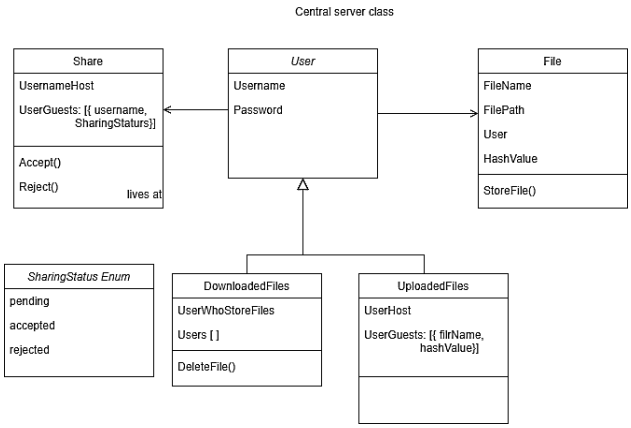
Veb aplikacija svaku komunikaciju sa centralnim serverom obavlja preko lokalnog servisa. Lokalni servis jednog korisnika nema direktnu komunikaciju sa drugim korisnicima. Lokalni servis prihvata svaki zahtev, beleži ga ako je potrebno i prosleđuje centralnom serveru. Na slici 2 može se videti arhitektura sistema.

Svi korisnici sistema imaju iste privilegije. Nije bilo dodatnih funkcionalnosti sistema koje bi zahtevale korisnike sa dodatnim privilegijama ili odvajanja korisnika na grupe.



Slika 2. *Arhitektura sistema*

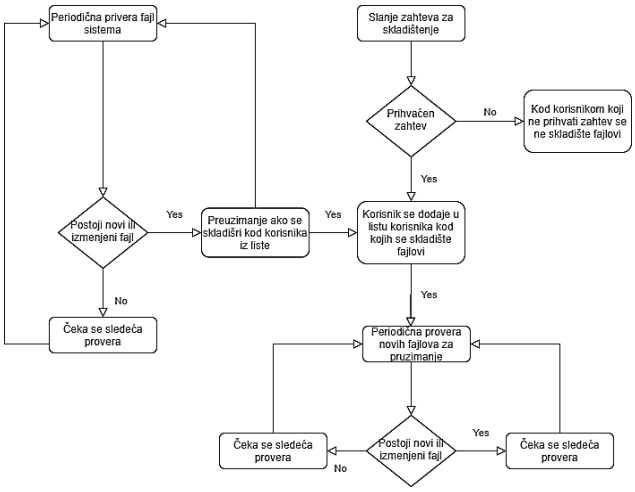
Dijagram klasa prikazan je na slici 3.



Slika 3. *Dijagram klasa sistema*

Sistem ovako zamišljen i implementiran ima tri bitne faze kroz koje prilazi tokom rada, koje se mogu videti i na slici 4:

- slanje i odgovor na zahtev za čuvanje dokumenata
- provera lokalnog direktorijuma i slanje fajla na čuvanje
- provera da li postoje fajlovi drugih korisnika koje treba preuzeti.



Slika 4. *Dijagram toka sistema*

5. IMPLEMENTACIJA

Sistem čine front-end i dve serverske aplikacije. Front-end je napisan u Angular okruženju a serverske strane u Node.js. Sve je razvijano u Visual Studio Code razvojnom okruženju, a za čuvanje podataka korišćena je Mongo baza podataka.

Front-end deo aplikacije sastoji se iz početne strane, pregleda korisnikovih fajlova, slanja zahteva za deljenje i pregleda notifikacija i odgovora na njih. Početna strana nudi izbor prelazaka na neku od ostale tri stranice. Na stranici za pregled fajlova korisnika, prikazani su fajlovi iz direktorijuma izabranog od strane korisnik da se prati njegovo stanje. JavaScript koji se pokreće u pretraživaču ima ograničen pristup lokalnom fajl sistemu, zato je implementacija zahtevala pomoć sa lokalnog servisa. Upućen je zahtev za dobijanje korisnikovih fajlova.

Lokalni servis poseduje otvoren API za front-end deo aplikacije. Korišćen je Express, postavljen body-parser i cors da bi zahtevi mogli da se upute iz Angular aplikacije. Lokalni server je otvoren na portu 52296.

Sledeći deo servisa je periodična provera izabranog direktorijuma. Za taj deo izabrana je node-schedule biblioteka. Podešeno je vreme na koliko će se vršiti provera i definisana je funkcija koja će obavljati proveru, listing 1.

```
schedule.scheduleJob('30 * * * * *',
async function(){
      var allFiles =
      monitoring.AllFiles(dir);
      await
      monitoring.isFileExist(dir,
allFiles);
});
```

Listing 1. *Pokretanje periodične provere*

Za svaki fajl u direktorijumu proverava se da li postoji sačuvan u Mongo bazi. Ako fajl ne postoji, čuvaju se njegov naziv, putanja do fajla, ime korisnika čiji je fajl i računa mu se hash vrednost prikazana u listingu 2.

```
const {hashElement} = require('hash-
folder');
const options = {algo: 'sha1'};
const fileHash = await
hashElement(filePath, options);
```

Listing 2. *Računanje hash vrednosti fajla*

Zatim se fajl šalje centralnom serveru koji ima zadatak da ga prosledi ostalim korsinicima i da ga izbriše iz svog skladišta. U slučaju da fajl postoji sačuvan u bazi, računa se njegova trenutna hash vrednost. Ona se upoređuje sa vrednošću koja je upisana u bazi. Ako se razlikuju te dve vrednosti znači da je došlo do izmene fajla i da je potrebno poslati novu verziju korisnicima na čuvanje. Sa novom verzijom fajla šalje se i njegova nova hash vrednost.

Na sličan način se preiodično obavlja i provera da li na centralnom serveru ima novih fajlova koje su postavili drugi korisnici, a potrebno ih je sačuvati. Na lokalnom serveru implementirana je klijentska strana FTP protokola. Pomoću njene implementacije vrši se slanje i pruzimanje fajlova sa centralnog servera. Prvi korak pre slanja je otvaranje komunikacije sa FTP serverom. To je omogućeno pomoću metode access(), kojoj su prosleđeni

adresa FTP servera, korisničko ime i lozinka iz konfiguracionog fajla. Nakon uspostavljanja komunikacije sledi provera da li se FTP serveru šalje fajl ili direktorijum i nakon toga slanje.

Centralni server ima otvoren API za komunikacije sa lokalnim servisima koji su pokrenuti kod svakog klijenta. Dalje ima podignut FTP server koji prima ili šalje fajlove klijenata. Kao i lokalni servis, koristi konfiguracioni fajl. U njemu je zadata putanja gde će se čuvati fajlovi i adresa na kojoj će biti podugnit FTP server. Podaci se čuvaju u Mongo bazi, a jedan od najbitnijih podataka za sinhronizaciju fajlova je njihova hash vrednost. Kada na centrani server stigne fajl, HTTP zahtevom stižu i informacije o tom fajlu koje sadrže hash vrednost. Korsnici koji preuzimanju fajlove, prvo zatraže informacije o fajlu. Nakon toga sledi provera da li imaju taj fajl i da li je to poslednja verzija. Provera se vrši upoređivanjem hash vrednosti. Kada svi korisnici preuzmu fajlove sa centralnog servera sledi oslobađanje prostora. Provera da li je moguće osloboditi prostor obavlja se tako što se upoređuju dve šeme u Mongo bazi, SharedSchema i DownloadedSchema.

6. ZAKLJUČAK

Tema ovog rada je pravljenje rezervnih kopija digitalnih dokumenata. Predstavljen je sistem koji je implementiran kako bi ispunio potrebe bekapa i skladištenja podataka na više lokacija. Objašnjeni su pojmovi koji su bitni za temu skladištenja podataka. Preporuka za dalji razvoj je uključivanje više bezbednosti u sistem. Protokole koji su korišćeni zameniti sa njihovim bezbednijim verzijama. Skladištenje podataka i konfiguracinog fajla osigurati od neželjenog pristupa. Proširenje sistema korsnicima sa više privilegija kao što su administratori moglo bi biti od koristi. Ako bi sistem bio u široj upotrebi bilo bi korisno da neko ima pristup i mogućnost nadgledanja i upravljanja takvim sistemom, a ostalim korisnicima uskratiti takve funkcionalnosti.

7. LITERATURA

[1] Dragan Ivanović, Branko Milosavljević. "Upravljanje digitalnim dokumentima", 2015.

[2] Azad Adam. "Implementing Electronic Document andRecord Managment Systems", 2007.

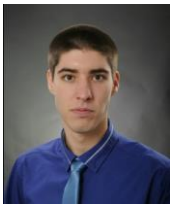
[3] Long-term preservation, https://www.cines.fr/en/long-term-preservation/a-concept-problems-2/long-term-preservation-concept/ (pristupljeno u oktobru 2020.)

[4] File Fixity, https://en.wikipedia.org/wiki/File_fixity (pristupljeno u oktobru 2020.)

[5] Node.js documentation, https://nodejs.org/en/docs/ (pristupljeno u septembru 2020.)

[6] MongoDB Manual, https://docs.mongodb.com/manual/ (pristupljeno u septembru 2020.)

Kratka biografija:



Marko Mijatović rođen je 29.8.1996. godine u Beogradu. Osnovnu školu „Svetozar Marković Toza“ završio je 2011. godine. Gimnaziju „Isidora Sekulić“ u Novom Sadu završio je 2015. godine. Iste godine upisao se na Fakultet tehničkih nauka, smer Primenjeno softversko inženjerstvo. Diplomirao u roku. Školske 2019/2020. godine upisao je master studije, smer Računarstvo i automatika, modul Elektronsko poslovanje. Položio je sve ispite predviđene planom i programom.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE36Gligorov

IMPLEMENTACIJA SIMULATORA ZA GAZEPOINT EYE TRACKER KAMERU

GAZEPOINT EYE TRACKER CAMERA SIMULATOR IMPLEMENTATION

Nenad Gligorov, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *Eye tracking je senzorska, hardverska tehnologija koja omogućava računarima ili drugim uređajima praćenje korisničkog pogleda. Eye tracker kamera može da detektuje prisustvo, pažnju i fokus korisnika. Omogućava jedinstven uvid u razumevanje ljudskog faktora prilikom korišćenja računara i olakšava izradu što prirodnijeg korisničkog interfejsa za širok spektar digitalnih uređaja. Takođe, mogućnost upravljanja računarom uz pomoć očiju predstavlja potencijalno olakšanje za korisnike kojima je onemogućen govor ili korišćenje ruku. Ovakav tip kamere je veoma skup, te je zbog toga nepristupačan većini populacije. Potencijalno rešenje ovog problema može da bude Gazepoint Eye Tracker simulator. U radu je detaljno opisan postupak implementacije softvera, način njegovog rada, kao i princip određivanja simuliranog korisničkog pogleda. Za izradu simulatora korišćen je radni okvir Angular, koji se odnosi na prezentacijski sloj aplikacije, dok će se za samu biznis logiku pobrinuti radni okvir Spring Boot u čijoj se osnovi nalazi programski jezik Java.*

Ključne reči: *eye-tracker, Spring Boot, Angular, Java, regioni od interesa, Markovljevi lanci, slučajni procesi, HTML, CSS*

Abstract – *Eye tracking is a senzor, hardware technology which allows computers or other devices to track user viewing point. Eye tracker camera can detect presence, attention and focus of a user. Also, it represents a potential relief for visually impared users or users unable to use their hands. This type of camera is very expensive, thus being unavailable to most of user population. Potentially, Gazepoint Eye Tracker simulator could solve this problem. Detailed process of implementation of the software, as well as the principle of determination of user viewing point is explained in this paper. Angular framework was used for the implementation of the front end of the application, while Spring Boot, whose core language is Java, was used to handle the bussiness logic.*

Keywords: *eye-tracker, Spring Boot, Angular, Java, regions of interest, Markov chains, random processes, HTML, CSS*

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

1. UVOD

Eye tracking je proces merenja kako tačke pogleda, tako i pomeraja oka u odnosu na glavu. Eye tracker je uređaj koji se koristi za merenje pozicije očiju i njihovog pomeraja. Najviše se koriste u istraživanjima vezanim za vizuelne sisteme, psihologiju, psiholingvistiku, marketing, kao ulazni uređaj pri interakciji čoveka sa računarom. Sve više ih koriste aplikacije za rehabilitaciju i pomoć hendikepiranim osobama.

S obzirom na njegovu nepristupačnost, koja proizilazi prvenstveno iz visoke cene, a koja u proseku iznosi oko 17500 dolara, stvara se potreba za izradom softverskog simulatora koji bi bio sposoban da u velikoj meri pokrije funkcionalnosti stvarnog uređaja. Konkretno, ideja za implementaciju ovakvog softvera nastala je prilikom implementacije Web platforme za ocenjivanje znanja uz praćenje pokreta oka [7], kada Gazepoint Eye tracker kamera [8] nije bila dostupna prilikom ulaska u fazu testiranja.

2. PROGRAMSKI JEZIK JAVA

Istorija Jave [1] počinje 1991. Godine, inicijalnim projektom pod nazivom "Java language project". U projektu su učestvovali Džejms Gosling (eng. James Gosling), Majk Šeridan (eng. Mike Sheridan) i Patrik Noton (eng. Patrick Naughton). Ovaj tim inženjera ubrzo je nazvan "Zelenim timom" (eng. Green Team).

Zanimljivo je napomenuti da je Java prvobitno dizajnirana u svrhe interaktivne televizije. Ideja je bila napraviti jezik pogodan za digitalne uređaje kao što su moderni televizori i set-top box uređaji. Kako će se kasnije ispostaviti, sama ideja je bila previše napredna za koncepte digitalne kablovske televizije devedesetih godina prošlog veka. Java je tako našla svoju primenu u web programiranju, da bi je nedugo zatim kupila firma Netscape.

Programski jezik Java [2] je platformski nezavistan, u skladu sa write once, run anywhere (WORA) filozofijom. Ovo podrazumeva da jednom napisan Java kod na bilo kojoj platformi, odnosto operativnom sistemu, kasnije može biti pokrenut na bilo kom drugom operativnom sistemu.

Java je potpuno objektno orijentisana. Svaki element je klasa, njen deo ili objekat neke klase. Koristi se za razvijanje desktop i mobilnih aplikacija, big data processing, embedded sisteme, web aplikacije. Nasleđivanje, enkapsulacija, polimorfizam i apstrakcija – osnovni koncepti Jave, kao i svakog drugog objektno-

orijentisanog jezika, u velikoj meri su olakšali implementaciju ovog simulatora.

3. SPRING FRAMEWORK I SPRING BOOT

Spring Boot [3] olakšava implementaciju stand-alone, production grade aplikacija zasnovanih na Spring platformi [4]. Moguće je koristiti Spring Boot za kreiranje Java aplikacija koje potom mogu biti pokrenute standardnim java –jar naredbama ili izvežene (eng. deployed) u war formatu.

Podržava rad sa Maven i Gradle alatima za izgradnju aplikacije. Takođe, omogućava rad sa više servletskih kontejnera, kao što su:

• Tomcat 9.0 (4.0 servletska verzija),

• Jetty 9.4 (3.1 servletska verzija),

• Undertow 2.0 (4.0 servletska verzija).

Sveukupno, Spring Boot značajno olakšava izradu kompleksnih enterprize aplikacije, bile one monolitne ili zasnovane na mikroservisima koristeći samo POJO-e (plain old Java object). Na taj način se izbegava potreba za korišćenjem EJB kontejnera, ali se i dalje ostavlja opcija za korišćenje nekog od gorepomenutih robusnih servletskih kontejnera.

Modularna organizacija omogućava usredsređivanje samo na one kole i pakete na kojima se trenutno radi, nezavisno od njihovog ukupnog broja.

Posebno je pogodan za implementaciju skalabilnih aplikacija. Podržava rad sa XML formatom podataka, kao i slanje tih istih podataka preko SOAP protokola, dok je Spring-ova web platforma dizajnirana po MVC šablonu.

Za razvijanje Gazepoint Eye tracker simulatora je ipak korisnija bila činjenica da je u Spring Boot-u, kao i u Spring¬-u, podržan rad sa REST servisima, kao i rad sa Web Socket-ima.

Takođe, dostupan je veliki broj API-a za povezivanje sa bazom podataka koji omogućavaju i umnogome olakšavaju code-first pristup projektovanju baze, pravljenje upita nad bazom, kao i upisivanje podataka u istu.

Ne treba izostaviti ni API za security, koji, iako komplikovan za inicijalno podešavanje, kasnije pruža veliku fleksibilnost pri zaštiti korisničkog pristupa REST endpoint-ima.

4. ANGULAR

Angular [5] predstavlja okruženje za razvijanje aplikacija na klijentskoj strani, odnostno frontend aplikacija. Baziran je na typescript-u. Prvi put se pojavio kao AngularJS 2010. godine. Prvobitno je razvijen kao biblioteka javascript-a, da bi 2014. počeo razvoj Angulara 2. Ovo je prva verzija razvijena u typescript-u. Za potrebe ovog projekta korišćen je Angular 8.

Najbitnija karakteristika bila bi mogućnost razvijanja single-page aplikacija. To znači da se cela aplikacija učitava na jednoj web stranici. Prilikom učitavanja novih ili promene podataka nema potrebe za ponovnim učitavanjem cele stranice. Samo onaj deo stranice koji se izmenio se ponovo učitava. Ovakav način rada drastično ubrzava rad, jer se izbegava učitavanje redudantnog sadržaja.

Takođe, bitno je napomenuti da se ceo kod izvršava direktno u pretraživaču. Budući da su pretraživači u suštini javascript kompajleri, prilikom kompajliranja Angular aplikacije celokupan kod se mašinski prevodi iz typescript-a u javascript.

Angular podržava dva režima rada:

• razvojni režim – programiranje i debugging se vrše u typescript-u,

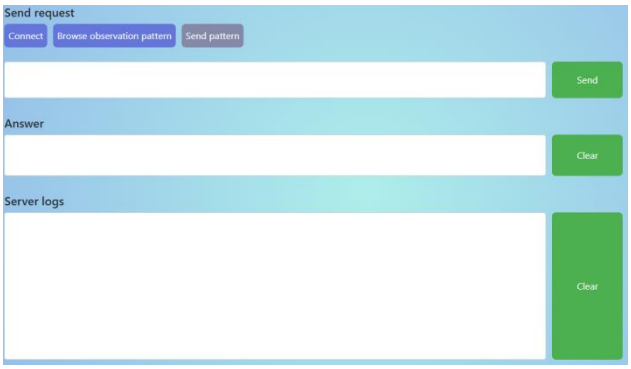
• produkcioni režim – kod se trajno prevodi u javascript i postavlja na neki web server.

Bazira se na MVC šablonu. Veoma bitna osobina je i data-binding, koja olakšava rad sa podacima iz modela.

5. O APLIKACIJI

Gazepoint Eye tracker simulator je web aplikacija uz pomoć koje je moguće simulirati rad Eye tracker hardvera. Korisnički softver je moguće povezati sa simulatorom uz pomoć Web socket-a, preko porta 4242 (što je ujedno i konekcioni port stvarnog uređaja).

Sama aplikacija je razdvojena u dva dela – prezentacioni sloj, koji je razvijen u Angular framework-u i sloj biznis logike, koji je implementiran uz pomoć Spring Boot platforme. Prezentacioni sloj simulatora je prikazan na slici 1.



Slika 1. *Prezentacioni sloj simulatora*

U gornjem levom uglu se nalazi dugme Connect. Pritiskom na ovo dugme izveštava se prezentacioni sloj o potrebi za povezivanjem sa Web socket-om na portu 4242. Na ovaj način se vrši povezivanje sa slojem biznis logike i omogućava slanje zahteva simulatoru, kao i primanje njegovih odgovora.

U tekstualnom bloku Send request se pišu zahtevi za slanje, u formatu srodnom XML-u, kao što je prikazano na slici 2.

Slika 2. *Zahtev za slanje*

U trenutnoj implementaciji omogućeni su sledeći zahtevi:

• ENABLE_SEND_DATA – get i set, podešavanje uslova za omogućavanje slanja podataka preko socket-a

• ENABLE_SEND_COUNTER – get i set, podešavanje uslova za omogućavanje slanja sistemskog brojača – pogodno za praćenje kontinuiteta primljenih odgovora

• CALIBRATE_START – get i set, simulira pokretanje kalibracije uređaja

• CALIBRATE_TIMEOUT – get i set, vrednost timeout flag-a

• CALIBRATE_DELAY – get i set, vrednost zakašnjenja kalibracije

• SCREEN_SIZE – get i set, dimenzije ekrana

• ENABLE_SEND_POG_BEST – set, podešavanje uslova za omogućavanje slanja koordinata korisničkog pogleda

Nakon poslatog zahteva, u tekstualnom bloku Answer se ispisuje odgovor servera. Tekstualni blok Server logs u realnom vremenu prikazuje serverske logove.

Kako je ovde reč o simulatoru, odnosno softverskoj komponenti, potrebno je zadati određen obrazac posmatranja simuliranog korisnika.

Celokupan ekran je podeljen u matricu regija od interesa, od kojih svaka predstavlja moguću poziciju korisničkog pogleda. Koristeći zadati obrazac, a uz pomoć lanaca Markova, vrši se proračun naredne regije interesa u koju će simulirani korisnik gledati. Naredna regija interesa može biti neka od susednih trenutnoj, ili se pogled može zadržati na istoj poziciji, kao što je prikazano prvom matricom na slici 3.

```
P(Xn+1 = Up) = x
P(Xn+1 = Right) = y
P(Xn+1 = Left) = z
P(Xn+1 = Down) = k
P(Xn+1 = Current) = j

[
    UL  UC  UR

    L   C   R

    DL  DC  DR
                ]
                    UL  UC  UR  L   C   R   DL  DC  DR
P = P(1) =          [
                UL      *   *   *   *   *   *   *   *
                UC      *   *   *   *   *   *   *   *
                UR      *   *   *   *   *   *   *   *
                L       *   *   *   *   *   *   *   *
                C      x+z x+j x+y  z   j   y  k+z k+j k+y  => (sum) = 3
                R       *   *   *   *   *   *   *   *
                DL      *   *   *   *   *   *   *   *
                DC      *   *   *   *   *   *   *   *
                DR      *   *   *   *   *   *   *   *
                                                        ]
    3x +  3y +  3z + 3k + 3j = 3   <=>  (T1)
```

Slika 3 *Delovi koda*

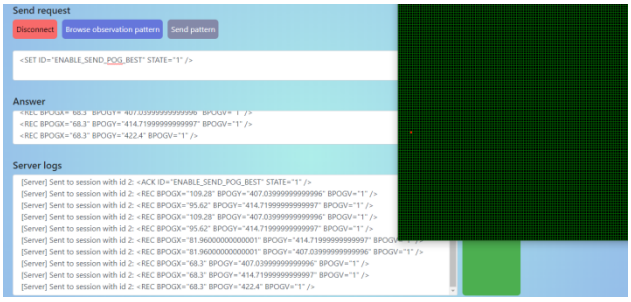
Budući da se Markovljevim lancima računa verovatnoća prelaska pogleda sa bilo kog polja na bilo koje buduće, za proračun simulatora nije potrebno sračunavati sve vrednosti u drugoj prikazanoj matrici. Potrebno je pronaći samo vrednosti koje se tiču trenutne gledane regije interesa. To u suštini znači da se računanje verovatnoće prelaska pogleda pojednostavljuje – više nije potrebno određivati celu Markovljevu matricu, već samo jedan njen red (sve vrednosti u ostalim redovima su na slici 3 popunjene zvezdicama).

Ovaj postupak značajno ubrzava proces računanja verovatnoća. Kako je poznata trenutna pozicija korisničkog pogleda, gubi se smisao računanja verovatnoća za sve ostale pozicije.

Takođe, potrebno je sračunati verovatnoće samo za jedan korak u budućnosti, imajući u vidu da simulator ima sposobnost praćenja trenutne situacije.

Pritiskom na dugme Browse observation pattern vrši se odabir obrasca korisničkog pogleda. Obrazac je potrebno zadati u JSON formatu. Dugme Send pattern omogućava slanje odabranog obrasca serveru.

Slanjem zahteva ENABLE_SEND_POG_BEST se otvara dodatni prozor koji modeluje matricu regija interesa. Crvenom bojom se u realnom vremenu obeležava trenutno posmatrana regija interesa. Izgled modelovane matrice je prikazan na slici 4.



Slika 4. *Modelovana matrica*

5. ZAKLJUČAK

Problem koji se rešavao u ovom radu tiče se implementacije softverskog simulatora Gazepoint eye tracker kamere. Opisan je način rada simulatora, kao i način odabira sledeće pozicije korisničkog pogleda korišćenjem slučajnih procesa.

Sledeći korak u unapređenju ovog simulatora je dodavanje novih zahteva ka serveru, kao i njihova obrada. Pored toga, moguća je i optimizacija algoritma za Markovljeve lance.

6.LITERATURA

[1] Java programiranje- https://www.programiz.com/java-programming (posećeno 12. juna 2020)

[2] Istorija Jave - https://www.javatpoint.com/history-of-java (posećeno 12. juna 2020)

[3] P. Webb, D. Syer, J. Long, S. Nicoll, R. Winch, A. Wilkinson, M. Overdijk, C. Dupuis, S. Deleuze, M. Simons, V. Pavić, J. Bryant, M. Bhave, E. Melendez, S. Frederick „Spring Boot Reference Documentation“

[4] Spring Framework - https://spring.io/ (posećeno 12. juna 2020)

[5] Angular - https://angular.io/ (posećeno 13. juna 2020.)

[6] T. N. Cornsweet, H. D. Crane “Accurate two-dimensional eye tracker using first and fourth Purkinje images”

[7] N. Gligorov „Web platforma za ocenjivanja znanja uz praćenje pokreta oka“

[8] Gazepoint – https://www.gazept.com (posećeno 19. avgusta 2020)

Kratka biografija:

Nenad Gligorov rođen je u Kraljevu 1996. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Primenjene računarske nauke i informatika odbranio je 2020. god.
kontakt: ngligorov@uns.ac.rs

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE37Cabrilo

PREDIKCIJA CIJENE AIRBNB SMJEŠTAJA UPOTREBOM ALGORITAMA MAŠINSKOG UČENJA

PREDICTING AIRBNB PRICES USING MACHINE LEARNING ALGORITHMS

Miljan Čabrilo, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu je vršena predikcija cijene Airbnb smještaja upotrebom algoritama mašinskog učenja. Podaci su preuzeti sa veb-stranice insideairbnb.com. Predikcija cijene smještaja je vršena na osnovu vrijednosti 62 atributa, koji opisuju smještaj, i na osnovu sentimenta korisničkih recenzija. Svaka korisnička recenzija je sentimentalno analizirana, a zatim je određena prosječna vrijednost sentimenta recenzija za svaki smještaj. Primijenjeno je više tehnika eksplorativne analize podataka i više algoritama za selekciju konačnog skupa atributa. Predikcija cijene smještaja je vršena i regresionim i klasifikacionim algoritmima. Korišteni su sledeći algoritmi: linearna regresija, LASSO regresija, ridge regresija, regresija potpornih vektora, Naive Bayes klasifikacija, Random Forest klasifikacija i SVM klasifikacija.*

Ključne reči: *Mašinsko učenje, Airbnb, Predikcija cijene, Regresija, Klasifikacija*

Abstract – *In this paper, the price of Airbnb accommodation was predicted using multiple machine learning algorithms. The dataset was downloaded from the insideairbnb.com website. The price prediction was based on the values of the 62 attributes, which describe the accommodation, and on the sentiment of the user reviews. Sentiment of the each user review was calculated and the average value of the review sentiment was determined for each accommodation. Multiple exploratory data analysis techniques and feature selection algorithms were applied. Both regression and classification algorithms were used. Following algorithms were selected: linear regression, LASSO regression, ridge regression, support vector regression, Naive Bayes classification, Random Forest classification SVM classification.*

Keywords: *Machine learning, Airbnb, Price prediction, Regression, Classification*

1. UVOD

Airbnb je jedna od najpopularnijih internet platformi za kratkoročno iznajmljivanje smještaja. Airbnb se ponaša kao posrednik između vlasnika i korisnika smještaja i omogućava brzo i jednostavno rezervisanje privatnih smještaja.

Vrijednost Airbnba je 2017. godine procijenjena na 31 milijardu dolara, a na njemu se nalaze oglasi za preko 5 miliona smještaja iz preko 190 zemalja.

Vlasnici definišu cijenu smještaja samostalno, a Airbnb nudi samo smjernice za kreiranje cijene u odnosu na lokaciju na kojoj se smještaj nalazi. Broj smještaja na Airbnbu raste svakodnevno, pa je za vlasnike veoma važno da odrede realnu cijenu smještaja, da bi ostali konkurentni na tržištu. Sa druge strane, korisnici moraju da utvrde da li je cijena smještaja prihvatljiva samo na osnovu podataka koje je vlasnik postavio. Cilj ovog rada jeste da se razvije pouzdan model za predikciju cijene smještaja upotrebom mašinskog učenja i sentimentalne analize, koji bi pomogao i vlasnicima i korisnicima smještaja. Upotrebom modela za predikciju cijene korisnici bi bili sigurni da je ponuđena cijena adekvatna u odnosu na kvalitet i lokaciju smještaja, a vlasnici bi bili sigurni da njihov smještaj nije potcijenjen.

Ovaj rad se sastoji od 6 poglavlja. U drugom poglavlju su navedeni prethodni radovi, slične tematike, i prodiskutovani njihovi rezultati. U trećem poglavlju je opisan skup podataka i koraci koji su sprovedeni u fazi eksplorativne analize podataka. U četvrtom poglavlju su navedeni korišteni algoritmi mašinskog učenja. U petom poglavlju su prikazani i diskutovani rezultati koje su postigli regresioni i klasifikacioni modeli. Šesto poglavlje predstavlja zaključak rada u kome su dati i mogući pravci daljeg razvoja.

2. PRETHODNA RJEŠENJA

U radu [1] autori Nikolenko, Rezaei i Rezazadeh su vršili predikciju cijene Airbnb smještaja koji se nalaze u Njujorku. Autori su smatrali da recenzije korisnika imaju veliki uticaj na cijenu smještaja. Izvršili su sentimentalnu analizu svih recenzija i za svaki smještaj su izračunali prosječnu vrijednost sentimenta recenzija. Primijenjeno je više algoritama mašinskog učenja: ridge regresija, K-means klasterovanje sa ridge regresijom, SVR regresija sa RBF kernelom, neuronska mreža i Gradien Boosting Tree regresija. Model SVR regresije je imao najmanju srednju apsolutnu grešku, najmanju srednju kvadratnu grešku i najveću R^2 mjeru.

U radu [2] autori Tang i Sangani su vršili predikciju cijene Airbnb smještaja lociranih u San Francisku i predikciju gradske četvrti u kojoj se smještaj nalazi. Za predikciju i cijene i gradske četvrti je korišten SVM klasifikator. U odnosu na cijenu, smještaji su podijeljeni u dvije grupe, jeftiniju i skuplju, medijanom cijene. Model za predikciju cijene je postigao tačnost od 81.7%,

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

a model za predikciju gradske četvrti je postigao tačnost od 42.2%. Autori su ustanovili da na tačnost modela za predikciju cijene najviše utiču atributi izvučeni iz oglasa, kao što su broj soba, broj kreveta itd. Na tačnost modela za predikciju gradske četvrti su najviše uticali atributi izvučeni iz opisa smještaja.

U radu [3] autori Yu i Wu su vršili predikciju cijene nekretnina. Koristili su relativno mali skup podataka od 1460 primjera, koji je sadržao vrijednosti 79 atributa za svaku od nekretnina. Predikciju su vršili upotrebom regresionih i klasifikacionih modela.

Od regresionih modela su koristili: LASSO regresiju, ridge regresiju, SVM regresiju i Random Forest regresiju, a od klasifikacionih modela su koristili: Naive Bayes, logističku regresiju, SVM klasifikaciju i Random Forest klasifikaciju.

Da bi smanjili dimenzionalnost skupa podataka autori su primijenili analizu glavnih komponenti. Analiza glavnih komponenti je generalno poboljšala performanse modela. Najveću tačnost od 69% kod klasifikacionih modela je postigao SVM model sa linearnim jezgrom, dok je kod regresionih modela najmanji korijen srednje kvadratne greške od 0.5269 imao model SVR regresije.

3. SKUP PODATAKA

Skup podataka preuzet je sa veb-stranice insideairbnb.com i sastoji se od dvije CSV datoteke, listings.csv i reviews.csv. Datoteka listings.csv sadrži podatke o 20.026 smještaja.

Svaki smještaj je opisan vrijednostima 62 atributa. Datoteka reviews.csv sadrži recenzije smještaja. Za svaku recenziju, pored sadržaja, postoji i informacija koja govori kome smještaju pripada.

3.1. Sentimentalna analiza

Uzimajući u obzir uticaj korisničkih recenzija na cijenu smještaja, a sa ciljem poboljšanja performansi prediktivnih modela, izvršena je sentimentalna analiza recenzija za svaki smještaj. Analiza je vršena pomoću Python programskog jezika i TextBlob biblioteke. Svakoj recenziji je dodijeljena vrijednost iz opsega [-1,1]. Gdje -1 označava izrazito negativnu recenziju, a 1 izrazito pozitivnu recenziju. Određivanje sentimenta TextBlob bibliotekom je moguće za tekstove pisane na francuskom, engleskom i njemačkom jeziku, dok je prepoznavanje jezika omogućeno za veći broj jezika. Da bi se ubrzao proces sentimentalne analize i da bi se odstranile recenzije pisane na jezicima za koje nije moguće određivati sentiment upotrebom TextBlob biblioteke izvršeno je filtriranje recenzija. Filtriranje je vršeno regularnim izrazima, a zadržane su recenzije koje sadrže samo standardne karaktere francuskog, engleskog i njemačkog jezika. Nakon filtriranja otpalo je 25.6% recenzija, odnosno od početnih 483.603 recenzija ostalo je 367.401 recenzija. Zatim je za svaku recenziju izvršeno određivanje jezika i na osnovu toga određivanje vrijednosti sentimenta.

Skup atributa, koji opisuje smještaj, je proširen novim atributom čija vrijednost predstavlja prosječan sentiment recenzija smještaja. Za svaki smještaj je izračunata prosječna vrijednost sentimenta recenzija i ta vrijednost je dodata u skup vrijednosti koji opisuje dati smještaj. Za smještaje koji nisu imali ni jednu recenziju usvojeno je da je prosječna vrijednost sentimenta recenzija 0.

3.2. Eksplorativna analiza

U prvom koraku eksplorativne analize podataka su uklonjeni svi očigledno nebitni atributi kao što su: last_review_id (jedinstvena oznaka posljednje recenzije), first_review_id (jedinstvena oznaka prve recenzije), host_id (jedinstvena oznaka vlasnika smještaja) itd. Pošto atributi zipcode (poštanski broj) i neighbourhood (gradska oblast) određuju približno istu geografsku površinu odlučeno je da se zadrži samo atribut neighbourhood.

Zatim je riješen problem nedostajućih vrijednosti. Za atribute security_deposit (sigurnosni depozit) i cleaning_fee (naknada za čišćenje) nedostajuće vrijednosti sa zamijenjene nulama. Pretpostavka je da ukoliko vlasnik ne navede ove vrijednosti da se one ne naplaćuju korisniku smještaja. Niz atributa review_score_clealiness, review_score_accuracy, review_score_location i review_score_communication predstavljaju prosječnu ocjenu korisnika za čistoću smještaja, preciznost podataka iz oglasa, lokaciju smještaja i komunikativnost vlasnika, respektivno. Niska vrijednost bilo koga od ovih atributa bi uticala negativno na cijenu smještaja, s toga je odlučeno da svi smještaji koji imaju nedostajuće vrijednosti za navedene atribute budu uklonjeni iz skupa podataka. Isti princip je primijenjen i za atribute koji opisuju broj soba, broj kupatila i broj kreveta u smještaju. Vrijednost atributa host_response_rate (označava brzinu kojom vlasnik odgovara na upite korisnika) je nedostajala u 43% slučajeva, s toga je atribut host_response_rate u potpunosti izbačen iz skupa atributa.

Kako bi se utvrdilo da li postoje atributi koji su u jakoj korelaciji izvršeno je računanje matrice korelacije. Jaka korelacija između atributa je indikator da jedan od koreliranih atributa treba ukloniti. Ustanovljeno je da su atributi availability_30, availability_60, availability_90, koji označavaju raspoloživost smještaja u periodu od 30, 60 i 90 dana, su visoko korelirani. Odlučeno je da bude zadržan samo atribut availability_60.

Nakon rješavanja problema nedostajućih vrijednosti i koreliranih atributa izvršena je transformacija nominalnih atributa u numeričke. Korišten je dummy encoding pristup. Takođe, izvršena je i transformacija atributa sa binominalnim vrijednostima u numeričke atribute sa skupom vrijednosti {0,1}.

Istraživanjem vrijednosti ciljnog atributa ustanovljeno je da on nema normalnu raspodjelu i da je raspodjela pomjerena ka vrijednosti od 110€. S obzirom da regresija pretpostavlja da su vrijednosti prediktora normalno raspodijeljene, izvršena je logaritamska transformacija cijene smještaja. Logaritamska transformacija gura vrijednosti atributa na desno čime se postiže raspodjela sličnija normalnoj raspodjeli. Pored cijene smještaja, udesno su bile pomjerene i vrijednosti atributa security_deposit i cleaning_fee. I na njih je primijenjena logaritamska transformacija.

Nakon izvršene eksplorativne analize od početna 63 atributa dobijena su 92 atributa. Uklonjeno je 30 početnih atributa, ali se ukupan broj atributa povećao zbog upotrebe dummy encoding tehnike. Ukupan broj smještaja ja smanjen na 17.725 sa početnih 20.026.

Dobijeni skup podataka je podijeljen na tri skupa trening skup (80% početnog skupa), validacioni skup i test skup (po 10% početnog skupa).

3.3. Selekcija konačnog skupa atributa

Selekcija konačnog skupa atributa je vršena upotrebom: selekcije unaprijed, selekcije unazad i selekcije na osnovu *p* vrijednosti modela linearne regresije. Unutar iteracija selekcije unaprijed i unazad skupovi atributa su poređeni R^2 mjerom modela linearne regresije. Selekcijom na osnovu *p* vrijednosti modela linearne regresije su odabrana 34 atributa sa najmanjim *p* vrijednostima.

Dobijeni skupovi atributa su međusobno upoređeni R^2 mjerom modela linearne regresije treniranog na validacionom skupu podataka. U tabeli 1 su prikazane R^2 mjere svih skupova atributa, uključujući i početni skup.

Tabela 1. *Rezultati algoritama za selekciju konačnog skupa atributa*

Skup atributa	R^2 mjera
Početni skup	0.508
Selekcija unaprijed	0.502
Selekcija unazad	0.498
p vrijednosti	0.553

4. METODOLOGIJA

Predikcija cijene smještaja je vršena i klasifikacionim i regresionim algoritmima. Odabrani su sledeći algoritmi:

- Random Forest klasifikacija,
- Naive Bayes klasifikacija,
- SVM klasifikacija sa RBF kernelom,
- linearna regresija,
- LASSO regresija,
- ridge regresija i
- SVR regresija sa RBF kernelom.

Srednja apsolutna greška (eng. Mean Absolute Error - MAE), srednja kvadratna greška (eng. Mean Squared Error - MSE) i R^2 mjera su korištene za evaluaciju regresionih modela. Za evaluaciju klasifikacionih modela korištene su sledeće mjere: tačnost, preciznost, odziv i F mjera.

Da bi se što više poboljšale performanse prediktivnih modela izvršena je optimizacija hiperparametara. Za ridge i LASSO regresiju optimizovan je regularizacioni hiperparametar *C*. Za SVR regresiju izvršena je optimizacija hiperparametra RBF kernela, kao i optimizacija hiperparametra *C*. Kod Random Forest modela optimizovana je maksimalna dubina stabla, a kod SVR regresije je izvršena optimizacija hiperparametara RBF kernela i hiperparametra *C*. Optimizacija hiperparametara je vršena nad validacionim skupom podataka.

Da bi se izvršilo obučavanje klasifikacionih modela bilo je neophodno klasifikovati podatke koji su dobijeni nakon pretprocesiranja i eksplorativne analize. Cijena smještaja određuje klasu kojoj smještaj pripada. Broj klasa i opsezi cijena koji ih definišu su određeni uzimajući u obzir iskustva iz prethodnih radova i potrebe korisnika. Povećanje opsega cijena i smanjenje broja kategorija bi dovelo do poboljšanje performansi svakog od modela. Međutim, takva predikcije bi bila veoma gruba i ne bi imala značaj za krajnjeg korisnika. Definisano je 9 cjenovnih klasa i to:

- od 20€ do 80€,
- od 80€ do 110€,
- od 110€ do 130€,
- od 130€ do 150€,
- od 170€ do 210€,
- od 210€ do 250€,
- od 250€ do 300€ i
- preko 300€.

5. ANALIZA REZULTATA I DISKUSIJA

Prikaz i analiza rezultata će biti podijeljeni u dvije cjeline. Prva cjelina se odnosi na rezultate dobijene upotrebom regresionih modela, a druga cjelina se odnosi na rezultate dobijene upotrebom klasifikacionih modela za predikciju cijene smještaja. Implementacija svih navedenih modela je vršena upotrebom RapidMiner alata.

5.1. Analiza rezultata regresionih modela

U tabeli 2 su prikazani rezultati regresionih modela. Svi regresioni modeli su imali relativno slične performanse iz čega zaključujemo da je selekcije atributa na osnovu *p* vrijednosti imala najviše uticaja na poboljšanje performansi. LASSO i ridge regresija nisu imali značajno bolje rezultate od linearne regresije, što znači da je logaritmovanje cijene smještaja uspješno umanjilo uticaj izuzetaka.

Tabela 2. *Rezultati regresionih modela*

	MAE	MSE	R^2
linearna regresija	0.237	0.114	0.549
LASSO regresija	0.213	0.107	0.565
ridge regresija	0.216	0.1042	0.566
SVR regresija	0.201	0.099	0.612

Od svih regresionih modela model SVR regresije se pokazao najbolje. Imao je najmanju srednju apsolutnu grešku, najmanju srednju kvadratnu grešku i najveću R^2 mjeru.

SVR model je najbolje uspio da modeluje nelinearne zavisnosti cijene smještaja i ostalih atributa smještaja. Primjenom optimizacije hiperparametra je uspješno spriječeno pretjerano prilagođavanje obučavajućim podacima.

5.2. Analiza rezultata klasifikacionih modela

Klasifikacioni modeli su prvo obučeni i evaluirani na skupu podataka sa atributima koji su dobijeni izborom prema *p* vrijednostima. Pošto je analiza glavnih komponenti (eng. Principal Component Anlysis – PCA) značajno poboljšala performanse klasifikacionih modela u radu [3], odlučeno je da se ona primjeni i u ovom radu.

Na skup atributa, dobijen izborom prema *p* vrijednostima, je primijenjena PCA, a zatim su obučeni modeli i evaluirani. Tabela 3 prikazuje performanse modela sa i bez PCA.

Tabela 3. *Performanse klasifikacionih modela*

	Naive Bayes	SVM	Random Forest
Tačnost prije PCA	0.39	0.44	0.58
Tačnost posle PCA	0.47	0.51	0.52

Primjena PCA je dovela do poboljšanja tačnosti kod Naive Bayes i SVM klasifikatora, a do opadanja kod Random Forest klasifikatora. U tabeli 4 prikazani su rezultati Random Forest modela bez analize glavnih komponenti.

Tabela 4. *Performanse Random Forest modela*

	Preciznost	Odziv	F mjera
[0€-80€]	0.55	0.62	0.58
[80€-110€]	0.66	0.51	0.57
[110€-130€]	0.42	0.49	0.45
[130€-150€]	0.38	0.67	0.48
[150€-170€]	0.57	0.45	0.50
[170€-210€]	0.38	0.46	0.41
[210€-250€]	0.52	0.50	0.51
[250€-300€]	0.44	0.32	0.37
300€ +	0.25	0.30	0.27

Iz tabele 4 vidimo da je Random Forest model najbolje performanse ostvario za prve dvije klase, a najlošije za posljednje dvije klase. Ukoliko uporedimo dobijene rezultate sa brojem smještaja klasa zaključujemo da su performanse modela u odnosu na klasu korelirane sa brojem smještaja u klasi. Što je manji broj smještaja u klasi to su lošije performanse modela i obratno. Za bolje diferenciranje klasi sa sličnim brojem smještaja bilo bi potrebno proširiti skup atributa sa atributima koji bolje i detaljnije opisuju smještaje (detaljniji opis namještaja i pogodnosti).

6. ZAKLJUČAK

U ovom radu obučen je niz regresionih i klasifikacionih modela za predikciju cijene Airbnb smještaja. Podaci su preuzeti sa veb-stranice insideairbnb.com. Podaci uključuju karakteristike smještaja (broj soba, broj kreveta itd), lokaciju smještaja i recenzije prethodnih korisnika.

Određivanje sentimenta recenzija je vršeno upotrebom TextBlob biblioteke. Tako dobijeni podaci su spojeni sa osnovnim skupom podataka, a zatim je izvršena eksplorativna analiza. Izbačeni su nepotrebni atributi, riješen je problem nedostajućih vrijednosti, uklonjeni su kolinearni atributi, izvršene su transformacije ciljnog atributa i transformacije nominalnih atributa u numeričke. Zatim su primijenjene tehnike za selekciju skupa atributa i to: selekcija unazad, selekcija unaprijed i selekcija na osnovu *p* vrijednosti. Najbolje se pokazao skup atributa odabran na osnovu *p* vrijednosti.

Od regresionih modela najbolje se pokazao model SVR regresije koji je imao R^2 mjeru od 0.612, srednju apsolutnu grešku od 0.201 i srednju kvadratnu grešku od 0.099. Najbolje performanse, kod klasifikacionih modela, je imao Random Forest klasifikator sa tačnošću od 0.58.

Planovi za dalji razvoj projekta:

- dobavljanje podataka o dodatnim uslugama i pogodnostima smještaja (internet, klima, doručak itd.) i njihovo uključivanje u postojeći skup podataka,
- analiziranje slika iz oglasa kako bi se utvrdila opremljenost smještaja i integracija tako dobijenih podataka u postojeći skup podataka,
- proširivanje skupa podatka udaljenostima smještaja od značajnih turističkih atrakcija i
- dobavljanje podataka o stopi kriminala, bezbjednosti i sigurnosti gradskih oblasti i uključivanje tih podataka u postojeći skup podataka.

7. LITERATURA

[1] L. Nikolenko, H. Rezaei, P. Rezazadeh, „Airbnb Price Prediction Using Machine Learning and Sentiment Analysis", Stanford, 2019

[2] E. Tankg, K. Sangani, „Neighborhood and Price Prediction for San Francisco Airbnb Listings ", Stanford, 2015

[3] H. Yu, J, Wu, „Real Estate Price Prediction with Regression and Classification", Stanford, 2016

Kratka biografija:

Miljan Čabrilo rođen je u Nevesinju 1996. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranio je 2020. godine.

Kontakt:
miljancabrilo@yahoo.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.29
DOI: https://doi.org/10.24867/12BE38Vujovic

INSPEKCIJA UPOTREBLJIVOSTI MOBILNIH APLIKACIJA ZA OČUVANJE ZDRAVLJA

USABILITY INSPECTION OF MOBILE HEALTH APPS

Nemanja Vujović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisan je problem inspekcije upotrebljivosti različitih mobilnih aplikacija za očuvanje zdravlja. Metod korišćen za ispitivanje je kognitivni prolazak, a na kraju je urađeno sumiranje dobijenih rezultata.*

Ključne reči: *Inspekcija, analiza, zdravlje, aplikacija*

Abstract – *This paper describes the problem of usability inspection of different Mobile Health apps. Examination is done by using Cognitive walkthrough method. In the end, a summary of obtained results is included in the paper.*

Keywords: *Inspection, analysis, health, application*

1. UVOD

Zadatak ovog rada jeste inspekcija upotrebljivosti mobilnih aplikacija za očuvanje zdravlja. Pod samom inspekcijom podrazumeva se temeljna analiza zasebnih funkcionalnosti health aplikacija i poređenje dobijenih rezultata. Samo određen broj funkcionalnosti će biti ispitan, a izbor će se vršiti na osnovu njihove važnosti, kao i na osnovu toga koliko se često koriste u odnosu na ostale funkcionalnosti u aplikaciji. To zapravo znači da će one najfrekventnije biti ispitane, dok one manje važne neće.

Funkcionalnosti koje će biti podvrgnute inspekciji su: izbor aktivnosti i načina vežbanja, povezivanje sa drugim uređajima i dnevni unos kalorija. Tehnika koja će biti korišćena za analizu jeste Cognitive walkthrough, odnosno kognitivni prolazak. Konačni rezultati biće prikazani tabelarno uz pomoć tablice kognitivnog prolaska, dok će svaka ispitana funkcionalnost biti potkrepljena odgovarajućom slikom.

Cilj ovog istraživanja jeste upoznavanje sa načinom rada, kao i pronalazak mogućih unapređenja za Mobile Health aplikacije. Kao modeli za inspekciju biće korišćene aplikacije Samsung health i Huawei health. Razlog ovog izbora leži u tome što su date aplikacije najpopularnije i najkvalitetnije na tržištu i kao takve najbolji su primer funkcionisanja mobilnih aplikacija za očuvanje zdravlja. Pored same analize i poređenja dveju aplikacija, odnosno njihovih određenih funkcionalnosti, ovaj rad će se u manjoj meri baviti i opisom tehnika za ispitivanje upotrebljivosti aplikacija, kao i razlozima zbog kojih je u ovom radu izabrana tehnika Cognitive walkthrough.

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

2. MOTIVACIJA I OPIS PROBLEMA

Motivacija za pisanje ovakve vrste rada leži u velikom broju razloga. Jedan od njih jeste sam razvoj tehnologije u poslednjih nekoliko godina. Svaka grana života pokrivena je odgovarajućom tehnologijom. Poseban napredak u poslednje vreme beleži proizvodnja mobilnih telefona, čiji osnovni sadržaj čine android aplikacije.

Jedna vrsta tih aplikacija su i aplikacije za očuvanje zdravlja, čijom analizom će se ovaj rad baviti. Dakle, potreba za stalnim unapređivanjem tehnologije, u ovom slučaju android-a, je glavni motiv pisanja ovog rada. Sam rad bi trebao da ponudi detaljnu analizu ove vrste aplikacija i da ukaže na eventualne nedostatke. Pored toga trebao bi predstaviti određena rešenja problema, ne programskim putem, ali kao neku ideju svakako. Osim napretka tehnologije i potrebe za unapređivanjem aplikacija, motiv za pisanje ove teme je i velika potreba čoveka za zdravljem, ali i za dobrim izgledom.

Veliku pomoć u datim potrebama omogućavaju health aplikacije. Svaki korisnik health-a može na vrlo jednostavan način da isprati rad svoga tela za vreme vežbanja, što je pomoć kakvu čovek ranije nije imao. Dodatne mogućnosti koje ove vrste aplikacija nude su i pomoć pri izboru aktivnosti kao i praćenje unosa kalorija. Korisnik može da izabere jednu od ponuđenih aktivnosti, kao što su: šetanje, trčanje, plivanje ili recimo vožnja biciklom i da na osnovu svoga vežbanja isprati rezultate. Kada je reč o unosu kalorija, aplikacija nudi opciju pravljenja obroka, koji bi trebao da se u potpunosti poklapa sa stvarnim obrokom datog korisnika i da na osnovu vrednosti makronutrijenata, namirnica koje su u taj obrok ubačene, izračuna količinu unešenih kalorija u okviru tog konkretnog obroka. Aplikacije za očuvanje zdravlja nude još jednu veoma važnu i kvalitetnu opciju, a to je uparivanje sa drugim uređajima.

Korisnik može na vrlo jednostavan način da svoju health aplikaciju sa mobilnog telefona upari sa, na primer, Health watch uređajem i tako dobije dodatne opcije, kao što su praćenje otkucaja srca i krvnog pritiska za vreme vežbanja, što su jako važni zdravstveni parametri. Osim ovog postoje još mnogi primeri povezivanja uređaja kod health-a. Neki od njih su fitnes narukvice koje zadaju dnevne zadatke korisnicima, zatim razne vrste holtera za praćenje rada srca, kao i pametne vage, koje daju informaciju o telesnoj težini korisnika, što kasnije omogućava lakše praćenje napretka. Sve ove funkcije dale su motiv za pisanje rada, odnosno dovele su do potrebe da se kvalitet njihove upotrebljivosti i prilagođenosti korisniku ispita, kako bi sami korisnici mogli jednostavnije i kvalitetnije da koriste ovakve aplikacije.

3. METODOLOGIJE

Za ispitivanje kvaliteta aplikacija kao i različitih vrsta sajtova postoji par raznovrsnih metodologija. U ovom radu biće korišćena tehnika Cognitive walkthrough koja se pokazala kao najkvalitetnija u sličnim radovima. Dodatno, u ovom poglavlju rad će se baviti analizom tehnika za ispitivanje kao i razlozima zbog kojih je Cognitive walkthrough najbolji izbor.

3.1. Izbor i analiza metodologija

Metode inspekcije aplikacija pojavile su se kao dodatna verzija metodama testiranja početkom devedesetih godina 20. veka. Koristeći metode inspekcije ocenjivač, u pravilu HCI stručnjak, procenjuje usklađenost interaktivnog sistema s važećim standardima i smernicama upotrebljivosti. Najčešće korišćene metode inspekcije su: evaluacija po heuristikama i kognitivni prolazak.

Evaluacija po heuristikama predstavlja jednu od dve najpoznatije i najkvalitetnije tehnike za inspekciju upotrebljivosti aplikacija. Ona radi po principu pregleda sistema i utvrđivanja dobrih i loših elemenata, odnosno funkcionalnosti istog. Upoređuje se kompletan sistem neke aplikacije sa smernicama upotrebljivosti koje su unapred određene. Tokom sprovođenja postupka heurističkog vrednovanja svaki ocenjivač prolazi kroz elemente sistema najmanje dva puta. Prilikom prvog prolaska samo se upoznaje sa radom aplikacije i njenim mogućnostima. Drugi prolazak služi za ocenjivanje funkcionalnosti i određenih elemenata sistema na osnovu ranije dobijenog spiska heuristika. Na kraju se dobija konačan spisak funkcionalnosti koje u manjoj ili većoj meri nisu ispoštovale heuristike. Postoji deset heuristika spram kojih se vrednuje kvalitet upotrebljivosti aplikacije. Heuristike vrednovanja su [1]: vidljivost statusa sistema, usaglašenost sistema sa realnim svetom, kontrola i sloboda u korišćenju, konzistentnost i standardi, prevencija grešaka, prepoznavanje pre pamćenja, fleksibilnost i efikasnost korišćenja, estetski i minimalistički dizajn, prepoznavanje i oporavak od grešaka i pomoć i dokumentacija. Date heuristike potrebno je ispoštovati u celosti kako bi aplikacija bila na visokom nivou upotrebljivosti. Nakon sprovodjenja tehnike evaluacije po heuristikama korisnik koji testira aplikaciju treba da navede gde se tačno problem pojavio kao i koja konkretno od heuristika je prekršena. Osnovna prednost ovakvog tipa ispitivanja aplikacija jeste mogućnost njene primene u svim fazama razvoja aplikacije, kao i brza i efikasna identifikacija, kako glavnih tako i sporednih problema upotrebljivosti [2].

Pored dve osnovne metode ispitivanja postoji još jedna, manje zastupljena, koja takođe daje dobre rezultate. Formalan pregled upotrebljivosti je formalizovana metoda za pregled pristupa kao i identifikovanje i opisivanje problema upotrebljivosti. Namenjena je timu evaluatora, čiji članovi se stavljaju u ulogu HCI stručnjaka. Pomoću metode ujedno se utvrđuju potencijalne performanse korisnika pri izvođenju zadataka. Metoda ima sedeće karakteristike: utvrđivanje i opis problema upotrebljivosti, veliki tim evaluatora gde svaki ima svoj zadatak i opisuje zaseban deo aplikacije i struktura životnog ciklusa upotrebljivosti, odnosno šest logičkih koraka ispitivanja koji osiguravaju da se identifikacija i opis problema upotrebljivosti obavi na učinkovit i kvalitetan način.

3.2. Cognitive walkthrough

U ovom radu analiza upotrebljivosti Mobile Health aplikacija biće sprovedena metodologijom Cognitive walkthrough ili tehnikom kognitivne šetnje. Metoda se zasniva na teorijskoj osnovi kognitivne teorije učenja kroz istraživanje. To je pristup upotrebljivosti specifičan za zadatak (za razliku od heurističke evaluacije koja je holističnija inspekcija upotrebljivosti).

Ideja počiva na tome da većina korisnika radije uči nove stvari na osnovu praktičnog primera, nego čitajući priručnik ili prateći niz uputstava. Metoda kognitivnog prolaska namenjena je evaluatorima koji simuliraju ponašanje korisnika tokom rešavanja zadataka u sistemu. Sastoji se od dve faze: pripreme i izvođenja. Faza pripreme je navođenje konkretnih koraka koji treba da se ispitaju da bi se utvrdila upotrebljivost aplikacije. Druga faza odnosi se na traženje odgovora na četiri osnovna pitanja tehnike kognitivnog prolaska. Pitanja na koja je potrebno odgovoriti su [3]:

- Da li će korisnik postići očekivani efekat?
- Da li će korisnik uočiti da je očekivana radnja moguća?
- Da li korisnik povezuje adekvatnu akciju sa rezultatom koji želi da postigne?
- Ako je akcija uspešno izvedena, može li korisnik uočiti napredak ka konačnom rešenju zadatka?

Potrebno je da korisnik odgovori detaljno na svako od postavljenih pitanja, kako bi dobio što je moguće preciznije rezultate. Prednosti metode kognitivnog prolaska uključuju delotvornu identifikaciju problema koji proizlaze iz interakcije s aplikacijom i usko su povezani s lakoćom učenja aplikacije, te mogućnosti utvrđivanja korisničkih ciljeva i ponašanja prilikom korišćenja aplikacije. S druge strane, složenost metode i preterana detaljnost u sprovođenju spadaju u glavne nedostatke ove metodologije.

4. INSPEKCIJA APLIKACIJA

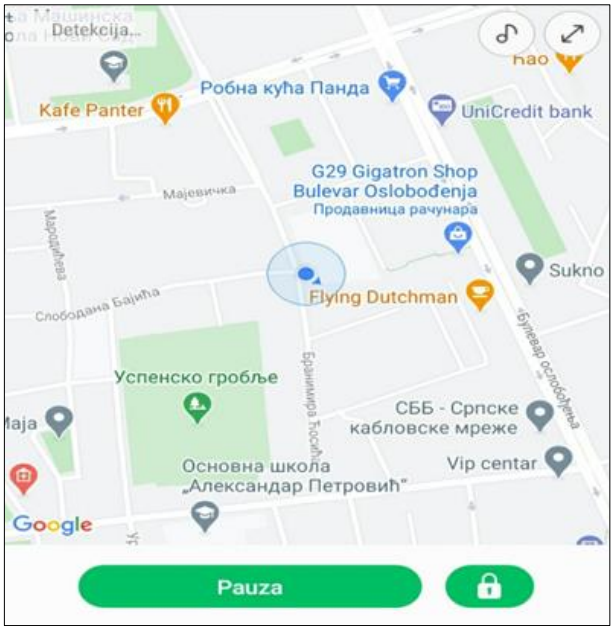
U ovom poglavlju biće sprovedena inspekcija upotrebljivosti funkcionalnosti Samsung i Huawei health aplikacija. Funkcionalnosti koje će biti ispitane su: izbor fizičke aktivnosti kod obe aplikacije, unos kalorija kod Samsung health-a i konekcija sa drugim uređajima kod Huawei health aplikacije.

4.1. Fizička aktivnost

Osnovna i najvažnija funkcionalnost health aplikacija jeste fizička aktivnost. Svrha postojanja ove vrste aplikacija ogleda se u potrebi čoveka za dobrim izgledom i u što većoj meri zdravim životom.

Sama analiza upotrebljivosti za izbor fizičke aktivnosti biće sprovedena kroz obe aplikacije. U slučaju Samsungove ona se sprovodi kroz osam koraka: izbora kartice gde je prikazana željena aktivnost, izbor prikaza, unos vrednosti koju korisnik želi da dostigne, klik na dugme gde piše započni, klik na dugme gde piše pauza, klik na dugme gde piše završi, odlazak na padajući meni gde piše svi pribori, izbor kartice u istoriji aktivnosti koju korisnik želi da pregleda. Svaki od oivh koraka potrebno je analizirati tehnikom kognitivnog prolaska. Odnosno dobiti odgovore na sva četiri osnovna pitanja metodologije Cognitive walkthrough.

Na osnovu detaljne analize utvrđeno je da prva četiri koraka u potpunosti zadovoljavaju uslove postavljene datim pitanjima. Za razliku od njih, peti korak narušava većinu kriterijuma. Korisnik u momentu treniranja nema na početnom ekranu opciju da završi aktivnost, što zapravo narušava kriterijum prva tri pitanja. Korisniku je potrebno prethodno iskustvo sa datom aplikacijom da bi znao da mora prvo da izabere opciju pauze, da bi nakon toga mogao da zaustavi merenje svih parametara u toku treninga. Pored toga na drugo pitanje je nemoguće odgovoriti pozitivno, jer se odmah pored dugmeta za pauzu nalazi dugme za zaključavanje, predstavljeno slikom katanca (slika 1), što korisnika može da dovede u stanje konfuzije i nemogućnosti da razazna koju opciju treba izabrati da bi došao u situaciju da završi aktivnost. Takođe, samim tim što jedna od opcija nije na početnom ekranu već u startu je dati kriterijum narušen. Treće pitanje takođe nije zadovoljeno iz istih razloga kao i prethodna dva.



Slika 1. *Ekran sa komandama za pauzu i zaključavanje*

Nakon analize Samsungove funkcionalnosti za izbor fizičke aktivnosti u okviru ovog poglavlja vrši se i analiza iste funkcionalnosti, samo ovoga puta kod Huawei health-a. Koraci za izvršavanje ove akcije su: izbor taba exercise, odabir željene vrste vežbanja, klik na dugme za pokretanje akcije, klik na dugme za pauzu i zadržavanje dugmeta za prekid akcije uz potvrdu izlaska. Kao i kod Samsung-a, svaki korak potrebno je detaljno analizirati kako bi se utvrdio kvalitet upotrebljivosti.

Da se primetiti da za razliku od prethodno analizirane aplikacije, ova ima manji broj koraka za analizu. Razlog za ovu pojavu leži u tome što u slučaju Huawei health-a funkcionalnosti izbora fizičke aktivnosti i pregleda istorije aktivnosti predstavljaju dve potpuno odvojene opcije. U ovom radu analizirana je samo funkcionalnost izbora fizičke aktivnosti. Nakon njene detaljne analize utvrđeno je da se u dva od pet koraka javljaju odstupanja od kriterijuma upotrebljivosti, ustanovljenog tehnikom kognitivnog prolaska. Prvi korak u potpunosti zadovoljava sva četiri postavljena pitanja. Za razliku od njega u drugom koraku dolazi do kršenja četvrtog kriterijuma upotrebljivosti. Nakon izbora fizičke aktivnosti koju korisnik želi da upražnjava, na ekranu ostaju prikazane i ostale aktivnosti iz ponude, sa tom razlikom da je izabrana vežba podvučena. Ovo ne daje očekivani odgovor na četvrto pitanje, jer povratna informacija o izboru aktivnosti nije urađena na adekvatan način. Korisnik, naročito onaj u starijem životnom dobu, neće sa lakoćom primetiti koju je vežbu izabrao, što može dovesti do nepotrebnog gubljenja vremena za ponovni izbor. Treći korak je najkvalitetnije urađen i dao je pozitivne odgovore na sva pitanja. U slučaju četvrtog koraka javlja se isti problem kao i kod Samsung health aplikacije, da se prvo mora izabrati opcija pauze, da bi se mogla napustiti vežba.

Takođe je prikazana i zbunjujuća opcija zaključavanja kao i još jedna dodatna opcija, takođe nepotrebna. Na posletku javlja se i narušavanje trećeg kriterijuma u okviru poslednjeg koraka. Ono se ogleda u tome da korisnik mora da zadrži dugme za izlazak iz datog ekrana, iako ono izgleda kao obično dugme za klik, uz to da nigde nije naglašeno da je dugme potrebno duže zadržati. Ovo daje negativan odgovor na treće pitanje, jer korisnik koji dođe do željenog ekrana i koji želi da zaustavi akciju koja traje, može da dođe u situaciju da ne zna na koji način to da izvrši. Na slici 2 prikazan je ekran sa dugmetom za izlazak, koje je potrebno zadržati, kao i dugmetom za eventualni nastavak akcije.

Slika 2. *Ekran sa komandama za nastavak i prekid akcije*

4.2. Praćenje kalorijskog unosa

Praćenje kalorijskog unosa je funkcionalnost koja ne postoji u okviru Huawei health aplikacije, odnosno to je funkcionalnost za koju će se sprovesti analiza isključivo Samsung health-a. Kao što je fizička aktivnost glavna karakteristika health aplikacija kada je potrošnja kalorija u pitanju, tako je unos hrane glavna kada su unešene kalorije u pitanju. Kada je reč o unosu obroka u okviru Samsung health-a, postupak se odvija iterativno na sledeći način: klik na sekciju sa izborom hrane, odabir tipa obroka, unos naziva željene hrane u pretragu, klik na željenu hranu i klik na dugme na kome piše gotovo.

Ovaj postupak zapravo predstavlja unos pojedinačnih namirnica, odnosno neće se njime samim po sebi dobiti celokupan obrok. Međutim, sam postupak unosa obroka je iterativni postupak unosa namirnica, tako da nije bilo potrebe objašnjavati isti algoritam nebrojeno puta.

Kada je reč o samoj analizi porebno je istaći da se ovaj postupak sastoji se iz pet koraka. Velikim delom daje pozitivne odgovore na pitanja ustanovljena tehnikom kognitivnog prolaska. Jedino u okviru drugog koraka dolazi do narušavanja drugog kriterijuma. Razlog za to je taj što su u okviru prikaza obroka presdtavljene tri vrste užine, iako je dovoljna i samo jedna, a dodatnu konfuziju unosi činjenica da su sve prikazane istom sličicom.

4.3. Veza sa drugim uređajima za očuvanje zdravlja

U ovom poglavlju biće izvršena inspekcija upotrebljivosti funkcionalnosti za uparivanje sa drugim uređajima. Povezanost sa drugim uređajima za očuvanje zdravlja predstavlja funkcionalnost koju ima Huawei health aplikacija, dok Samsung health nema. Odnosno Samsung-ovi uređaji se povezuju međusobno na potpuno drugačiji način, koji nije tema ovog rada.

Koraci za povezivanje sa drugim uređajima su sledeći: izbor taba devices, odabir opcije za dodavanje novog uređaja, izbor vrste uređaja za dodavanje, klik na željeni uređaj uz potvrdu izbora.

Kompletan postupak sastoji se iz četiri koraka i veoma je jednostavan. Svaki korak u potpunosti zadovoljava sve kriterijume tehnike Cognitive walkthrough, zbog čega detaljnija analiza ove funkcionalnosti nije potrebna.

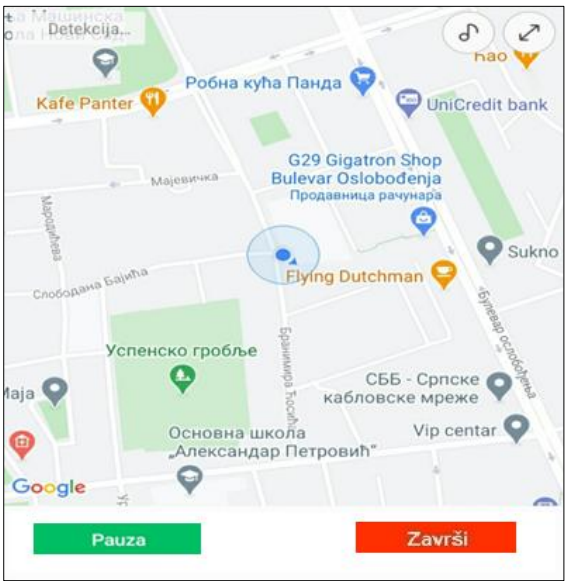
5. DISKUSIJA I POREĐENJE REZULTATA

Nakon detaljne analize svih funkcionalnosti Samsung i Huawei health aplikacija, potrebno je prikazati rezultate, diskutovati određena poboljšanja, kao i u ovom slučaju, koji to dozvoljava, uporediti slične ili iste funkcionalnosti i njihove rezultate.

Analiza je pokazala da su funkcionalnosti izbora fizičke aktivnosti dveju aplikacija veoma slične, kao i da se njihove razlike oslikavaju u finesama. Jedina razlika koja pravi prednost u korist Samsung-ove aplikacije jeste ta što se kod Huawei health-a pri prekidu aktivnosti mora zadržati dugme duže, što narušava treći kriterijum kognitivnog prolaska. Prednost kod Huawei-ja je ta što je istorija aktivnosti zasebna funkcionalnost, pa se korisnik može lakše snaći pri izboru. Sve ostalo se uglavnom poklapa.

Pored poređenja dveju istih funkcionalnosti, različitih aplikacija, u ovom poglavlju je potrebno i dati određena rešenja problema. Za sve navedene nedostatke moguće je na vrlo jednostavan način pronaći rešenja.

Za izbor aktivnosti kod Samsung-a, rešenje bi moglo da predstavlja dodavanje dugmeta za nasilno prekidanje akcije na početni ekran (slika 3).



Slika 3. *Ekran sa mogućim rešenjem za prekid akcije*

Isti je slučaj i kod Huawei health-a. Jedina razlika je u tome što je ovde potrebno pronaći rešenje i za dugme za prekid akcije. Konkretno rešenje je vrlo jednostavno i intuitivno i ogleda se u tome da se treba promeniti zadržavanje dugmeta u klik.

Na posletku problem kod unosa kalorija je takođe vrlo lako rešiv. Postoje dva moguća rešenja. Prvo je da se dve užine obrišu i ostavi se mogućnost izbora samo jedne. Drugo, manje kvalitetno, je da se svaka užina predstavi različitom slikom i da se tako spreči mogućnost da korisnik dođe u stanje konfuzije.

6. ZAKLJUČAK

U radu je opisan problem analize dveju health aplikacija i poređenja njihovih rezultata, pomoću metodologije Cognitive walkthrough.

Glavni razlog izbora ove teme jeste taj što su health aplikacije široko primenjive u današnjem svetu. Pored toga, veoma su popularne među ljudima koji se aktivno bave treningom, ali i onima koji veliku pažnju posvećuju svom zdravlju.

Analiza samih aplikacija sprovedena je tehnikom kognitivnog prolaska. Ova tehnika se oslanja na lično iskustvo evaluatora i kao takva je mnogo popularnija od drugih metodologija. Svaka funkcionalnost ispitana je na osnovu četiri osnovna pitanja ove tehnike.

Funkcionalnosti ispitivane u okviru ovog rada su: izbor aktivnosti i načina vežbanja kod obe aplikacije, unos kalorija kod Samsung health aplikacije i veza sa drugim uređajima kod Huawei health aplikacije.

Kao i svaki rad ovog tipa i ovaj ima određene nedostatke i prostor za unapređenje. Tu se pre svega misli na mali broj ispitanih funkcionalnosti, ali i na prisustvo samo jednog evaluatora. Savet za buduća unapređenja je da se ispitaju funkcionalnosti praćenja telesne težine, ali i izmene profila, koje su zajedničke za obe aplikacije. Pored toga bilo bi poželjno da se broj evaluatora poveća makar na tri, kao i da se ispita bar još jedna aplikacija.

7. LITERATURA

[1] Jakob Nielsen i Rolf Molich „Heuristic evaluation of user interfaces“, april 1990.

[2] Jakob Nielsen „ Usability inspection methods”, april 1994.

[3] Cognitive walkthrough, https://medium.c/evaluation-of-a-user-interface-using-cognitive-walkthrough-real-case-ac94014003d8

Kratka biografija:

Nemanja Vujović rođen je 1996. godine u Novom Sadu. Završio je gimnaziju „Jovan Jovanović Zmaj“ takođe u Novom Sadu 2015. godine. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu 2019. godine i iste godine je upisao Master studije na smeru Računarstvo i automatika.

kontakt: vujovicnemanja670@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.39
DOI: https://doi.org/10.24867/12BE39Zoric

ЕВАЛУАЦИЈА УПОТРЕБЉИВОСТИ ИКОМЕРЦ АПЛИКАЦИЈА НА МОБИЛНИМ УРЕЂАЈИМА

USABILITY EVALUATION OF ECOMMERCE MOBILE APPLICATIONS

Марко Зорић, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *У овом раду је разматрана употребљивост Амазон, АлиЕкспрес и Банггуд икомерц мобилних апликација. Употребљивост је разматрана кроз хеуристике за развој доброг корисничког интерфејса, конкретно Шнајдерманових осам златних правила. Све три апликације су разматране кроз свих осам хеуристика и истакнута су подударања и мимоилажења са хеуристикама и дате на крају додатне смернице како би се та мимоилажења могла исправити.*

Кључне речи: *HCI хеуристике, икомерц, мобилне апликације*

Abstract – *In this paper was evaluated usability of Amazon, AliExpress and Banggud e-commerce mobile applications. Usability was evaluated through heuristics for development of good user interface, specifically Schneiderman's eight golden rules. All three applications were evaluated through all eight heuristics and all matches and diversions of user interface with heuristics were pointed out and at the end additional guidelines for how those divergencies could be rectified were added.*

Keywords: *HCI heuristics, e-commerce, mobile applications*

1. УВОД

Развој мобилних телефона и њиховог рапидног усвајања од стране становништва као алатке за свакодневно коришћење представља један од најважнијих аспеката технолошког развоја с краја 20. и почетка 21. века. Иако су први мобилни телефони били управо и искључиво то – телефони, врло брзо се дошло до закључака да се мобилни телефони могу користити за много више него пуког телефонирања, тако да су мобилним телефонима убрзо додаване нове функционалности попут *SMS* порука, камера, видео игара и повезивања са интернетом.

Велики корак у развоју мобилних телефона јесте појава паметних телефона, односно *smartphone*-а, а поготово појава *Apple*-овог *iPhone*-а који је био први телефон који је у потпуности уклонио физичку тастатуру са телефона и ослањао се искључиво на виртуелну тастатуру исцртану на самом екрану. Такође, *iPhone* је први телефон који је омогућио рендеровање целих веб страница на телефон помоћу свог интернет претраживача Сафарија, док су се до тада странице морале посебно форматирати коришћењем технологија попут *WML*, *cHTML* или *XHTML* [1]. *iPhone* је, такође, омогућио и бесплатне алате за креирање third-party апликација које су се могле скинути са *App Store*-а [2].

Ове иновације су биле изузетно популарне код корисника и убрзо су усвојене од свих осталих произвођача паметних телефона, и тако је мобилни телефон све више наличио малом персоналном рачунару него телефону. Убрзо су се појавиле многе third-party апликације, међутим уникатни проблеми са приказивањем информација на мобилним телефонима нису нестали, већ су постали израженији.

Највећи проблем мобилних телефона јесте њихова величина, односно прецизније, величина њиховог екрана. Мобилни телефон, као што му и само име каже, је преносив уређај који треба да стане у џеп корисника и кога око половине корисника преферира да користи једном руком [3]. Самим тим, постоји одређена граница величине телефона пре него што његово коришћење постане непрактично. Појавом модерних паметних телефона корисници су почели све више да се ослањају на њихово коришћење за обављање многих својих свакодневних потреба, тако је проблем мобилних телефона, који је и до тада постојао, постао још значајнији: како на релативно малом екрану приказати потенцијално велику количину информација, а да све то буде лако приступачно и интуитивно кориснику.

2. ХЕУРИСТИКЕ

HCI (*Human-computer interaction*) представља мулти-дисциплинарну област која се бави дизајном, евалуацијом и имплементацијом интерактивних компјутерских система за људско коришћење и са проучавањем главних феномена које окружују те системе [4]. Једна од кључних аспеката *HCI*-ја јесте израда корисничких интерфејса. *HCI* хеуристике представљају скуп смерница помоћу којих се остварује довољно добро израђен кориснички интерфејс. Придржавање хеуристикама не гарантује стварање доброг интерфејса, али одступање од њих гарантује стварање лошег.

Пошто хеуристика обухвата методе и технике решавања проблема базираних на искуству [5], не постоји свеопште прихваћен скуп хеуристика, већ су

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Драган Иветић, ред. проф.

различити научници излагали своје скупове. У овом раду су били коришћени тзв. „Осам златних правила“ [6] које је први пут изнео професор компјутерских наука универзитета у Мериленду Бен Шнајдерман 1997. године у својој књизи „Designing the User Interface: Strategies for Effective Human-Computer Interaction“. Његових осам златних правила су:

1. тежити за конзистентношћу
2. тежити универзалној употребљивости
3. омогућити информативне повратне информације
4. дизајнирање дијалога наглашене затворености
5. спречити грешке
6. омогућити лако поништавање акција
7. држати корисника у контроли
8. смањити преоптерећеност меморије

Тежити ка конзистентношћу подразумева да стил појединачног елемента треба да буде конзистентан кроз цео интерфејс. То подразумева коришћење истих боја, иконица, фонтова, величина слова и слично кроз цео интерфејс. Ни један елемент интерфејса не би требао да штрчи у односу на елемент истог типа [6].

Тежити универзалној употребљивости подразумева да треба да се препознају потребе различитих типова корисника за представљање информација и управљањем апликацијом. Приликом дизајнирања интерфејса неопходно је имати у виду и кориснике који нису упознати са апликацијом, као и оне који је већ дуже време користе. Такође, треба водити рачуна о корисницима различитог старосног доба, технолошке писмености и корисницима са неком врстом инвалидитета [6].

Омогућити информативне повратне информације подразумева да за сваку акцију корисника је неопходно да постоји повратна информација на интерфејсу и да та повратна информација треба да буде јасна и недвосмислена. За мање или честе акције повратна информација може бити мала и не мора бити претерано наглашена, док за веће и ређе акције би требала бити посебно наглашена [6].

Дизајнирање дијалога наглашене затворености подразумева да секвенце акција треба да буду организоване у групе које имају почетак, средину и крај. Организовање секвенци акција на овај начин изазива код корисника осећај задовољства након завршетка секвенце и припрема их за следећу секвенцу [6].

Спречити грешке подразумева да се интерфејс дизајнира тако да се у што већој мери спречи стварање грешака корисника приликом коришћења апликације. Један прост пример би био да се забрани уношење слова у текстуално поље у којем треба да буду само бројеви. Уколико корисник направи грешку, неопходно је да се кориснику укаже где је направио грешку и понуди јасно објашњење како да грешку отклони [6].

Омогућити лако поништавање акција подразумева да, колико год је то могуће, корисницима треба омогућити поништавање претходно извршених акција. Ово смањује или уклања нервозу корисника приликом коришћења апликације јер знају да могу да пониште акцију уколико негде погреше. Ово, даље, подстиче корисника да испробава до тад њему непознате акције и тако открива нове функционалности апликације [6].

Држати корисника у контроли подразумева да је неопходно остварити код корисника осећај да они управљају апликацијом и да интерфејс реагује на њихове акције. Потреба за ових осећајем је нарочито изражена код искуснијих корисника који не воле да буду изненађени неочекиваном реакцијом програма на неку њима већ познату акцију или скуп акција [6].

Када се каже „смањити преоптерећеност меморије“ мисли са људску меморију, односно памћење. Људи, у просеку, нису у стању да запамте велику количину информација у кратком временском року и интерфејс не треба да захтева од корисника да памти више информација него што је потребно. На пример, не треба кориснику приказивати инструкције за обављање неког задатка на једној страници, а онда тај задатак приказати на другој, већ би све то требало да буде на истој страници [6].

3. ЕВАЛУАЦИЈА АМАЗОН АПЛИКАЦИЈЕ

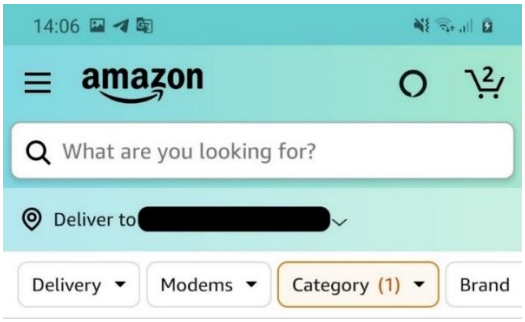
Амазон је највећа светска интернет продавница која пружа услуге пласирања и испоруке робе другим компанијама путем Амазон сајта. Поред тога, Амазон има и своје произвођаче чије производе такође продаје кроз свој сајт. На енглеском се овакав тип интернет продавница зове *online marketplace*.

3.1. Испитивање конзистентности

Апликација користи четири доминантне боје: белу за позадине, наранџасту за примарну дугмад (додавање производа у корпу, потврђивање информација, итд.), сиву за секундарну дугмад и падајуће меније и тиркизну као боју заглавља апликације.

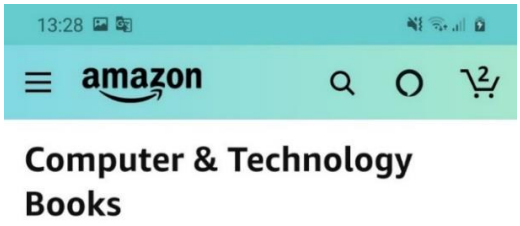
Установљено је да је, што се тиче боја, величина дугмади и слова и коришћених фонтова, апликација скоро у потпуности конзистентна са малим незнатним замеркама.

Међутим, установљено је да апликација има велику неконзистентност код приказивања поткатегорија и филтера кроз целу апликацију. Поткатегорије су понегде приказане помоћу падајућег менија (Слика 1), док су на другим местима приказане помоћу хајперлинкова (Слика 2), а негде нису уопште приказане (Слика 3). Не постоји никакво правило када се негде шта приказује. На неким *PLP (Product List Page)* активностима се поткатегорије налазе и на средини активности, међутим ни ту нису увек све излистане, већ само одабране. Једино место где се поткатегорије конзистентно налазе јесте дно саме активности што није препоручљиво место за њих јер се корисник форсира да прелиста целу *PLP* активност да би до њих дошао.



Слика 1. *Поткатегорије и филтери путем падајућих менија*

Слика 2. *Поткатегорије приказане путем хајперлинкова*



Слика 3. *Поткатегорије нису приказане*

3.2. Испитивање осталих хеуристика

Што се тиче осталих хеуристика, установљено је да су оне углавном задовољене, али се може оставити замерка да апликацији недостаје ноћни режим рада и неки начин на који би се повећала величина слова у апликацији јер су она доста мала и могу бити премала за одређене кориснике.

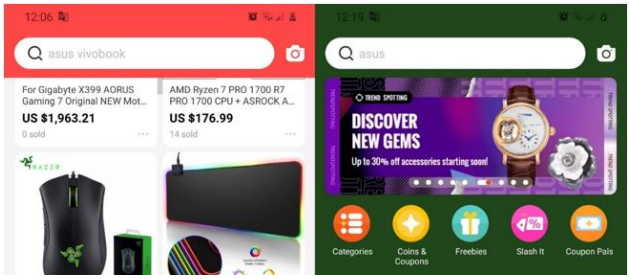
4. ЕВАЛУАЦИЈА АЛИЕКСПРЕС АПЛИКАЦИЈЕ

АлиЕкспрес је икомерц сервис са центром у Кини и у власништву кинеске компаније Алибаба. Сачињен је од малих предузећа претежно из Кине, али и из других земаља (нпр. из Сингапура) који пружају услугу куповине производа интернационалним интернет купцима [7].

Продавци на АлиЕкспресу могу бити компаније, али и појединци. АлиЕкспрес се разликује од Амазона по томе што се понаша искључиво као икомерц платформа и не продаје производе директно потрошачима, већ директно повезује компаније са купцима [8].

4.1. Испитивање конзистентности

За разлику од Амазон апликације за коју је установљено да има великих неконзистентности код навигације кроз категорије, АлиЕкспрес апликација нуди веома добро осмишљено решење тог проблема, а и има конзистентно приказиване филтере производа. Међутим, АлиЕкспрес апликација има великих проблема са конзистентношћу боја и дизајна кроз различите функционалности апликације, а понекад и код исте функционалности или активности (Слика 4).



Слика 4. *Неконзистентност главне активности*

АлиЕкспрес апликација користи белу боју као позадинску, црну за текст и црвену као водећу боју апликације. Апликација користи и друге боје код навигације одређених менија и истицања одређених аспеката корисничког интерфејса (плаву, зелену, тиркизну, наранџасту, итд.), међутим, не постоји јасно правило кад и где се која од тих боја користи.

4.2. Испитивање дијалога наглашене затворености

Процес наручивања производа код АлиЕкспрес апликације се врши из једног корака, што се не виђа често код икомерц апликација. Стандард је да се процес издели на неколико акција, тако да свака акција има своју засебну активност.

Код АлиЕкспрес апликације, међутим, све је смештено на једну активност. Не може се рећи да то њу чини претрпаном, нити да је кориснику нејасно шта треба да ради, али одузима се одређена доза задовољства коју корисник осећа док пролази кроз кораке наручивања код других апликација. Такође, овакав начин процеса наручивања може створити одређену дозу бриге и нервозе код корисника да случајно није унео нешто погрешно од својих података или да случајно не кликне дугме за наручивање без обзира на то што је апликација можда веома робусно направљена.

4.3. Испитивање осталих хеуристика

Што се тиче осталих хеуристика, установљено је да су оне углавном задовољене, али се може оставити замерка да апликацији недостаје ноћни режим рада и неки начин на који би се повећала величина слова у апликацији јер су она доста мала и могу бити премала за одређене кориснике.

5. ЕВАЛУАИЈА БАНГГУД АПЛИКАЦИЈЕ

Банггуд је компанија из Хонг Конга основана 2006. године као компанија која се специјализује за развој софтвера, међутим компанија је променила свој фокус на међународни икомерц и од тада почела да продаје велик број типова производа и доставља их широм света [9]. Компанија се бави искључиво *B2C* (Business to Consumer) продајом.

Банггуд покушава да нуди својим купцима најнижу могућу цену за одређени производ, а да тај производ буде и што је могуће квалитетнији. Ово остварују тако што робу набављају директно од произвођача и снижавају цену купцима уколико нађу негде повољнију цену за исти производ [10].

5.1. Испитивање конзистентности

Банггуд апликација користи белу боју као позадинску, црну за текст и наранџасту као водећу боју апликације. Апликација веома наличи АлиЕкспрес апликацији што се тиче графичког интефејса, али, за разлику од АлиЕкспрес апликације, нема великих проблема са неконзистентношћу у било ком аспекту свог дизајна. Постоји једна функционалност апликације где се одједном промени доминантна боја, али та промена није драстична и ова неконзистентност није велика.

5.2. Испитивање дијалога наглашене затворености

Као и код АлиЕкспрес апликације, и код Банггуд апликације се процес наручивања врши у једном кораку што представља исти проблем као и код АлиЕкспрес апликације. Мала предност коју има Банггуд у односу на АлиЕкспрес јесте што се метода плаћања бира на истој активност, док код АлиЕкспрес апликације бирање методе плаћања води до друге активности, па се са те враћа поново на активност за наручивање. Ово Банггуд апликацију чини нешто прегледнијом.

5.3. Испитивање осталих хеуристика

Што се тиче осталих хеуристика, установљено је да су оне углавном задовољене, али се може оставити замерка да апликацији недостаје ноћни режим рада и неки начин на који би се повећала величина слова у апликацији јер су она доста мала и могу бити премала за одређене кориснике.

6. ЗАКЉУЧАК

Код све три апликације које су разматране нађени су аспекти дизајна које не задовољавају неке од хеуристика. Већина хеуристика су задовољене код све три апликације, међутим одступања од хеуристика присутна у њима су довољно велика да би одређени корисници одлучили да не користе више дате апликације и траже алтернативе. Највећи проблем Амазон апликације јесте неконзистентност код приказивања филтера и поткатегорија, док је највећи проблем код АлиЕкспрес и Банггуд апликација смештање целокупног процеса наручивања на једну активност и неконзистентност у коришћењу боја.

Иако су ови недостаци значајни, они би се могли релативно лако уклонити. Код Амазон апликације неопходно је да се категорије и поткатегорије приказују на једном месту на активности, а не више и да оне буду или увек видљиве на екрану или увек доступне да буду видљиве. Што се тиче филтера, они морају бити на исти начин видљиви на свакој *PLP* страници.

Код АлиЕкспрес и Банггуд апликација потребно је да се процес наручивања подели у кораке као код Амазон апликације и као што и прописују икомерц стандарди. Даље је потребно одредити сет боја које ће се конзистентно користити кроз целу апликацију. Сет боја би требао бити што мањи, али уколико се жели користити већи сет, требало би користити монохроматски сет боја.

7. ЛИТЕРАТУРА

[1] Википедија, „Smartphone - Wikipedia“, en.wikipedia.org/wiki/Smartphone, Страница о паметним телефонима на Википедији

[2] Ryan Block, „Live from Apple's iPhone SDK press conference | Engadget“, www.engadget.com/2008-03-06-live-from-apples-iPhone-press-conference.html, Страница о првом представљању App Store - а

[3] Steven Hoober, „How The Users Really Hold Mobile Devices? :: UXmatters“, www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php, Страница о првом представљају App Store - а

[4] A. Sears, J. Jacko, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Application – 2nd Edition*, New Jersey: Lawrence Erlbaum Associates, 2007, p. 5

[5] И. Видојевић, *Речник социјалног рада*, Београд: Tiro-erc, 2006

[6] Бен Шнајдерман, “Ben Schneiderman”, www.cs.umd.edu/users/ben/goldenrules.html, Кратак опис осам златних правила

[7] АлиЕкспрес, „AliExpress – Wikipedia“, en.wikipedia.org/wiki/AliExpress, АлиЕкспрес страница на Википедији

[8] АлиЕкспрес, „AliExpress – Wikipedia“, en.wikipedia.org/wiki/AliExpress, АлиЕкспрес страница на Википедији

[9] Banggood, „About Banggood“, www.banggood.com/About-Banggood_hl14, Кратко објашњење шта је Банггуд

[10] Banggood, „Why are Banggood's products so cheap?“, www.banggood.com/About-Banggood_hl71_at257, Објашњење зашто су производи јефтини

Кратка биографија:

Марко Зорић рођен је у Зрењанину 1995. год. Мастер рад на факултету техничких наука из области Електротехнике и рачунарства – Рачунарство и аутоматика одбранио је 2020. год.
контакт: m.s.zoric5@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE40Sabados

AUTOMATSKO GENERISANJE SKUPA PODATAKA ZA TRENIRANJE MODELA ZA AUTOMATSKO PREPOZNAVANJE OSOBE NA SLICI

AUTOMATED DATASET GENERATION FOR PERSON-IDENTIFICATION MODEL TRAINING

Noemi Sabadoš, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Kvalitetni skupovi podataka su retko dostupni, što otežava treniranje kvalitetnih modela mašinskog učenja. Cilj ovog rada jeste generisanje skupa podataka koji bi mogao biti korišćen za obuku modela mašinskog učenja čiji bi cilj bio da uporedi slike ljudi sa slikama poznatih ličnosti i prepozna sličnosti u izgledu. Generisanje skupa podataka se vrši automatski, na osnovu unapred zadatih kriterijuma. Svrha kriterijuma je da učine skup podataka raznovrsnim, sa ciljem poboljšanja generalizacije nad njime obučenih modela. Podržani kriterijumi uključivanja slika u skup podataka su sledeći: (1) da li se osoba smeši ili ne, (2) specifikacija smera pogleda, (3) činjenica da li osoba ima: bradu, šiške, kosu, (4) da li osoba nosi naočare ili kapu i (5) da li osoba rukom zaklanja lice. Prikupljeni skup podataka sadrži slike o 100 glumca, koje su skinute sa interneta prema zadatim kriterijumima.*

Ključne reči: *skup podataka, generisanje skupa podataka, skup podataka o licu, skup podataka o slavnim ličnostima, detekcija atributa lica*

Abstract – *The basis for successful training of quality machine learning models are the datasets, but quality ones are hard to find in most cases. This paper aims to generate a dataset that could be used in machine learning as a training set for comparing celebrities with ordinary people. The generation process is automatic and based on predetermined criteria, aimed to make the dataset diverse. Dataset diversity hould enhance the generalization of the trained models. Among the supported criteria are: (1) whether the person is smiling, (2) the direction of the person's gaze, (3) whether the person has a beard, bang, or hair, (4) whether the person wears glasses or a hat, or (5) is the persons' hand in front of the face. The generated dataset contains publicly available images of 100 actors.*

Keywords: *dataset, generating dataset, face dataset, celebrity dataset, face attribute detection*

1. UVOD

U oblasti informacionih tehnologija, mašinsko učenje je oblast koji se najbrže razvija. Korektan rad sistema zasnovanih na mašinskom učenju u velikoj meri zavisi od skupa podataka korišćenog za treniranje modela. Loš kvalitet skupova podatka uzrokuje loš rad sistema.

Postoje javno dostupni skupovi podataka za prepoznavanje ličnosti na slici. Međutim, ni jedan se ne fokusira na anotaciju smera pogleda osobe na slici. Pri tome, ovakve anotacije bi bile od velike pomoći za aplikacije čiji je cilj određivanje sličnosti lica prikazanog na dve različite slike. Cilj ovog rada je napraviti skup podataka koji će ublažiti ovaj i slične nedostatke. Pored smera pogleda, razmatrane su još neke osobine lica koje mogu isto da doprinesu detekciji sličnosti:

- prepoznavanje osmeha
- prepoznavanje brade
- prepoznavanje šiški
- prepoznavanje ćelavosti
- prepoznavanje naočara
- prepoznavanje kape i/ili kačketa
- prepoznavanje ruku koje zaklanjaju lice.

U drugom poglavlju će biti predstavljeni radovi koje su imali uticaj na rešenje predstavljeno u ovom radu. U trećem poglavlju će biti opisane faze generisanja skupa podataka. O rezultatima će biti reč u četvrtom poglavlju. U petom poglavlju prikazan je rezime rada i pravci budućeg razvoja.

2. POSTOJEĆA REŠENJA

Postoji mnogo javno dostupnih skupova podataka sa slikama javnih ličnosti, ali je fokus ovog rada na skupovima podataka koji sadrži slične karakteristike razmatrane u ovom rešenju. Odnosno, skup podataka bi trebao da sadrži podatke o smeru glave, da osobe na slikama budu poznate ličnosti i da, osim toga, sadrže više osobina koje mogu pomoći u definisanju sličnosti lica na dve različite slike. Po znanju autora, trenutno ne postoji ni jedan skup podataka koji zadovoljava sve ove kriterijume.

Jedan od najpoznatijih javno dostupnih skupova podataka je *Labeled Faces in the Wild* (LFW) [1]. Skup podataka je napravljen za prepoznavanje lica. Verifikacija lica ili drugačije *pair matching*, je metoda u kome upoređujemo dve slike i određujemo da li je ista osoba prisutna na obe slike. Skup sadrži 13.233 slika o 5.749 osoba. Za većinu osoba prisutna je samo jedna slika, ali je za 1.680 imena prisutno 2 ili više slika. Dodatna važna osobina jeste da su slike snimane u prirodnoj okolini, dakle, ne u veštački pripremljenim uslovima. Cilj da su prikupljene slike iz svakodnevnice, bez uticaja okoline ili poboljšanja.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Jelena Slivka, vanr. prof.

Rad „*Names and faces in the news*“ [2] je dodao slike iz *online* novina. Tako slike iz LFW sadrže slike o osobama, koje su se pojavile u ovim člancima, ali uglavnom se radi o slikama poznatih ljudi koje ne sadrže nikakve dodatne informacije poput smera pogleda, osmeha, itd.

Skup podataka *CelebFaces Attributes Dataset* (CelebA) [3] sadrži slike poznatih ličnosti na kojima je označeno mnogo različitih osobina. Ovaj skup podataka je najsličniji onom predstavljenom u ovom radu, sa izuzetkom smera pogleda, koji nije uključen u [3]. Skup podataka sadrži 10.177 osoba na više od 200 hiljada slika. Pored toga, na slike su dodate anotacije sa 5 lokacijskih obeležja i 40 binarnih atributa.

Među anotiranim osobinama nalazi se postojanje šiški, osmeha, naočara, i sl. U [3] kreiran je skup podataka na osnovu slika iz *CelebFaces+* [4], pri čemu su kod svake manuelno dodate anotacije za 40 atributa. Nažalost, smer glave nije bio između ovih osobina. Cilj rada [5] jeste da se kreira skup podataka koji će služiti za treniranje softvera za prepoznavanje lica, procenu smera pogleda ili lokalizaciju tačaka lica. Konačan skup sadrži 367.888 slika od 8.277 osoba. Slike nisu iz kontrolisane okoline, već iz svakodnevnice. Skup podataka sadrži i anotaciju graničnog okvira lica, koja je verifikovana manuelno. Za svaku sliku su dodate i lokacije ključnih tačaka, pozicija (rotacije oko z, y i x ose) i informacije o polu.

Ovaj skup već sadrži smer pogleda, međutim, u drugoj formi od očekivane, to jest, nisu zabeleženi uglovi rotacija i definisani smerovi. Ne sadrži nikakve dodatne informacije koje bi mogli biti iskorišćene u definiciji sličnosti dve osobe na slici.

3. GENERISANJE SKUPA PODATAKA

Prvi korak za generisanje skupa podataka prikazanog u ovom radu je kolekcija liste poznatih osoba. Za ovo je korišćena lista[1] od 1000 poznatijih glumaca. Za prvih 100 glumaca sačuvano je oko 200 slika sa *Bing Image Downloader*[2]. Sledeći korak je bio filtriranje slika (više detalji u poglavlju 3.1). Nakon kolekcije slika, slede faze generisanje skupa podataka. Na kraju su manuelno uklonjene pogrešne anotacije.

Koraci generisanja skupa podataka se mogu videti na slici 1. Prvo je detektovano lice na slici pomoću kombinacije *Histogram of Oriented Gradients* (HOG) obeležja i SVM (*Support Vector Machine*) modela [6]. Nakon toga su filtrirane slike gde oči nisu otvorene. Nakon toga sledi detekcija: osmeha, smera glave, šiški, brade, ćelavosti, naočara, kape i ruke ispred lica. Za svaku od navedenih osobina su detektovane slike iz filtriranog skupa. Nakon što su za datu osobu pronađene 3 slike po gore navedenim kriterijumima, potraga za slikama ove osobe se završava.

Slika 1. Koraci kreiranja skupa podataka

[1] www.imdb.com/list/ls058011111/

[2] github.com/gurugaurav/bing_image_downloader

3.1. Filtriranje slika

Zbog velike količine prikupljenih podataka, filtriranje je trebalo biti automatsko. Ovaj korak je neophodan jer su prikupljene slike pomoću *Bing* pretraživača koji ne garantuje da je tražena osoba na slici.

Prvi korak za filtriranje je detekcija lica. Ako broj pronađenih lica nije jedan, slika je uklonjena. Sledeći korak je upoređivanje lica na slici sa referentnom slikom (za koju znamo da sadrži traženu osobu). Cilj upoređenja lica je da se filtriraju slike na kojima je lice osobe jako različito, odnosno, da se uklone direktorijumi koji verovatno sadrže slike različitih osoba. Za referentnu sliku uzeta je profilna sliku od *TMDb* [3] stranice. Upoređivanje slika sa referentnom je vršeno pomoću *Openface*-a [7], pri čemu su filtrirane slike koje su bile jako različite. Poslednji korak je bio brisanje duplikata. Ovo je izvršeno *scalable template matching* metodom.

3.2 Prepoznavanje zatvorenih očiju

Inspiraciju za prepoznavanje zatvorenih očiju je dao rad [8] u kome je opisana metoda za prepoznavanje treptanja na video snimcima. Otvorenost očiju se vrši merenjem *EAR* (*eye aspect ratio*). U ovom cilju izvršena je klasifikacija sa sledećim ulaznim podacima: 3 normalizovane udaljenosti tačaka očiju, EAR, pol i rasa. Najbolji rezultat je postugnut primenom *Gradient Boosting* modela.

3.3 Prepoznavanje osmeha

Konvolucione neuronske mreže (*Convolutional Neural Networks,* CNN) su najefektivnije za klasifikaciju slike i stoga su često korišćene za prepoznavanje osmeha [9, 10]. Ovo je bio razlog za odabir CNN modela za prepoznavanje osmeha, na osnovu *Keras mnist_cnn.py*[4] primera. Ova arhitektura trenirana je na slikama iz *SMILEsmileD*[5] skupa, i korišćena je za prepoznavanje osmeha.

3.4 Prepoznavanje smera glave

Za prepoznavanje smera glave isprobano je više načina, ali je na kraju kreirana posebna neuronska mreža. Mreža ima 15 ulaznih neurona, koji kao ulaz primaju udaljenosti ključnih tačaka lica. Mreža ima jedan skriveni sloj sa 60 neurona. Aktivaciona funkcija je *softmax*. Za treniranje je korišćen *Head Pose Image Database* [11] skup podataka. Skup sadrži 2 790 slika o 15 osoba. Smer pogleda je određen sa 2 ugla. Vertikalni ugao označava levi i desni smer, a horizontalni gore i dole. Mnogo slika ovog skupa podataka je moralo biti odbačeno, jer ovde predstavljena aplikacija nije detektovala lica na slikama uslikanim iz profila. Na kraju je za treniranje korišćeno 792 slika.

Prilikom testiranja detekcije, nije primećena nijedna slika gde je osoba gledala na dole, ali je bilo mnogo slučajeva da je za lice sa centralim smerom pogleda prediktovano da osoba gleda dole. Zbog ovoga, svaki dole prediktovan smer je nadalje računat kao centralni. Smer pogleda na gore je takođe redak, i preciznost detekcije je svega 0.49,

[3] www.themoviedb.org

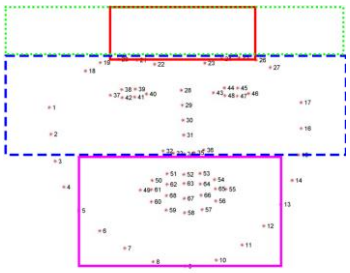
[4] github.com/keras-team/keras/blob/

[5] github.com/hromi/SMILEsmileD

tako da i on odbačen. Na kraju, aplikacija traži smerove levo, desno i centriran.

3.5 Prepoznavanje brade

Za prepoznavanje korišćena je ista arhitekturu kao kod prepoznavanje osmeha (poglavlja 3.2). Za treniranje su korišćeni isečci slika iz *CelebA* [3]. Isečci su predstavljali područja usana (slika 2), i model je treniran na ovim slikama. Detekcia brade je izvršena samo u slučaju da je osoba na slici muškarac.



Slika 2. Isecanje za različite detekcije – ćelavosti: crvena puna linija, naočare: plava isprekidana linija, kapa: zelena tačkasta linija, brada: ljubičasta puna linija

3.6 Prepoznavanje šiški

Prepoznavanje šiški je izvršeno na sličan način kao prepoznavanje brade (poglavlje 3.5). Jedina razlika je da se ovde za treniranje koriste slike sa tamnom kosom, jer su svetle kose zbunjivale detekciju. Najvažniji deo je bio izbeći *false positive* (FP) u trening skupu, tako da su slike koje su bile rizičnije izbačene iz trening skupa. Rizičnije slike su one na kojima aplikacija prepoznaje i ćelavost (poglavlje 3.7) i šiške na istoj slici. Takođe su odbačene slike gde je prepoznata kapa (poglavlje 3.9).

3.7 Prepoznavanje ćelavosti

Problem prepoznavanja ćelavosti je sličan problem kao detekcija brade ili šiški, tako da je opet korišćena ista CNN arhitektura. I u ovom slučaju su korišćene slike i labele iz *CelebA* [3] skupa. Međutim, sada je umesto da se koristi celo lice za treniranje korišćen samo deo lica između spoljnih uglova oka i od obrva do vrha okvirne glave (slika 2). Slično kao i kod prepoznavanja šiški, i ovde su izbačene slike na kojima su detektovane šiške ili kape. Aplikacija je detektovala ćelavost samo ako je osoba na slici bio muškarac.

3.8 Prepoznavanje naočara

Za prepoznavanje naočara isprobane su dve metode. Obe metode treba da prepoznaju naočare u ciju dodavanja takvih slika u skup podataka.

Prvi pristup je koristio istu CNN arhitekturu za prepoznavanje drugih aspekata lica (poglavlje 3.2, 3.5-3.7). Takođe je korišćen *CelebA* [3] skup podataka, gde je isečeno područje koja počinje od nosa i ide do najviše tačke obrva (slika 2).

Drugi pristup je detekcija objekta pomoću *Mask R-CNN* [12]. Za ovaj pristup sačuvane su slike za treniranje iz *Open Images Dataset* [13] sa naočarima za sunce, gde je dodat granični okvir za svake naočare. Nakon toga je istreniran *Mask R-CNN* model. Dodat je kriterijum da naočare trebaju biti prepoznate sa minimalnom verovatnoćom 0,9 u da se pronađeno područje nalazi u području lica.

3.9 Prepoznavanje kape i/ili kačketa

Mask R-CNN [12] pristup je isproban i na problemu prepoznavanja kape, ali nije bio uspešan. Ovde je zato ipak korišćena jednostavna CNN arhitektura (poglavlje 3.2, 3.5-3.8). Prvi korak je bila anotacija slika iz *Open Images Dataset* [13], gde je isečena regija od obrva do vrha lica (slika 2). Nad takvim slikama je trenirana CNN.

3.10 Prepoznavanje ruku ispred lica

Prvi korak za detekciju ruku koje zaklanjaju lice je bio detekcija ruke. Za procenu tačke ruke je korišćen postojeći model *OpenPose* [14]. Model procenjuje 21 tačku za ruku. Aplikacija je trebala samo da odluči da li je ta ruka ispred lica ili ne. Prvo je isečeno lice iz slike, nakon čega se vrši predikcija tačaka ruke. Ako je aplikacija na isečenoj slici lica našla bar 5 tačaka, slika je anotirana da sadrži ruku ispred lica.

4. REZULTATI I SKUP PODATAKA

U ovom poglavlju prvo će biti predstavljeni rezultati za svako prepoznavanje na jednom velikom skupu. Nakon toga će biti predstavljen izgenerisani skupu podataka.

4.1. Rezultati prepoznavanja

Da testiram svoje detekcije, želela sam da kreiram jedan test skup podataka koji će biti sličan onome koji je cilj bio generisati. Za filtriranje su korišćene profilne slike iz *TMDb* stranice. S obzirom na to da se tamo se može naći više od jedne slike, sve su sačuvane. Ukupno sam uspela da sakupim 1 921 sliku, od čega su 1 313 slika žena, a 608 slike muškaraca. Za pojedinačne slike sistem nije mogao da pronađe lice, neke tačke na licu i slično. Te slike su preskočene za određenu detekciju. Anotacije skupa podataka sam uradila manualno.

U tabeli 1 mogu se videti rezultati za evaluaciju. Budući da kreiranji test skup sadrži jako mnogo negativnih primera, mikro f1 mera nije toliko značajna. Umesto toga, cilj je maksimizovati preciznost. Prikazani rezultati ne sadrže prepoznavanje zatvorenih očiju jer u ovom skupu nije postojao ni jedna slika gde bi oči bile zatvorene. Od 1 778 slika, samo sam 5 puta dobila rezultate da oči nisu otvorene.

U rezultatima prikazanim u tabeli 1 može se videti da prepoznavanje osmeha, smera glave i brade nema gotovo nijedan FP rezultat. Prepoznavanje šiški je nešto lošije, ali broj FP nije značajan. Najveći problem sa ovom detekcijom jeste da sistem nije prepoznao mnogo slika sa plavim šiškama.

Za prepoznavanje ćelavosti i naočara gotovo za svaku treću sliku nije bio dobar rezultat. Detekcija ćelavosti ima problem sa sedom kosom i visokim čelom. Rezultati za prepoznavanje kape i ruke ispred lica su takođe nezadovoljavajući zbog velikog broja FP.

Tabela 1: Rezultati prepoznavanja različitih osobina slike

Prepoznavanje	Mikro f1 mera	Preciznost
prepoznavanje osmeha	0.77	0.97
prepoznavanje smera glave	0.95	0.95
prepoznavanje brade	0.83	0.98
prepoznavanje šiški	0.84	0.83
prepoznavanje ćelavosti	0.92	0.59
prepoznavanje naočara	0.98	0.62
prepoznavanje kape i/ili kačketa	0.98	0.35
prepoznavanje ruku ispred lica	0.95	0.29

4.2. Izgenerisan skup podataka

Ceo proces generisanje skupa podataka je automatski, pomoću implementirane aplikacije. Kao izvor slika, korišćen je *Bing*, što nije bila najbolja opcija. Od 200 sačuvanih slika, nakon filtriranja je preostalo svega oko 20-30 slika po osobi. Dobavljene slike sadržale su mnogo duplikata, slika više ljudi ili slike bez ljudi, slike slabog kvaliteta, itd. Nakon generisanja skupa, manuelno su eliminisane pogrešno prediktovane slike. Zbog lošeg izvora, konačan skup je sadržao samo 716 sliku. Za svaku osobu, za svaku detekciju možemo da imamo maksimum tri slike. U tabeli 2. mogu se videti statistički podaci o generisanom skupu podataka.

Tabela 2: Statistički podaci i broj slike za različite kriterijume

	Žena	Muškarac	Ukupno
Broj osoba	60	40	100
Broj slika	450	266	716
Osmeh	179	113	292
Smer centar	177	118	295
Smer levo	47	21	68
Smer desno	45	13	58
Šiške	150	28	178
Brada	-	80	80
Ćelavost	-	45	45
Naočare	5	19	24
Kapa i/ili kačket	8	3	11
Ruka ispred lica	30	6	36

5. ZAKLJUČAK

U ovom radu predstavljena je aplikacija za generisanje skupa podataka na osnovu više kriterijuma kao što su smer glave, postojanje osmeha, brade, šiški, ćelavosti, naočara, kapa i ruku ispred lica. Prepoznavanje ovih kriterijuma vršeno je automatski. Najbolji rezultati su dobijeni na prepoznavanju osmeha, smera glave i brade (preciznost preko 0.95). Najgore rezultate imaju prepoznavanje kape i ruku ispred lica (preciznost oko 0.3).

Pomoću implementirane aplikacije generisan je novi skup podataka. Izvor slika je bio *Bing*, koji nije predstavljao najbolju opciju. Konačan skup ima 716 slika od 100 glumaca.

Budući plan je ispraviti prepoznavanja sa lošijim rezultatima, i generisati skup podataka korišćenjem drugog izvora.

6. LITERATURA

[1] Huang, Gary B., et al. "Labeled faces in the wild: A database for studying face recognition in unconstrained environments." 2008.

[2] Berg, Tamara L., et al. "Names and faces in the news." Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.. Vol. 2. IEEE, 2004.

[3] Liu, Ziwei, et al. "Deep learning face attributes in the wild." Proceedings of the IEEE international conference on computer vision. 2015.

[4] Sun, Yi, Xiaogang Wang, and Xiaoou Tang. "Deep learning face representation from predicting 10,000 classes." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[5] Bansal, Ankan, et al. "Umdfaces: An annotated face dataset for training deep networks." 2017 IEEE International Joint Conference on Biometrics (IJCB). IEEE, 2017.

[6] Déniz, Oscar, et al. "Face recognition using histograms of oriented gradients." Pattern recognition letters 32.12 (2011): 1598-1603.

[7] Amos, Brandon, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. "Openface: A general-purpose face recognition library with mobile applications." CMU School of Computer Science 6.2 (2016).

[8] Soukupová, Tereza, and Jan Cech. "Real-time eye blink detection using facial landmarks." 21st Computer Vision Winter Workshop. 2016.

[9] Chen, Junkai, et al. "Smile detection in the wild with deep convolutional neural networks." Machine vision and applications 28.1-2 (2017): 173-183.

[10] Zhang, Kaihao, et al. "Facial smile detection based on deep learning features." 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR). IEEE, 2015.

[11] Gourier, Nicolas, Daniela Hall, and James L. Crowley. "Estimating face orientation from robust detection of salient facial features." ICPR International Workshop on Visual Observation of Deictic Gestures. 2004.

[12] He, Kaiming, et al. "Mask r-cnn." Proc. of the IEEE international conference on computer vision. 2017.

[13] Kuznetsova, Alina, et al. "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale." arXiv preprint arXiv:1811.00982 (2018).

[14] Simon, Tomas, et al. "Hand keypoint detection in single images using multiview bootstrapping." Proc. of the IEEE conference on Computer Vision and Pattern Recognition. 2017.

Kratka biografija:

Noemi Sabadoš rođena je 1995. godine u Vrbasu. Osnovne akademske studije završila 2018. godine na Fakultetu Tehničkih Nauka Novi Sad, na kom brani i master rad 2020. godine iz oblasti Elektrotehnike i računarstva – Softversko inženjerstvo i informacione tehnologije.
kontakt: nomisabi@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE41Cukovic

ЈЕДАН ПРИМЈЕР АНАЛИЗЕ СИСТЕМА ТОЛЕРАНТНОГ НА ОТКАЗЕ

AN EXAMPLE OF FAULT TOLERANT SYSTEM ANALYSIS

Ивана Ћуковић, *Факултет техничких наука, Нови Сад*

Oblast – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Kratak sadržaj – *Овај рад се бави истраживањем поузданости сервера задуженог за прикупљање и пренос података приликом примјене вруће резерве.*.

Ključne reči: *Поузданост сервера, Редуданса, Врућа резерва*

Abstract – *This paper examines reliability of server which is used for collecting and transmission of data on which is applied hot-standby redundancy.*

Keywords: *Reliability of server, redundancy, hot-standby redundancy*

1. УВОД

Сваки софтвер је подложан многим грешкама које могу бити предвиђене или непредвиђене. Свака грешка на софтверу са собом носи одређене посљедице.

Циљ сваке компаније је да произведе рјешење које је толерантно на отказе. С обзиром на то да постоји велики број типова отказа, то није тако једноставно. Испитивањем типова отказа, дошло се до закључка које особине сваки софтвер треба да има како би био толерантан на отказе. Једна од тих особина на коју ће се посебно обратити пажња је поузданост. Поузданост се карактерише као способност континуалног извршавања без прекида. До постизања ове особине се може доћи на различите начине, а један од тих начина је примјена технике редундансе

Редуданса се огледа у дуплирању неког сервера, како би у случају његовог пада неки од тих дупликата могао да преузме његову улогу и да настави са радом. Број сервера који ће се налазити у позадини, као и начин њиховог рада даље зависе од типа редундансе која је примјењује. Као предмет истраживања овог рада посматраће се врућа резерва, као и њене предности и недостаци.

1.1. Грешка и узрок грешке

Веома често се грешка (*енгл. error*) и узрок грешке, односно отказ, (*енгл. fault*) користе у истом контексту, иако у њиховом значењу постоје знатне разлике.

Узрок грешке се односи на дефекте на хардверу или на неку софтверску грешку (*енгл. bug*), док грешка представља манифестацију тог узрока или *bug*-а [1].

Хардверски узроци грешака могу бити класификовани као пролазни (*енгл. transient*), испрекидани (*енгл. intermittent*) и стални (*енгл. permanent*). Поред ове подјеле, хардверски узроци грешака се још могу подијелити на добронађудне (*енгл. benign*) и злонамјерне (*енгл. malicious*) [1].

1.2. Типови отказа и толерантност на отказе

Постоје различити типови отказа [2]:

- Отказ у сличају пада (сервер се зауставља, али је до тада радио коректно)
- Отказ у случају пропуста (сервер отказује код одговора на захтјеве клијената)
  - Пропуст при пријему (сервер отказује код пријема порука)
  - Пропуст при слању (сервер отказује код слања порука)
- Отказ због временског ограничења (одговор од сервера касни дуже од специфичног периода)
- Отказ при одговору (одговор сервера је некоректан)
  - Отказ због вриједности (вриједност у одговору сервера је лоша)
  - Отказ због промјене стања (сервер одступа од стандарних активности)
- Произвољан или византијски отказ (сервер може дати произвољан одговор у произвољно вријеме).

Толерантност на отказе представља способност система да настави извршавање без прекида у случају да једна или више његових компоненти откаже [1]. Системи који су толерантни на отказе морају да испуњавају сљедеће:

- доступност,
- поузданост,
- сигурност и
- одрживост.

1.3. Редуданса

Једна од основних особина које купац захтјева за свој поризвод од произвођача је поузданост. Поузданост се може постићи на разне начине, као што су побољшање поузданости компоненти или примјена техника редундансе.

Редуданса представња главни метод за постизање поузданости и сигурности система [3]. Додавањем редундансе се повећава цијена и комплексност система. Опште прихваћена чињеница је да се не може постићи поузданст система отпорног на грешке

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Дарко Чапко, ванр. проф.

без употребе редудансе. Постоје два основна типа редудансе, активна редуданса и резерва (*енгл. standby redundancy)* [4]. На слици 1 можемо видјети графички приказ подјеле редудансе

Слика 1. *Графички приказ подјеле редудансе*

Активну редудансу карактерише то што су све исправне редундантне компоненте у оперативном стању све вријеме, чак иако је у том моменту за нормално функционисање система потребно само њих неколико.
У овом случају нема промјене на исправну комроненту или покретања нове исправне компоненте након пада друге [5].

За разлику од активне редудансе, резерву карактерише оперативно стање компоненти само у случају када постоји недостатак компоненти које су у том стању, а неопходне су за успјешно функционисање система. Резерва се даље може класификовати као хладна (*енгл. cold*), топла (*енгл. warm)* и врућа (*енгл. hot*) [4].

Хладна резерва подразумијева коришћење редундантне компоненте на начин да она буде заштићена од стреса који носи оперативно стање. Без изложеност овом стресу, вјероватноћа испада те компоненте је веома ниска, претпоставља се да је приближна нули, све до момента када мора да постане оперативна и замијени компоненту на којој се десио испад [6].

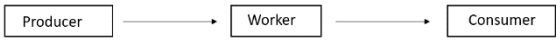
Топлу резерву карактерише степен вјероватноће испада редундантне компоненте који је између степена вјероватноће хладне и степена вјероватноће вруће редудансе приправности, која ће бити детаљно објашњена у наредном поглављу. Хладна, као и врућа резерва, представљају један од облика топле резерве [7].

Када редундантна компонента ради у паралели са примарном компонентом, обављајући идентичан посао и посједовајући исте податке, спремна да у сваком моменту преузме примарну улогу, говоримо о врућој резерви. Овај облик редудансе обезвјеђује брз опоравак након испада примарне компоненте, али и додатне операције и трошкове будући да троши енергију и материјале.

С обзиром да се налази под потпуним стресом који носи оперативно стање, на овој компоненти може да се деси испад и прије него што она замјени примарну компоненту, односно степен вјероватноће испада редуданте компоненте је једнак степену вјероватноће испада примарне компоненте [7].

2. ФОРМУЛАЦИЈА ПРОБЛЕМА И ПРЕДЛОЖЕНО РЈЕШЕЊЕ

Сваки прекид у раду система може да изазове велике посљедице које је тешко исправити. Посматране су посљедице прекида рада система за прикупљање и прослеђивање података у електроенергетском систему.

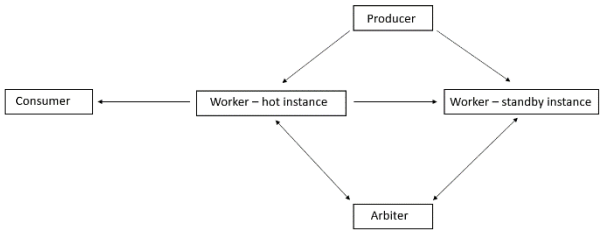


Слика 2 – *Архитектура основног система*

На слици 2 је приказана архитектура упрошћеног система за прикупљање и прослеђивање података у електроенергетском систему. Систем се састоји од три основне компоненте:

- *Producer*, компонента која шаље податке
- *Worker*, средишња компонента која прикупља и даље просљеђује податке
- *Consumer*, крајња компонента којој се испоручују прикупљени подаци на даљу обраду.

Проблем који се јавља приликом овог рјешења је у случају испада на *Worker*-у. Када компонента престане да ради, подаци са паметних мјерила не долазе до *Consumer*-а до његовог поновног подизања, а подаци који су стигли прије пада, а нису прослијеђени се губе. Количине података које се изгубе приликом оваквог испада компоненте могу да изазову велике проблеме, услијед неажурних података у електроенергетском систему.



Слика 3 – *Архитектура предложеног софтверског рјешења*

На слици 3 је приказана архитектура предложеног софтверског рјешења како би се избјегли проблеми губљења података.

Компонента *Worker* добија своју копију, односно креира се још једна њена инстанца. Двије инстанце се разликују по томе што се при покретању једна поставља за примарну (енгл. *hot instance*), а друга за редундантну инстанцу (енгл. *standby instance*).

Producer у овом рјешењу комуницира и са примарном и са редундантном инстанцом истовремено, шаљући им исте податке. Разлог је управо то што врућу резерву карактерише да обје њене инстанце имају исте податке.

Иако имају исте података, остатак посла не извршавају на исти начин. Обје инстанце прихватају исте податке, чувају их, али када дође до дијела да се ти подаци прослеђују даље, то ради само примарна компонента. Слањем података само са примарне компоненте се избјегава дуплирање података на пријемној страни. Поред тога што примарна инстанца шаље податке ка компоненти *Consumer*, она обавјештава редундантну компоненту о идентификационом броју података који је последњи прослиједила, како би

редундантна инстанца, у случају да постане примарна, знала од ког дијела треба да настави слање. Подразумјева се да сваки податак који прође кроз систем има јединствен идентификациони број.

Arbiter води рачуна која инстанца ће бити примарна,а која редундантна. *Arbiter* покреће обје инстанце компоненте *Worker*. Инстанца која се прва покрене добија примарну улогу, а друга редундантну. Након покретања примарне и редундантне инстанце, њен задатак је да на одређен временски период „прозива“ обје и да оне узвраћају на тај позив како би знала да су још увијек активне. У овом конкретном случају дефинисани период је једна секунда.

3. ТЕСТИРАЊЕ

Тестирано је предложено рјешења са 5 инстанци компоненте *Producer* које шаљу фиксан број података у једној секунди. Тестови су урађени за 10.000, 15.000 и 20.000 података који стижу са компоненте *Producer* у једној секунди, при чему се тестови извршавају док број послатих података не достигне 1.000.000. Подаци који се шаљу у секунди од стране компоненте *Producer* су груписани у пакете, не шаљу се појединачно.

Пад примарне инстанце компоненте *Worker* је симулиран гашењем те инстанце у произвољном временском тренутку. Будући да се ради о веома малим јединицама времена у којима се све извршава, немогуће је одредити тачан временски моменат у коме желимо да компонента престане са својом функционалношћу.

Тестирањем су обухваћени случајеви до којих може да се дође примјеном овог рјешења.

3.1. „Идеалан“ случај примјене предложеног рјешења

„Идеалним“ случајем можемо сматрати случај када након пада примарне инстанце, редундантна одмах преузима њену улогу и наставља са радом. Под овим се подразумијева да је примарна инстанца оборена у моменту након што је прослиједила податке примљене у претходној секунди, као и да је након слања тих података обавјестила редундантну инстанцу о идентификационом броју последњег послатог података. Након преузимања примарне улоге, инстанца зна тачно од ког података треба да настави слање.

Можемо посматрати два различита периода рада овог предложеног рјешења. Један период је прије пада примарне компоненте, а други након њеног пада и преузимања њене улоге од стране редундантне компоненте.

Први посматрани случај је слање 10.000 података са 5 инстанци. Просјечно вријеме које је потребно да подаци стигну од компоненте *Producer* до компоненте *Consumer*, у случају када је редундантна компонента ниједном није преузела улогу примарне, је 981,6 милисекунди са стандардном девијацијом од 309 милисекунди. Када посматрамо просјечно вријеме преноса података након што је редундантна компонента преузела улогу примарне, можемо закључити да нема велике разлике. Просјечно вријеме преноса у том случају је 876 милисекунди са стандардном девијацијом од 319 милисекунди. За пренос свих података, просјечно вријеме трајања преноса података износи 906,7 милисекунди са стандардном девијацијом од 319,6 милисекунди.

Како би показали да овај случај није примјењив само када је број података мањи, тестиран је случај када се са истог броја инстанци компоненте *Producer* шаље дупло већи број података. Свака инстанца у периоду од једне секунде шаље 20.000 података.

Како би показали сличност са претходним примјером, посматра се вријеме које је потребно за пренос података прије пада примарне инстанце, и након њеног пада и преузимања рада од стране редундантне инстанце. Прије пада примарне инстанце просјечно вријеме слања података је 1,99 секунди са стандардном девијацијом од 448 милисекунде, а након пада 1,49 секудни са стандардном девијацијом од 407 милисекунди. Можемо примјетити да временска разлика између првог и другог периода није значајна и да је веома слична случају када се шаље 10.000 података. Просјечно вријеме преноса података је 1,84 секунди са стандардном девијацијом од 494 милисекунде.

Када посматрамо просјечно вријеме преноса података у случају са 10.000 података и у случају са 20.000 података видимо одређену разлику. Међутим, та разлика је управо због количине података која се преноси.

3.2. Веће кашњење проузроковано нагомилавањем података

Да би боље разумјели овај случај потребно је добро познавати рад редундантне компоненте. Када се покрене инстанца компоненте *Worker* она прима податке, складишти их и одбројава временски период од једне секунде након чега се иначе врши слање прикупљених података. Међутим, редундантна инстанца компоненте након истека периода од једне секунде не ради ништа. Задржава податке које има док год јој примарна компонента не пошаље идентификациони број последњег послатог података, а затим брише све податке који су стигли прије њега, као и њега самог.

У овом конкретном случају примарна компонента је „оборена“ тако да у неком временском периоду ниједна компонента није имала улогу примарне. Управо због тога је редундантна компонента, када је постала примарна, имала већи број скупљених података које је требало прослиједити. Тај већи број података, који се разликује од просјечног броја података послатих у претходном периоду, доводи до кашњења које затим утиче на све наредне пакете података. Подаци се и даље шаљу сваке секунде, али компонента *Consumer* их не преузима тако брзо.

За разлику од случаја описаног у претходном примјеру, у овом случају ћемо видјети како број података који се шаље значајно утиче на перформансе предложеног рјешења.

Посматрањем резултата добијених тестирањем када се шаље 10.000 података закључује се да након промјене улоге примарне и редудантне компоненте, број података је већи од просјечног броја који се шаље. У даљем наставку рада се тај број пакета враћа на просјечан. То је управо посљедица што у периоду од једне секунде ниједна од ове двије инстанце компоненте *Worker* није имала улогу примарне и долази до нагомилавања података.

Будући да се ради о не тако великој количини података просјечно вријеме слања података, које износи 1,12 секунди са стандардном девијацијом од 413 милисекунди, је и даље приближно просјечном времену слања у нормалним условима. Ако посебно посматрамо период прије пада примарне компоненте, гдје је просјечно вријеме 965,7 милисекунди са стандардном девијацијом од 314 милисекунди, и период након њеног пада, гдје је просјечно вријеме 1,21 секунда са стандардном девијацијом од 433 милисекунди, ту примјетимо одређену разлику. Та разлика је изазвана временом потребним да се пренесе први пакет података након што је редудантна компонента преузела улогу примарне.

Када се посматра случај слања 20.000 података са 5 инстанци, може да се примјети значајнија временска разлика. Када посматрамо само период прије пада примарне компоненте, видимо да је просјечо вријеме преноса података 2,26 секунди са стандардном девијацијом од 612 милисекунди, док је након њеног пада то вријеме много веће, 5,18 секунди са стандардном девијацијом од 743 милисекунде. Као и у претходном примјеру, ово кашњење након пада је изазвано обимом података у првом пакету. Тај пакет је до 3 пута већи од свих послатих прије и каког њега.

3.3. Дуплирање података

За овај случај тестирања је извршено слање 15.000 података са 5 инстанци компоненте *Producer*.

У овом случају примарна инстанца компоненте *Worker* је „оборена“ у моменту када је податке које је скупљала секунду прослиједила даље, али није стигла да јави редудантној компонентни идентификациони број последњег послатог податка. Након преузимања примарне улоге, редудантна компонента, након истека временског периода од једне секунде, даље прослеђује све податке који се налазе код ње. Управо због овога долази до дуплирања података на пријемној страни, а из примјера можемо видјети да је послато 60.000 дуплираних података.

Дуплирање података у овом случају са собом повлачи проблем из примјера описаних у поглављу 3.2. Када посматрамо период прије пада примарне компоненте, просјечно вријеме преноса података је 1.46 секунди са стандардном девијацијом од 375 милисекунди, а након њеног пада и преузимања њене улоге од стране редудантне компоненте, просјечно вријеме је 5,09 секунди са стандардном девијацијом од 701 милисекунде. У овом примјеру, просјечно вријеме за пренос података у току цјелокупног рада је 3,63 секунде са стандардном девијацијом од 1,87 секунди.

4. ЗАКЉУЧАК

Примјена вруће резерве умногоме повећава поузданост система. Обезбјеђује његов континуалан рад у великом броју случајева. Наравно, за неке случајеве се та поузданост не може осигурати. Највећи недостатак редудансе је могућност отказа додатних компоненти које се користе за активирање компоненте из стања резерве када је то потребно.

Основна предност се огледа управо у томе што се падом примарне инстанце у било ком моменту обезбјеђује да сви подаци стигну на одредиште. Свака предност са собом носи и одређене недостатке. У овом случају ти недостаци су представљени дуплирањем података, као и додатним кашњењем приликом испоруке. Међутим, када се предност упореди са дуплирањем и кашњењем података, јасно је да та два недостатка могу да се занемаре. Посљедице које они изазивају су далеко мање од посљедица изазваних губитком података у системима са критичном мисијом.

5. ЛИТЕРАТУРА

[1] Israel Koren and C. Mani Krishna, „*Faule-Tolerant Systems“*, Morgan Kaufmann, 2020.

[2] Andrew S. Tanenbaum, „*Distributed Systems Principles and Paradigmas“*, Prentice-Hall, 2007.

[3] Z. Wang, Y. Chen, W. Men and R. Kang, “Failure behavior analysis of hot standby system based on BDD method”, *Safety and Reliability–Safe Societies in a Changing World* , pp. 2441-2448, CRC Press, 2019.

[4] Suprasad V. Amari and Glenn Dill, “A new method for reliability analysis of standby system”, IEEE *Annual Reliability and Maintainability Symposium,* pp. 417-422, Relex Software Corporation, 2009.

[5] W. Kuo, V.R.Prasad, F.A.Tillman, C.L.Hwang, *“Optimal reliability Design: Fundamentals and Applications”*, Cambridge university press, 2001.

[6] David W. Colt, Cold-standby redundancy optimization for nonrepairable systems, *Iie Transactions*, Vol. 33, No.6, pp. 471-478, 2001.

[7] Suprasad V. Amari, Hoang Pham and Ravindra B. Misra, “Reliability Characteristics of k-out-of-n Warm Standby Systems”, *IEEE Transactions on Reliability*, Vol.61, No.4, pp. 1007-1018., 2012.

Кратка биографија:

Ивана Ћуковић рођена је у Корану 1995. године. Дипломирала је на Факултету техничких наука из области Електротехнике и рачунарства – Примјењено софтверско инжењерство 2018. године.
контакт: icukovic14@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4:621.395
DOI: https://doi.org/10.24867/12BE42Keca

TEHNIKE UMETNIČKOG MENJANJA SLIKE PRIMENOM DUBOKOG UČENJA

DEEP LEARNING TECHNIQUES OF ARTISTIC IMAGE MODIFICATION

Milan Keča, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu su predstavljene metoda umetničkog menjanja stila slika. Metode se baziraju na upotrebi pretreniranih konvolucionih neuronskih mreža za prepoznavanje sadržaja i stila slika. Predstavljeni su iterativni algoritmi, prenos stila i Deep Dream, kao i njihova unapređenja. Analizirano je ponašanje algoritama menjajući parametre algoritma i komentarisani su rezultati. Implementacija svih algoritama je izvršena u programskom jeziku Python i paketu Tensorflow.*

Ključne reči: *Algoritam prenosa stila, Deep Dream, konvolucione neuronske mreže*

Abstract – *This paper describes the methods of artistic image transformation. The methods rely on convolutional neural networks trained for image recognition to differentiate between image style and content. The paper describes iterative algorithms such as style transfer and Deep Dream, as well as improvements of these original algorithms. The behavior of these algorithms is analyzed by experimenting with different parameter configurations. All algorithms were implemented using the Python programming language and Tensorflow package.*

Keywords: *Style transfer, Deep Dream, convolutional neural networks*

1. UVOD

U ovom radu izvršena je implementacija i poređenje različitih metoda generisanja sadržaja slika i prenos različitih vizuelnih atributa. Vizuelnim atributima se smatraju informacije poput boja, nijanse, teksture i drugi oblici. U radu su predstavljeni metodi prenosa stila sa jedne slike na drugu, kao i *Deep Dream* – algoritam koji generiše kreativne, "halucinogene", efekte na slici iterativnim pojačavanjem šablona koje je mreža trenirana da prepoznaje.

Prikaz semantičkog sadržaja slike u različitim stilovima je veoma izazovan zadatak procesiranja slika. Najteži korak predstavlja eksplicitno razdvajanje semantičkih informacija slike od stila kojim je taj sadržaj predstavljen. Ovaj problem je olakšan upotrebom konvolucionih neuronskih mreža. Tehnika prenosa stila bazirana na konvolucionim neuronskim mrežama podrazumeva korišćenje različitih reprezentacija slike dobijenih iz konvolucionih neuronskih mreža optimizovanih za prepoznavanje objekata.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Jelena Slivka, vanr. prof.

Na taj način se efikasno mogu izdvojiti primitivnije informacije slike, kao što su linija, nijansa, boja i apstraktinije informacije slike, kao što su objekti koji se nalaze na slici. Algoritam prenosa stila omogućuje razdvanje informacija slike i njihovo ponovno sklapanje. Rezultat algoritma su slike na kojima se nalazi željeni sadržaj nacrtan željenim stilom [1].

Deep Dream je eksperiment koji vizualizuje šablone naučene od strane neuronske mreže. Slično detetu koje gleda u oblake i pokušava da interpretira nasumične oblike, *Deep Dream* interpretira i pojačava šablone koje pronalazi na slici. Algoritam je razvijen od strane inženjera *Google-a* koji su koristili pretreniranu *Inception* konvolucionu mrežu [2].

Za konstrukciju *Deep Dream* eksperimenata može se koristiti bilo koja konvoluciona neuronska mreža obučena na slikama. Umesto da eksplicitno tražimo šablone koje želimo mreža da interpretira i pojača, puštamo mrežu da to uradi sama izborom odgovarajućih konvolucionih slojeva.

2. UPOTREBA ALGORITMA PRENOSA STILA

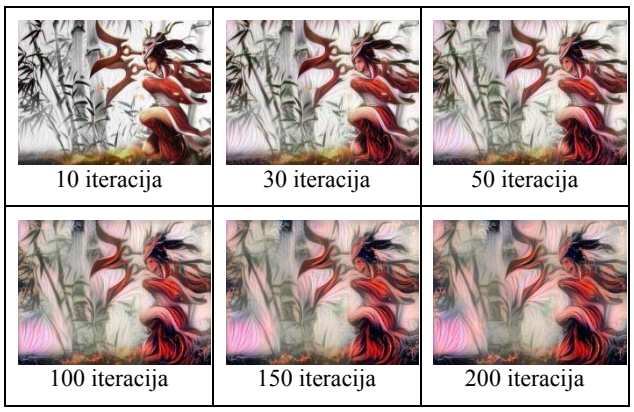
Napreci dubokih konvolucionih mreža su doveli do toga da je moguće izdvojiti apstraktnije, semantičke osobine slike. Razdvajanje sadržaja od stila na slici je neophodno za prenos stila sa jedne slike na drugu. U radu [1] je u ovu svrhu korišćena pretrenirana VGG-19 mreža [3], sa napomenom da je za ovaj zadatak moguće koristiti bilo koju konvolucionu neuronsku mrežu istreniranu za prepoznavanje objekata na slikama.

Na slici 1 prikazan je primer primene prenosa stila.

Slika 2. prikazuje međurezultate ovog iterativnog procesa, odnosno, slike generisane nakon određenog broja iteracija algoritma prenosa stila.

U ovom poglavlju ćemo analizirati kako na algoritam prenosa stila utiču faktori poput izbora algoritma za iterativnu optimizaciju slike (poglavlje 2.1) i izbora sloja konvolucione neuronske mreže za koji se pretpostavlja da efektivno reprezentuje stil kojim je slika nacrtana.

Slika 1. *Primer prenosa stila. Iterativno se prenosi stil sa slike stila (desno), na sliku sadržaja (levo) i dobija se rezultujuća slika (sredina).*



Slika 2. *Prikaz iterativnih izmena slike prilikom prenosa stila*

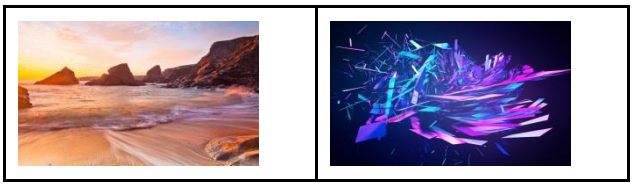
2.1. Izbor algoritma za iterativnu optimizaciju slike

U nastavku je predstavljeno nekoliko primera prenosa stila u kojima se koriste različiti algoritmi optimizacije za iterativno menjanje ciljne slike.

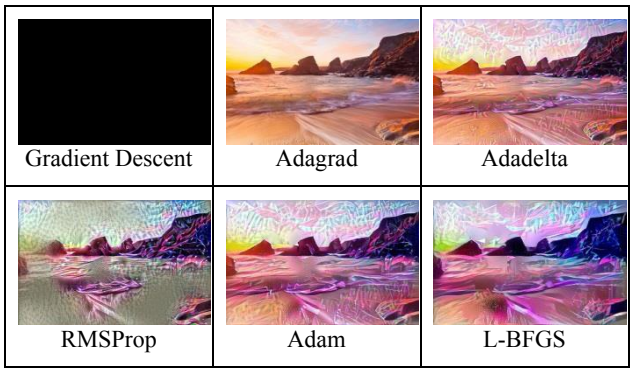
U eksperimentima izvršenim u ovom radu isprobani su sledeći algoritmi: *Gradient Descent*, *Adagrad*, *Adadelta, RMSProp, Adam i* L-BFGS.

Iako se za optimizaciju ciljne slike može koristiti bilo koji algoritam, u eksperimentima isprobanim u ovom radu su najbolje rezultate proizveli L-BFGS i *Adam* (slika 3). U prikazanom eksperimentu, svi parametri prenosa stila su fiksirani na iste vrednosti, a menja se samo optimizacioni algoritam. Na slici 3 su prikazane slike korišćene za sprovedene eksperimente.

U sprovedenim eksperimentima prikazanim na slici 4 pokazalo se da Adam i L-BFGS algoritmi daju najbolje rezultate u najmanjem broju iteracija. U ovom kontekstu, rezultati su najbolji u smislu da su šabloni slike stila u najvećoj prebačeni na sliku sadržaja, uz najmanji šum. Rezultati, naravno, zavise i od podešavanja specifičnih parametara svakog od algoritama optimizacije, kao i ostalih parametara algoritma za prenos stila, koji su u ovom eksperimentu bili fiksni.



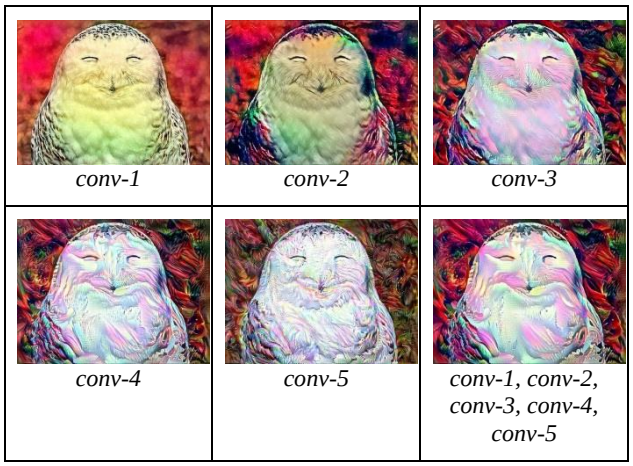
Slika 3. *Prikaz slike sadržaja (levo) i slike stila (desno), korišćene za poređenje optimizujućih algoritama.*



Slika 4. *Poređenje različitih optimizera za prenos stila.*

2.2. Izbor slojeva za reprezentaciju stila

U radu [1] su iz pretrenirane VGG-19 mreže korišćeni sloj *conv4_2* za reprezentaciju sadržaja i slojevi *conv1_1*, *conv2_1*, *conv3_1*, *conv4_1*, *conv5_1* za reprezentaciju stila. U sledećim eksperimentima će biti prikazani rezultati algoritma prenosa stila upotrebom različitih slojeva za reprezentaciju stila. Ostali parametri algoritma će biti isti za sve eksperimente.



Slika 5. *Poređenje različitih reprezentacija stila (sloja konvolucione neuronske mreže) za prenos stila.*

Iz postignutih rezultata (slika 5) se može zaključiti da se korišćenjem prvog sloja efektivno prenose samo boje, dok se teksture menjaju u veoma maloj meri. Upotrebom dubljih slojeva postiže se efekat detaljnijih tekstura, ali se gube informacije o boji slike stila. Korišćenjem svih pet slojeva postižu se detaljne teksture i očuvanje boje slike stila.

4. OČUVANJE ORIGINALNIH BOJA U PRENOSU STILA

Tehnika prenosa stila će neizbežno, pored tekstura, preneti i boje sa slike stila na sliku sadržaja. U određenim situacijama to može biti nepoželjno. U tom slučaju se kao ekstenzija originalnom algoritmu prenosa stila može koristiti tehnika prenosa boja sa jedne slike na drugu [4].

U ovom poglavlju su predstavljeni eksperimenti prenosa stila, uz očuvanje originalnih boja slike sadržaja. Očuvanje originalnih boja u prenosu stila je motivisano opažanjem da je vizuelna percepcija mnogo osetljivija na promene u osvetljenosti nego na promene u boji [4].

Modifikacija boja rezultujuće slike podrazumeva da se izračunaju kanali osvetljenosti (*luminance*) slike sadržaja L_C i slike stila L_S koristeći YIQ prostor boja. Zatim se primenjuje standardni algoritam prenosa stila kako bi se dobila rezultujuća slika i kanal osvetljenosti L_T. Informacije o boji slike sadržaja su predstavljene kanalima I i Q i kombinuju se sa kanalom Y rezultujuće slike kako bi se dobila krajnja slika [4].

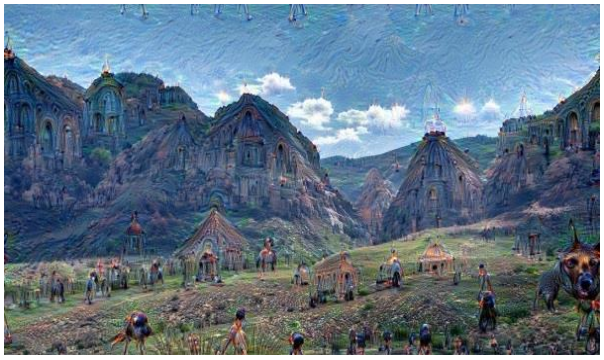
Pored YIQ prostora boja, mogu se koristiti i drugi, kao što su YUV, LUV, Lab, YCrCb, jer na određen način opisuju osvetljenost na slici. Na slici 6 prikazane su ulazne slike sadržaja i stila, dok su na slici 7 prikazani rezultati prenosa stila bez očuvanja boja i sa očuvanjem boja.

Slika 6. *Slika sadržaja (levo) i slika stila (desno)*

Slika 7. *Rezultat prenosa stila bez očuvanja originalnih boja slike sadržaja (levo) i sa očuvanjem originalnih boja (desno)*

5. UPOTREBA *DEEP DREAM* ALGORITMA

Deep Dream [2] je eksperiment koji vizualizuje šablone naučene od strane neuronske mreže. Naglašava i poboljšava prepoznate šablone na slici. To čini propuštajući sliku kroz mrežu, računajući gradijente aktivacija određenih slojeva i, na kraju, optimizujući sliku tako da se maksimizuju aktivacije istih slojeva. Maksimizacijom aktivacija izabranih slojeva slike, pojačavaju se šabloni koje mreža prepoznaje. Na slici 8 prikazan je rezultat *Deep Dream* algoritma.



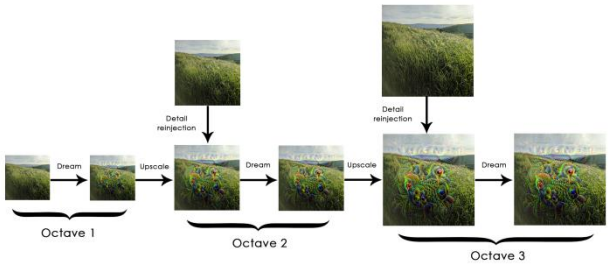
Slika 8. *Rezultat Deep Dream algoritma* [2].

Osnovni oblik *Deep Dream* algoritma [2] proizvodi zadovoljavajuće rezultate, ali postoji i nekoliko problema. Rezultujuća slika je puna šuma i rezolucija je niska. Pored toga, izgenerisani šabloni se javljaju na istom nivou granularnosti [5].

To znači da svi dodati šabloni na slici imaju sličan nivo detalja i slične su boje i veličine. Navedeni problemi se mogu rešiti optimizacijom na različitim skalama slike i ubacivanjem izgubljenih detalja iz originalne slike.

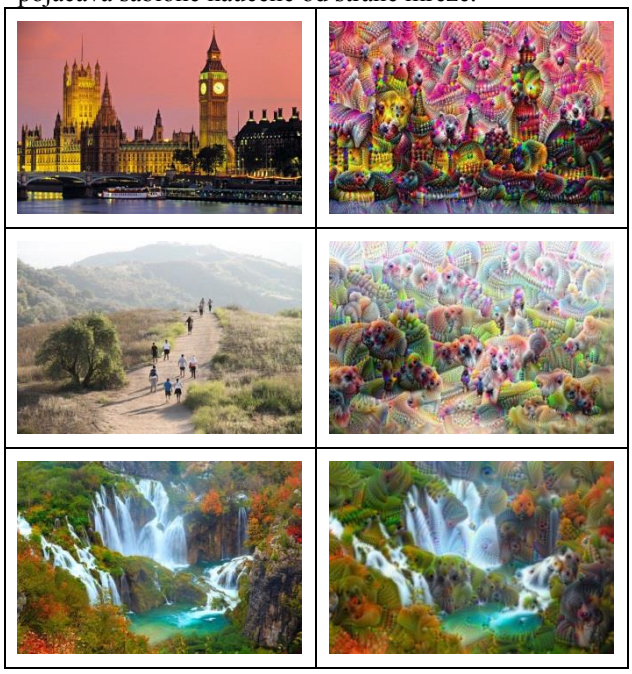
Svaka skala predstavlja određenu rezoluciju originalne slike. To omogućava da se šabloni izgenerisani na nižim skalama budu inkorporirani u šablone na višim skalama i popunjeni dodatnim detaljima (slika 9).

To se postiže tako što se uradi *gradient ascent* za modifikaciju slike i zatim poveća veličina slike, što predstavlja jednu oktavu. Proces se ponavlja za određeni broj oktava [2].

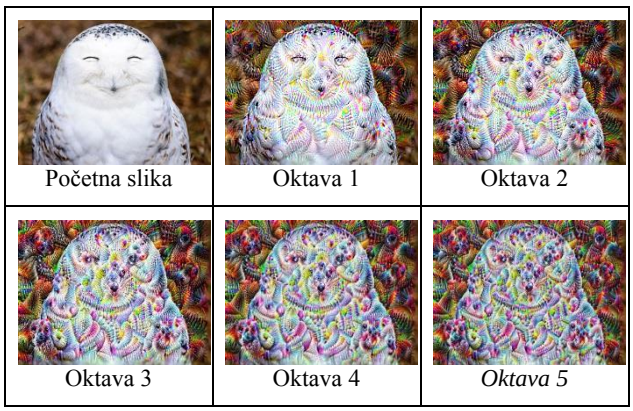


Slika 9. *Oktave Deep Dream algoritma.*

S obzirom da je *Inception V3* [6] mreža trenirana na slikama veličine 299 x 299, najbolji rezultati dobijaju se na slikama u opsegu veličine 300 x 300 ili 400 x 400, jer algoritam podrazumeva i skaliranje slika. Bez obzira na to, algoritam se može upotrebiti na slikama bilo koje veličine. Na rezultujućim slikama će se na kraju najviše videti šabloni očiju pasa ili perja ptica, jer su slike sa takvim šablonima najzastupljenije u skupu slika na kojima je trenirana mreža. Na slici 10 prikazani su rezultati *Deep Dream* algoritma dobijeni koristeći 5 oktava. Na slici 11 prikazano je kako svaka oktava dalje pojačava šablone naučene od strane mreže.



Slika 10. *Primeri upotreba algoritma Deep Dream*



Slika 11. *Prikaz rezultujuće slike nakon svake oktave*

6. ZAKLJUČAK

U ovom radu predstavljene su metode za kreativno generisanje slika pomoću konvolucionih neuronskih mreža. Pored toga, predstavljeno je kako eksplicitno razdvajanje semantičkih informacija slike od stila kojim je taj sadržaj predstavljen omogućuje bolje razumevanje šablona koje neuronske mreže nauče prilikom treniranja, ali i otvara nov prostor za manipulisanje slikama. Motivacija za razvoj ovih algoritama je, s jedne strane, bila istražiti i bolje razumeti na koji način funkcionišu konvolucione neuronske mreže i šta mogu da nauče prilikom treniranja. S druge strane, pokazano je na koji način se mogu koristiti neuronske mreže za umetničke potrebe.

U radu su pokazani primeri slika generisani algoritmom prenosa stila [1] i *Deep Dream* algoritmom [2]. Oba algoritma se zasnivaju na upotrebi različitih slojeva konvolucionih mreža za dobijanje različitih reprezentacija slika. Svaka reprezentacija slike se koristi kako bi se uneli novi detalji u sliku. Prikazani su rezultati oba algoritma, kao i kako njihove različite konfiguracije utiču na rezultate.

Arhitekture mreža korišćene kod oba algoritma su konvolucione mreže trenirane na *Imagenet* [7] skupu podataka. Za algoritam prenosa stila, korišćena je VGG-19 [3] mreža, dok je za *Deep Dream* korišćena *Inception-V3* mreža [6]. Obe mreže mogu da prepoznaju različite nivoe kompleksnosti šablona slike, počevši od primitivnih (linija, tekstura i osnovnih geometrijskih oblika), do čitavih objekata na slici. Iz tog razloga su ove mreže veoma zahvalne za manipulisanje slikama.

Osnovni nedostatak ovog rada jeste korišćenje relativno skromnog broja primera uz relativno mali broj iteracija algoritma. Da bi se proverilo da li izvedeni zaključci važe u opštem slučaju, potrebno je testirati algoritame na većem broju slika uz veći broj iteracija. Nemogućnost takvih eksperimenata u ovom istraživanju proizilazi iz zahtevnosti samih algoritama, kako memorijskih, tako i procesorskih.

7. LITERATURA

[1] L. A. Gatys, A. S. Ecker, M. Bethge, A Neural Algorithm of Artistic Style, 2015.

[2] Google AI Blog. 2015. Inceptionism: Going Deeper Into Neural Networks. [online] Available at: <https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>

[3] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recongition, 2015.

[4] L. A. Gatys, M. Berthge, A. Hertzmann, E. Shechtman, Preserving Color in Neural Artistic Style Transfer, 2016.

[5] TensorFlow. 2020. Deepdream | Tensorflow Core. [online] Available at: <https://www.tensorflow.org/tutorials/generative/deepdream>

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Rethinking the Inception Architecture for Computer Vision, 2015.

[7] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.

Kratka biografija:

Milan Keča rođen je u Subotici 1993. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Tehnike umetničkog menjanja slike odbranio je 2020.god. kontakt: vonum.mk.@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/12BE44Smiljanic

MODELOVANJE SISTEMA ZA POTRAŽIVANJE, DOBAVLJANJE I ŠTAMPANJE SERIJSKIH KODOVA KORIŠĆENJEM ACTIVITI FRAMEWORK-A

MODELING OF A SYSTEM FOR DEMANDING, SUPPLYING AND PRINTING OF SERIAL CODES USING ACTIVITI FRAMEWORK

Nikola Smiljanić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je prikazano modelovanje sistema za potraživanje, dobavljanje i štampanje serijskih kodova uz korišćenje Activiti framework-a. Biće navedena specifikacija sistema zajedno sa svim koracima koje je potrebno izvršiti da bi se došlo do cilja. U radu će se, takođe, definisati osnovni elementi sistema kao i osnovni pojmovi activiti framework-a.*

Ključne reči: *Serijski kodovi, štampanje, activiti, proces, agregacija, serijalizacija.*

Abstract – *The paper presents a modeling of a system for demanding, supplying and printing serial codes using the Activiti framework. The system specification will be listed along with all the steps that need to be performed to reach the goal. The paper will also define the basic elements of the system as well as the basic concepts of the activity framework.*

Keywords: *Serial codes, printing, Activiti, process, aggregation, serialization.*

1. UVOD

Mnoge velike multi-nacionalne kompanije u okviru svoje proizvodnje žele da svoje proizvode obeleže jedinstvenim kodovima za potrebe procesa *Track & Trace*-a. Pojam *Track and trace* predstavlja sistem u kome svaki, kodom označeni, proizvod može biti praćen. Pod tim se podrazumeva njegova prošla i sadašnja lokacija ali i plan na kojoj lokaciji će da završi. Na osnovu koda i informacija koje su s njim povezane, tačno može da se utvrdi kada je proizvod kreiran, spakovan, napustio skladište, dostavljen u maloprodajni objekat, prodat, kome je prodat. *Track and trace* stvara bolje uslove za poslovanje, za jedinstvenu identifikaciju i precizno praćenje proizvoda. Da bi se kodovi za proizvode uopšte odštampali na iste, potrebno ih je dobaviti. Ti kodovi su često nazvani i serijski brojevi koji su jedinstveni za svaki proizvod.

Činioci ovog poslovnog sistema jesu kompanija koja ima proizvodnju (proizvođač) i kompanija koja je pružalac usluga *Track and Trace*-a. Sam proces se sastoji iz toga da kompanija koja ima proizvodnu liniju traži kompaniju koja joj može pružiti usluge *Track and Trace*-a. Nakon uspešnog pronalaska, sklapa ugovor sa kompanijom koja će joj isporučiti takav sistem. Na osnovu porudžbine, se generišu serijski kodovi na određen način i štampaju se na ambalažu.

Proizvodi se nakon toga pakuju po određenom principu koji je takođe definisan u porudžbini. Ovakav sistem se može modelovati kao jedan poslovni proces koji za cilj ima uspešno odštampane kodove i zapakovane proizvode.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Miroslav Zarić.

2. KORIŠĆENE TEHNOLOGIJE

Tehnologije i alati koji su korišćeni prilikom razvijanja rešenja su *Spring Boot* okruženje uz *Maven*. Za rad sa bazom podataka koristi se *MySQL* uz MySQL *Wokrbench* dok se za modelovanje poslovnog procesa koristi *Activiti framework* uz *BPMN 2.0* notaciju.

2.1 Activiti Framework

Activiti je platforma za upravljanje poslovnim procesima napisana u *Java* programskom jeziku. Poslovni proces predstavlja niz pojedinačnih aktivnosti čije izvršavanje ima za cilj stvaranje nekog proizvoda ili usluge koja ima vrednost za određenu grupu korisnika. Procesi, sami ili u interakciji sa drugim procesima, treba da na efektivan i efikasan način doprinesu stvaranju vrednosti za korisnika u okviru neke organizacije.

Sistem za upravljanje poslovnim procesima je generički softver, koji obezbeđuje koordinisano izvršavanje poslovnog procesa na osnovu eksplicitne reprezentacije poslovnog procesa.

Ta reprezentacija može biti tekstualna i formalna. Formalna reprezentacija je u suštini dijagram, odnosno model poslovnog procesa koji se sastoji od skupa aktivnosti i ograničenja za njihovo izvršavanje. Instanca poslovnog procesa predstavlja jedan konkretan slučaj izvršavanja datog modela (u poslovnom okruženju neke organizacije).

Instanca poslovnog procesa sastoji se od instanci aktivnosti. Sistem za upravljanje poslovnim procesima može se koristiti za uspostavljanje kontrole nad poslovnim procesom [1,2].

2.2 Bpmn 2.0

BPMN 2.0 je standard za grafički prikaz dijagrama prilikom kreiranja i modelovanja poslovnih procesa. Kreiran je od strane *Business Process Management Initiative (BPMI)* 2004. Zvanična specifikacija je objavljena u februaru 2006. godine dok je verzija 2.0

razvijena 2010. BPMN pruža grafičku reprezentaciju modela.
BPMN nastoji da pokrije sve nivoe apstrakcije pri modelovanju. Nudi raznovrsnu ali i jednostavnu notaciju. Ideja je da se grafički oblici približe funkcionalnostima koje predstavljaju, stoga i obični korisnici vrlo lako mogu da steknu uvid šta određeni proces obavlja samim pogledom na dijagram.
Jednostavan, lak za upotrebu i lako čitljiv ne samo *developer*-ima već i ljudima koji su manje tehnički potkovani što im omogućava detaljniji uvid i mogućnost učestvovanja prilikom razvijanja samog modela.
Procesi su opisani dijagramima sa nizom grafičkih elemenata, ovakva vizuelna prezentacija olakšava praćenje logike i toka procesa. Grafički elementi se klasifikuju po kategorijama [3,4].

3. OPIS SISTEMA I IMPLEMENTACIJA REŠENJA

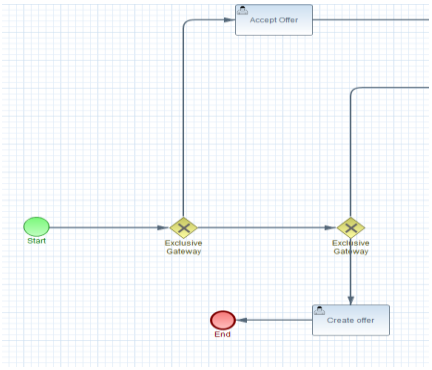
Pre detaljnog opisa sistema i njegovih procesa, potrebno je upoznati se sa osnovnim pojmovima samog sistema: *GTIN - Global Trade Item Number* predstavlja identifikacioni broj za neku vrstu proizvoda. Primera radi, sve flaše Koka-Kole od 0.5 l imaju isti GTIN. *Item master* predstavlja konačni proizvod (materijal + ambalaža).
Packaging Order (PO) - zahtev za pakovanje za određeni produkt. *Packaging order* mora da sadrži informaciju o načinu pakovanja dok *item master* sadrži podatak o nivou pakovanja.
Nivo pakovanja može biti - *Product, Carton, Bundle, Case. Case* može da sadrži više *Bundle*-ova, odnosno, *Bundle* može da sadrži više *Carton*-a. *Carton* može da sadrži više *Product*-a dok sam *Product* predstavlja jedan proizvod.

Način štampanja kodova može biti različit. Postoje tri tipa: serijalizacija, agregacija, kodifikacija. Serijalizacija sadrži *GTIN*, ali svaki pojedinačni produkt dobija poseban serijski broj na samo jednom nivou. Agregacija predstavlja serijalizaciju na više nivoa.

Može da se pakuje u *Carton,Bundle,Case*(pri čemu *Carton*-i se pakuju u *Bundle*, a *Bundle*-ovi u *Case*). Kodifikacija sadrži isključivo *GTIN*. Ne zahteva posebne serijske kodove, tako da se na sve proizvode štampa jedan isti *GTIN* [5].

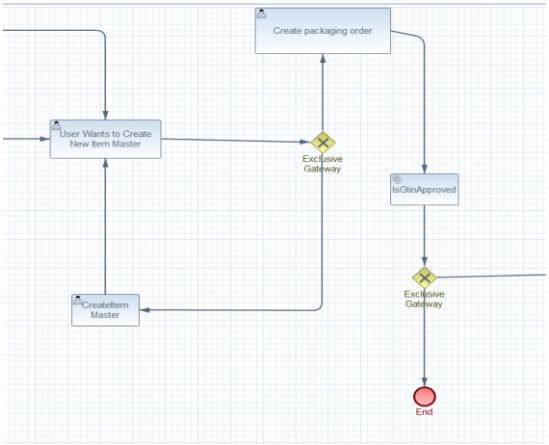
3.1. Modelovanje procesa

U narednih nekoliko koraka biće objašnjeno kako je implementirano rešenje. Podeljeno je u segmente. Započinje se sklapanjem ponude između korisnika koji je zadužen za upravljanje sistemom i proizvođača Nakon toga je potrebno popuniti forme za kreiranje *Item master-a i Packaging Order*-a. Sledi okidanje procesa za zahtevanje kodova posle kog dolazi deo sa štampanjem i pakovanjem.
Prvi segment jeste sklapanje ponude između korisnika koji je zadužen za upravljanje sistemom i firme koja potražuje kodove. Korisnik koji predstavlja kompaniju proizvođača bira da li želi da radi sa novim kompanijama ili sa onima gde već postoji istorija saradnje. Ukoliko želi novu, mora da pošalje ponude kompanijama iz željene industrije.



Slika 3-1-1 Dijagram za ponude (prihvatanje i odbijanje)

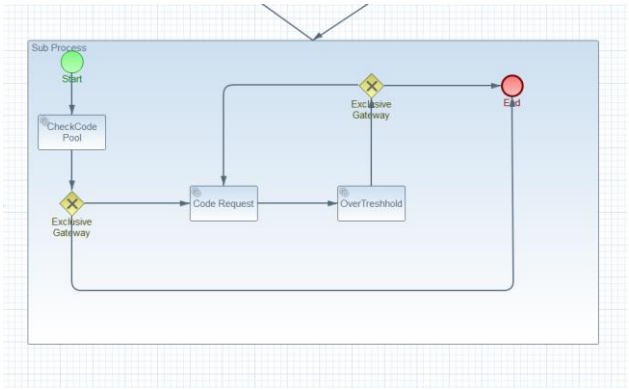
Nakon uspostavljanja saradnje dolazi do kreiranja neophodnih činilaca u sistemu (*PO* i *Item master*). Na slici 3-1-2 se može videti deo dijagrama koji opisuje ovaj segment. Proces prvo nailazi na zadatak (engl. *Task*) "*User wants to create new item master*". Ukoliko potvrdi, prelazi u stanje "*Create Item Master*". U suprotnom ide na "*Create Packaging order*". Ovakav tok procesa je osmišljen ukoliko bi korisnik zahtevao agregaciju. Tada se javlja potreba da se unese nekoliko *Item Mastera* (npr. u prvom koraku se potvrdi "*User wants to create new item master*", kreira se *Item master* za *Carton*, nakon toga je ponovo potrebno potvrditi ukoliko treba da se kreira *Item master* za *Case ili Bundle*).



Slika 3-1-2 Item master i Packaging order

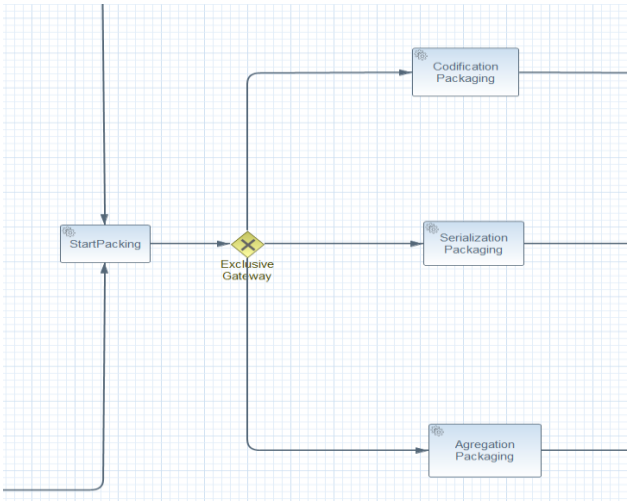
Ukoliko je način pakovanja označen kao serijalizacija ili agregacija, potrebno je da se obavi zahtev za izdavanje koda (engl. *Code Request)*, gde se dobavljaju serijski kodovi koji se štampaju na ambalažu proizvoda. *Code Request* se vrši za svaki *Item master* koji je odabran za *Packaging Order*.
Potproces za zahtevanje kodova se može videti na slici 3-1-4. Okida se tako što se nakon *HasLevelForRequest* dobije informacija da postoji nivo (*Item master*) za koji nije odrađen *CodeRequest*. Ulazi se u stanje *Check Code Pool* gde se proverava da li u skladištu svih kodova postoje kodovi za taj *Item Master*. U tom skladištu se čuvaju neiskorišćeni kodovi za eventualnu kasniju upotrebu. Često se dešavaju situacije kada određeni korisnik sarađuje više puta sa jednom firmom za iste proizvode, stoga često koristi iste kodove. Otud potreba za skladištenjem kodova. Ukoliko postoji dovoljno kodova u skladištu, potproces se završava. Ako ne postoji, onda se radi *Code Request*.

Po jednom zahtevu se traži količina kodova koja je definisana u *Item Masteru*. I zahteva se do onog trenutka dok se ne dostigne postavljena vrednost - prag (engl. *Threshold)* nakon čega se zahtev šalje još jednom. Ovaj mehanizam je potreban zato što prilikom pakovanja postoji mogućnost da dođe do neuspešnog lepljenja koda ili nemogućnosti da se nalepljeni kod skenira, nakon toga mora da se nalepi novi. Da ne bi došlo do situacije da prilikom pakovanja fali kodova, naručena količina je malo veća nego što je potrebno. Ostatak će svakako ostati u skladištu za kodove. Na dijagramu je jasno prikazana "petlja" zahtevanja kodova nakon koje se potproces završava.



Slika 3-1-3 Code Request

Kada su kodovi uspešno smešteni u bazu podataka, može da se počne sa štampanjem i pakovanjem. Na sledećoj slici, 3-1-4 se nalazi primer dijagrama.



Slika 3-1-4 Štampanje i pakovanje

Servisni zadatak "*Start Packaging*" služi za to da usmeri dalji tok procesa. U zavisnosti od tipa pakovanja, proces ulazi u izvršavanje jedne od tri aktivnosti nakon grananja. *Codification Packaging, Serialization Packaging, Agregation Packaging*, svaka od njih obavlja različitu vrstu pakovanja. *Codification packaging* preuzima *GTIN* nakon čega vrši štampanje istog na proizvod (ambalažu). *Serialization packaging* preuzima iz code pool-a kodove za taj *PO*, tj za taj *Item Master* i štampa serijske kodove na ambalažu. *Agregation packaging* preuzima kodove koji su stigli u *code pool* i pravi raspored po nivoima u zavisnosti koliko ih je navedeno u *PO*. Kodovi na višem nivou moraju imati podređene (engl. *Children*-e), odnosno kodovi na nižim nivoima moraju biti vezani za neki „roditeljski" kod.

Nakon što su uspešno spakovani, šalju se korisniku na email kao izveštaj u zadatku "*SendMail*". Sledeći korak je pakovanje u kutije koju obavljaju radnici i nakon što su kodovi poslati korisniku, radnik firme unosi potvrdu da su kodovi na putu jednostavnim potvrđivanjem i time se proces završava.

4. ZAKLJUČAK

U radu je prikazana detaljna upotreba i korišćenje sistema za potraživanje, dobavljanje i štampanje kodova koji uz određene izmene i prilagođavanje može da se primeni u kompanijama koji kao deo proizvodnje, rade sa kodovima. Takođe je primenljivo u serijalizaciji i *track and trace* domenu pri farmaceutskoj ili prehrambenoj industriji. Gde će se na osnovu koda pratiti svaki proizvod i znaće se njegova putanja kao i trenutna lokacija. Ovo je korisno kada u *real-time-database* imamo lokaciju ili praćenje određenog proizvoda pa se tačno može utvrditi kada je kreiran, spakovan, napustio skladište, dostavljen u maloprodajni objekat, prodat itd...

Stvoriće se bolji uslovi za poslovanje za jedinstvenu identifikaciju, precizno snimanje i automatsko deljenje vitalne informacije o proizvodima i lokacijama. Takođe zbog svih prednosti koje Activiti i BPMN 2.0 nude, biće olakšano učestvovanje u kreiranju i razvijanju poslovnog procesa netehničkim korisnicima. Standardi u lancu snabdevanja štede vreme i novac smanjenjem papirne dokumentacije i administracije. Oni omogućavaju da se procesi unutar najvećih svetskih industrija brže odvijaju. Ovakvi sistemi stvaraju zajednički temelj za poslovanje, za jedinstvenu identifikaciju, precizno snimanje i automatsko deljenje vitalne informacije o proizvodima.

4. LITERATURA

[1] https://livebook.manning.com/book/activiti-in-action/chapter-1/14 (pristupljeno u julu 2020.)
[2] Tijs Rademakers, "Activiti in Action: Executable Business Processes in BPMN 2.0" 2012.
[3] https://www.visual-paradigm.com/guide/bpmn/what-is-bpmn/ (pristupljeno u julu 2020.)
[4] https://www.activiti.org/userguide/#bpmn20 (pristupljeno u julu 2020.)
[5] https://adents.com/article-aggregation-exploring-the-pharmaceutical-supply-chain-parent-child-relationship-6272.html(pristupljeno u avgustu 2020.)

Kratka biografija:

Nikola Smiljanić rođen je u Loznici 1994. godine. Master studije upisao 2017/2018 godine na smeru Računarstvo i automatika na Fakultetu tehničkih nauka u Novom Sadu. Master rad odbranio je 2020.god.
kontakt: nikola94nwa@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE45Gavrilovic

DODELA RADNIH ZADATAKA U POSLOVNOM PROCESU UPOTREBOM *PHARO* PROGRAMSKOG JEZIKA

ASSIGNMENT OF WORK TASKS IN THE BUSINESS PROCESS USING THE PHARO PROGRAMMING LANGUAGE

Nemanja Gavrilović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *u radu su opisani poslovni procesi, alati koji se koriste za njihovo modelovanje i programski jezici Smalltalk i Pharo. Prikazani su šabloni za dodelu zadataka učesnicima u procesu. Predstavljena je implementacija, uz prikaz klasa i njihovih metoda kojima se dolazi do željenog rešenja.*

Ključne reči: *Poslovni procesi, šabloni za dodelu zadataka, Pharo programski jezik*

Abstract – *The thesis describes the business processes, the tools used for their modeling and the programming languages Smalltalk and Pharo. Templates for assigning tasks to participants in the process are shown. The implementation is presented, with a presentation of classes and their methods by which the desired solution is reached.*

Keywords: *process engine, assigning templates, Pharo*

1. UVOD

Poslovni proces sastoji se od skupa aktivnosti koje se koordinisano izvršavaju u odgovarajućem organizacionom i tehnološkom okruženju. Ove aktivnosti zajedno dovode do poslovnog cilja. Svaki poslovni proces izvršava jedna organizacija, ali on može imati interakcije sa poslovnim procesima drugih organizacija [1]. Učesnici u poslovanju organizacije obavljaju određene zadatke. Ti zadaci mogu im biti dodeljeni na različite načine. Zadatak ovog rada je kreiranje proširenja za *NewWave* softver, koje će omogućiti rad sa korisnicima i njihvoim grupama, kako bi se omogućila dodela zadataka.
Rešenje je potrebno implementirati u *Pharo* programskom jeziku.

2. POSLOVNI PROCESI

Poslovni proces može se definisati i kao niz aktivnosti koje pokreće određeni događaj (ili više njih), a čiji je zadatak ostvarivanje zajedničkog cilja. Proces koristi resurse prilikom ostvarivanja definisanog cilja, podložan je spoljašnjim uticajima i njime treba upravljati.

Kako se obim proizvodnje povećavao i usložnjavao i tehnologija napredovala, tako su procesi postali veći i zahtevniji. Današnji, moderni procesi, imaju svoj životni ciklus.

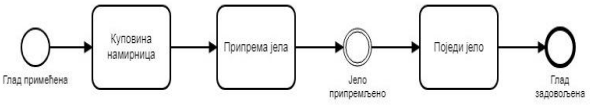
NAPOMENA:
Ovaj rad proistekao je iz master rada čiji je mentor bio dr Miroslav Zarić, vanr. prof.

Prvu fazu čine dizajn i analiza. Druga faza jeste konfigurisanje, u kojoj se odlučuje da li će poslovni proces koristiti informacioni sistem ili ne. Sledeća faza je izvršavanje, u kojoj se pokreću instance procesa da bi se ostvarili željeni rezultati organizacije. I na kraju, peta faza je evaluacija, u kojoj se analiziraju podaci prikupljeni u prethodnim fazama.
Osnovni cilj svakog poslovnog procesa je da stvori vrednost za korisnika. Cilj organizacije ne mora se isključivo izvršavati kroz jedan poslovni proces. Cilj se može izvršiti i kroz više procesa koji su u međusobnoj interakciji sa drugim procesima ili procesima iz drugih organizacija. Oni zajedno treba da na efektivan način doprinose stvaranju vrednosti za korisnika.
Prvu fazu čine dizajn i analiza. Druga faza jeste konfigurisanje, u kojoj se odlučuje da li će poslovni proces koristiti informacioni sistem ili ne. Sledeća faza je izvršavanje, u kojoj se pokreću instance procesa da bi se ostvarili željeni rezultati organizacije. I na kraju, peta faza je evaluacija, u kojoj se analiziraju podaci prikupljeni u prethodnim fazama.
Osnovni cilj svakog poslovnog procesa jeste da stvori vrednost za korisnika. Cilj organizacije ne mora se isključivo izvršavati kroz jedan poslovni proces. Cilj se može izvršiti i kroz više procesa koji su u međusobnoj interakciji sa drugim procesima ili procesima iz drugih organizacija. Oni zajedno treba da na efektivan način doprinose stvaranju vrednosti za korisnika.

3. BUSINESS PROCCES MODELING AND NOTATION(BPMN)

BPMN je standard za modelovanje poslovnih procesa koji definiše grafičke notacije za specificiranje poslovnih procesa.
Nastao je kao proizvod *Business Process Management Initiative* (BPMI). Grafičkim notacijama standardizuju se elementi procesa, čime se postiže nezavisnost od platforme na kojoj će se proces izvršavati.
Elementi koji čine BPMN dijagrame mogu se podeliti u 4 grupe:

1. Objekti toka (*flow objects*) : događaji, aktivnosti, grananja
2. Objekti povezivanja *(connecting objects)*: asocijacije, tokovi poruka (*Messages Flow*), tokovi procesa (*Sequence Flow*)
3. *Swimlanes*: *pools* ili *lane*
4. *Artifacts:* grupe, anotacije, objekti podataka



Slika 1: Proces pripreme jela [2]

Događaji se označavaju krugovima koji sadrže simbole u sebi u zavisnosti od tipa događaja. Događaj predstavlja nešto što se desilo u toku procesa. Aktivnosti koristimo kada je potrebno neku radnju obaviti nad podacima u procesu.
Korisnički zadaci (*User Tasks*) su najčešće korišćeni tip. U ovom tipu aktivnosti ljudi vrše direktan uticaj nad podacima koji se nalaze u poslovnom procesu. Veza između organizacione strukture i poslovnog procesa ostvaruje se dodelom radnih zadataka.
Radni zadaci predstavljaju instance aktivnosti koje su dodeljene određenim zaposlenim na obavljanje [1]. Da bi tok procesa postojao, potrebno je povezati aktivnosti i događaje i utvrditi uslove kojima se određuje tok kretanja. U BPMN-u povezivanje se ostvaruje upotrebom odgovarajućih grananja i spajanja.
U zavisnosti od potreba modela, kretanja nakon donošenja odluke mogu biti paralelne, bira se samo jedna putanja, obe, sve ili neke od.

4. XML PROCESS DEFINITION LANGUAGE (XPDL)

XPDL je opisan u ovom poglavlju kao jedan od najčešće korišćenih jezika za modelovanje i definisanje poslovnih procesa u kompanijama. Dati su osnovne informacije o njegovom nastanku i prikazani su elementi koji se koriste prilikom kreiranja.

4.1 Osnovne informacije i elementi

XPDL [3] je jezik baziran na XML-u, koji služi za opis definicije procesa, kreiranog od Workflow Management Coalition (WfMC) [4]. WfMC je osnovana u maju 1993. godine.
Workflow Reference Model (WRM) objavljen je 1995. godine i još uvek se koristi kao osnova za mnoge poslovne procese i sisteme koji su danas u upotrebi. WRM se sastoji od pet interfejsa koji ovaj model čine generičkim i daju mu mogućnost da odgovori različitim zahtevima u procesu poslovanja.
Osnovni cilj XPDL-a jeste da sačuva i razmenjuje dijagram procesa, tj. omogućava jednom alatu da modeluje proces, drugom da ga čita i modifikuje i na kraju, nekom alatu da izvršava proces.
Uz pomoć XPDL-a mogu se definisati specifični delovi poslovnog procesa. Specifikacija XPDL-a koristi kao mehanizam za razmenu definicija procesa. XPDL formira standard za razmenu koji omogućava proizvodima da podržavaju različite interne prikaze definicija procesa sa funkcijom uvoza/izvoza.
Glavni elementi XPDL koji se mapiraju na elemente BPMN su: *Package* (Paket), *Application* (Aplikacija), *Workflow-Process* (Proces), *Activity* (Aktivnost), *Transition* (Tranzicija), *Participant* (Učesnik), *DataField* (Polje podatka) i *DataType* (Tip podatka).

5. DODELA KORISNIČKIH ZADATAKA

Dodela korisničkih zadataka je jedan od ključnih aspekata za uspešno izvršavanje procesa u organizacionim strukturama kompanija. Prikazano je kako BPMN izvršno okruženje upravlja zadacima i kako korisnički zadaci utiču na njegov rad i izvršavanje procesa.

5.1 Šabloni za dodelu zadataka

Svaki korisnički zadatak u procesu poslovanja ima svoj životni ciklus. Ovaj tip zadatka može prelaziti iz stanja u stanje, sve dok se ne proglasi da je završen. Da bi proces uspešno nastavio da funkcioniše i teži ka cilju zbog kojeg je modelovan, neophodno je proglasiti odgovornog za izvršavanje korisničkog zadataka. Ukoliko se izvršilac ne navede proces će ostati zaglavljen i neće ići dalje od trenutnog stanja. Kada je zadatak kreiran inicijalno mu je dodeljen korisnik ili grupa korisnika, u zavisnosti od konfiguracije procesa. Zadatak može biti dodeljen jednom korisniku, listi korisnika ili listi grupa. Svaki zadatak, da bi uspešno bio obavljen, mora se razrešiti do korisnika. To znači da ga neko mora preuzeti na sebe, ne može se ostati na nivou grupe.
U procesu modelovanja poslovnog sistema, pored definisanja toka procesa i određivanja aktivnosti koje su neophodne da bi se postigao cilj, potrebno je i odrediti koja aktivnost kom delu organizacione strukture pripada. Uočavamo nekoliko šablona koji nam to omogućavaju.
Kada je prilikom modelovanja određeno da se sve instance neke aktivnosti dodele tačno jednoj osobi radi se o direktnoj dodeli. Zadaci se mogu dodeljivati na osnovu uloge korisnika, prethodne istorije. Odluka ko je izvršilac ne mora se dodati u procesu modelovanja, već se može to uraditi i prilikom izvršavanja procesa.
Razdvajanjem nadležnosti se zasniva na tome da više izvršilaca odradi isti posao, uz ograničenje da jedan izvršilac isti posao ne može obaviti više od jednom. Autorizacijom omogućavamo dodelu na osnovu pozicije u organizacionoj strukturi. Šablon obrade predmeta (case handling) koristimo kada je potrebno da izvršilac bude upoznat sa slučajem.

6 . OSNOVE PROGRAMSKOG JEZIKA *SMALLTALK* I PHARO

U ovom poglavlju opisan je programski jezik Smalltalk na kojem se bazira Pharo u kojem je izvršena implementacija dodele zadataka u poslovnom procesu.

6.1 Programski jezik *Smalltalk*

Smalltalk je nastao ranih sedamdesetih godina prošlog veka u Xerox-ovom centru za istraživanje Palo Alto Research Center (PARC). Prva verzija pojavila se u javnosti 1980. godine, ali nije bila dostupna svima već samo izabranim kompanijama, da bi 1983. nakon izlaska druge verzije postala javno dostupna.
Smalltalk je objektno-orijentisan, dinamčki programski jezik [5]. S obzirom da je sve objekat, u Smalltalk-u ne postoje tipovi, pa se tako novi objekat može kreirati kao

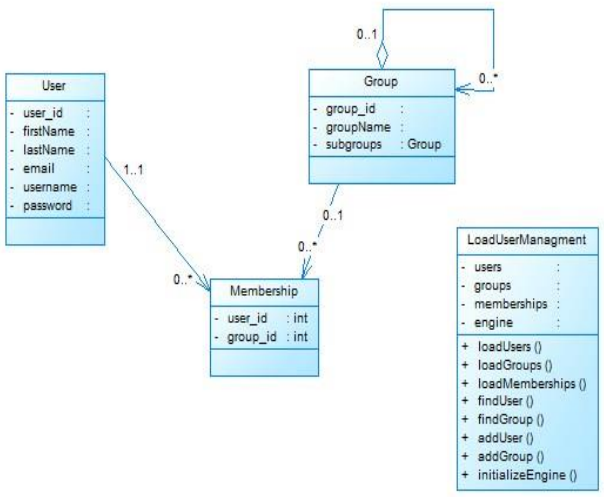
nova instanca klase. Primitivni tipovi kao što su int, boolean ili character su takođe objekti, odnosno instance odgovarajućih klasa i operacije nad njima se vrše slanjem poruka. Klase su objekti koji su instance njihovih meta klasa. Objekti komuniciraju slanjem poruka drugim objektima. Bilo koja poruka može biti poslata bilo kojem objektu.

6.2 . Programski jezik *Pharo*

Pharo su stvorili S.Ducasse i M.Denker u martu 2008. godine [6], fokusirajući se na moderno softversko inženjerstvo i razvojne tehnike i tehnologije. Prva verzija Pharo programskog jezika pojavila se u aprilu 2010. godine i nalazi se pod licencom MIT-a. Uspešno se koristiti na svim operativnim sistemima, MAC, Linux, Android, iOS i Windows. Kao što je rečeno komunikacija između objekata obavlja se preko poruka. Postoje tri tipa poruka: unarne, binarne i poruke ključnih reči (keyword). Unarne se izvršavaju prve, zatim binarne i na kraju keyword. Poruke koje imaju isti prioritet izvršavaju se s' leva na desno, osim ukoliko se ne koriste zagrade, koje menjaju tok izvršavanja.

7 . IMPLEMENTACIJA REŠENJA

Kao što je već rečeno, tema ovog rada je dodeljivanje korisničkih zadataka prilikom modelovanja poslovnog procesa. Za potrebe implementacije korišćen je *Pharo* programski jezika u razvojnom okruženju *Pharo Virtual Machine*.
Glavni zadatak je bio napraviti proširenje (*plug-in*) za *NewWave Process Engine* [7], kojim će se omogućiti učitavanje predefinisanih korisnika i grupa i omogućiti njihovo dodavanje korisničkim zadacima. *NewWave* se razvija na Fakultetu tehničkih nauka u Novom Sadu na Katedri za informatiku. *NewWaveUserManagment* je proširenje koje je kreirano za potrebe ovog rada. Na slici 2 predstavljen je dijagram klasa korišćenih prilikom implementacije.



Slika 2: Dijagram klasa modula za upravljanje korisnicima i grupama

Klase *User* i *Group* predstavljaju entitete koji se dodeljuju određenim zadacima. Klasa *User* je modelovana sa standardnim poljima kojima se opisuju korisnici: *firstName*, *lastName*, *email*, *username*, *password*. Klasa *Group* sadrži naziv i može sadržati podgrupe koje su takođe instance klase *Group*. Prilikom dodeljivanja korisnika ili grupe konkretnom zadatku se prosleđuje identifikacioni broj korisnika/grupe. Veza između ova dva entiteta ostvarena je uz pomoć klase *Membership,* radi lakše kontrole i upravljanja nad učesnicima u procesu.
Svaki korisnik može, a i ne mora imati grupu. Takođe, može imati više od jedne. U grupi se može naći više od jednog korisnika. Klasa u čijoj je nadležnosti rad sa ovim entitetima je *LoadUserManagment.*
LoadUserManagment je klasa u kojoj se nalaze svi trenutni korisnici, grupe i veze između njih. Takođe, ima referencu prema *engine*-u. Metode koje omogućavaju učitavanje korisnika, grupa i članstava su respektivno *loadUsers, loadGroups* i *loadMemberships,* sa parametrom koji predstavlja putanju do fajla. Podaci u fajlovima se nalazi u JSON formatu. Da bi se tako struktuirani podaci mapirali na klase u *Pharo*-u korišćena je *NeoJSON* biblioteka [8].

```
neoJsonMapping: mapper

        mapper for: #ArrayOfGroups

                        customDo: [ :mapping |
mapping listOfElementSchema: self ].



        mapper for: self do: [ :mapping |

                mapping mapInstVars: #(groupId
groupName users).

                (mapping mapInstVar: #subgroups)
valueSchema: #ArrayOfGroups ].
```

Listing 1: Mapiranje entiteta Group na JSON za učitavanje

Ovako učitani podaci nalaze se u kolekcijama objekata u klasi *LoadUserManagment* i to redom, *users*, *groups* i *memberships*. Ova klasa omogućava i dodavanje entiteta prilikom definisanja samih zadataka prilikom modelovanja. Dodavanje u kolekcije moguće je pomoću metoda *addUser*, *addGropu*, *addMembership* gde su parametri objekti klasa koji se dodaju.

8 . ZAKLJUČAK

U radu su prikazani koncepti poslovnih procesa, objašnjene su faze životnog ciklusa procesa i prikazan je proces njegovog modelovanja, kao i načini komunikacije unutar samog procesa, ali i eksterne veze između više procesa. Predstavljen je BPMN koji se koristi za modelovanje poslovnog procesa. Pored BPMN-a, u ovoj delatnosti, najviše se koristi XPDL, zbog svog formata koji je nezavisan od platforme. Dat je pregled osnovnih informacija o XPDL i njegovih elemenata. Obrađeni su šabloni na osnovu kojih se dodeljuju izvršioci zadataka.
Kao glavni zadatak ovog rada bilo je kreiranje proširenja *NewWave* softvera, kojim će se omogućiti podrška za korisnike i grupe korisnika koji učestvuju u procesu. Implementacija rešenja izvršena je u programskom jeziku *Pharo*, koji je nastao iz *Smalltalk*-a. Trenutno u *NewWave*-u moguća je samo direktna dodela korisnika ili njihovih korisničkom zadatku, pa bi ovo rešenje trebalo

proširiti sa još nekim šablonom za dodeljivanje izvršioca zadatka.

9. LITERATURA

[1] Miroslav Zarić, predavanja iz predmeta „Upravljanje poslovnim procesima“, Katedra za informatiku, FTN, Novi Sad, 2019. godine

[2] https://camunda.com (pristupljeno u avgustu 2020.)

[3] http://xpdl.org/ (pristupljeno u septembru 2020.)

[4] “Workflow Management Coalition Workflow Standard: Workflow Process Definition Interface – XML Process Definition Language (XPDL) (WFMCTC-1025). “ Technical report, Workflow Management Coalition, Lighthouse Point, Florida, USA, 2002.

[5] Adele Goldber , „Smalltalk-80: The Interactive Programming Environment“

[6] Black, A., Ducasse, S., Nierstrasz, O., Pollet, D., Cassou, D., Denker, M., „Pharo by example. 2009.“

[7] https://github.com/skaplar/NewWave

[8] https://ci.inria.fr/pharo-contribution/job/EnterprisePharoBook/lastSuccessfulBuild/artifact/book-result/NeoJSON/NeoJSON.html (pristupljeno u septembru 2020.)

Kratka biografija:

Nemanja Gavrilović rođen je 14. avgusta 1995. godine u Šapcu, Srbija. Srednju školu završio je 2014. godine u Malom Zvorniku, smer ekonomski tehničar. Iste godine upisuje Fakultet tehničkih nauka u Novom Sadu, odsek Računarstvo i automatika. Na trećoj godini osnovnih akademskih studija opredelio se za usmerenje Primenjene računarske nauka i informatika. Nakon osnovnih studija upisuje master studije na istom odseku i smeru, usmerenje Elektronsko poslovanje.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.3.049.77
DOI: https://doi.org/10.24867/12BE46Mrkaic

REALIZACIJA APLIKACIJE ELEKTROKARDIOGRAFA ZA DETEKCIJU SRČANIH ANOMALIJA

REALISATION OF THE ELECTROCARDIOGRAPHIC APPLICATION FOR DETECTION OF HEART ANOMALIES

Nebojša Mrkaić, *Fakultet tehničkih nauka, Novi Sad*

Oblast– ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad opisuje kako će se kreirati aplikacija koja će uspešno detektovati srčane probleme osobe koja ga koristi. Definisana su stanja poput ubrzanog rada srca, usporenog rada srca, normalnog rada srca i prestanka rada srca. Takođe je ubačena i statistika broja otkucaja srca, prosečan broj otkucaja u minuti kao i grafičko prikazivanje srčanog ritma.*

Ključne reči: *EKG, infarkt, obrada signala, stres, srce, zdravlje.*

Abstract – *This paper describes how the application is going to be created that detects heart failures and anomalies. Those failures are defined as increased heart rate, decreased heart rate, normal heart rate and no heart rate. Heart statistics is also implemented with graphic display of heart rhytm.*

Keywords: *ECG, heart attack, signal processing, stress, heart, health.*

1. UVOD

U današnjem svetu ljudi ne obraćaju pažnju na svoje zdravlje pošto je svet dosta promenjem i ubrzava se život iz dana u dan. Zbog toga okreću se nezdravom životu ne zato što oni hoće već zato što su primorani. Takođe, broj penzionera se iz dana u dan povećava pošto se kvalitet života poboljšava i životni vek se produžava.

U njihovim situacijama, postoje periodi života kada su usamljeni i nemaju nikog ko da vodi računa o njima. Svi ti primeri mogu da pokažu da je potrebno svetu da se osloni na naprednu tehnologiju ne samo po pitanju zanimacije i poslova već i povodom zdravlja. A u zdravlju, najčešće nam strada organ koji nije u mogućnosti da se regeneriše a bez njega ne možemo – srce.

Rad srca je izuzetno bitan da je na maksimumu jer na osnovu njega naše telo dobija hranljive materije koje su mu non stop potrebne.
Takođe, veoma veliki broj bolesti se pojavljuje iz situacije kada srce ne funkcioniše kako treba. Zbog svega toga, ova aplikacija bi nam omogućila da deo zdravlja kontrolišemo iako imamo srčanih oboljenja ili smo u potpunosti zdravi.

Da bismo započeli kreiranje aplikacije, potrebno nam je znanje iz više oblasti pošto je ova aplikacija hibridnog karaktera. Dok moramo poznavati medicinu i kako naše telo funkcioniše, moramo isto tako biti upućeni u hardverske komponente elektrokardiograma i računara. Takođe, moramo znati i softvere elektrokardiograma i računara, odnosno signal sa elektrokardiograma i pisanje koda na računaru. Kada se skupi određeno znanje iz tih oblasti, tada se može pristupati sa biomedicinarske strane koja obuhvata sve navedene oblasti od elektrotehnike, preko medicine do programiranja.
Najkompleksniji deo aplikacije jeste kako uspeti na osnovu jednostavnog elektrokardiografskog signala izvršiti klasifikaciju u svoju odgovarajuću grupu. Svaki srčani problem predstavlja poseban problem koji se mora tretirati posebno na svoj način. Nikako se ne sme doći u opciju da dođe do pogrešnog klasifikovanja srčanog signala jer time dolazi do pogrešne upotrebe lekova koji mogu da ne doprinesu nikakav boljitak, a čak može da dođe i do pogoršavanja stanja.
Zbog toga se mora veoma precizno sa dovoljno predznanja konstruisati ova aplikacija da ne bi imala mogućnost otežanja osobi koja već ima problema sa srcem.

2. ELEKTROKARDIOGRAF

Elektrokardiograf je uređaj koji registruje i beleži električnu aktivnost na površini kože koja nastaje kao posledica rada srca (Slika 1.). EKG (Elektro Kardio Gram) je grafički prikaz električne aktivnosti koju ispisuje elektrokardiograf. On meri promenu potencijala na elektrodama postavljenim na površini kože u blizini srca. EKG je važno dijagnostičko sredstvo u lečenju mana i oboljenja srca, kao i naravno za potvrdu normalnog stanja – zdravlja srca.

Slika 1. *Elektrokardiograf*

U konvencionalnom 12-kanalnom EKG-u, postavlja se 10 elektroda - na ekstremitetima pacijenta i na površini grudnog koša.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio Platon Sovilj.

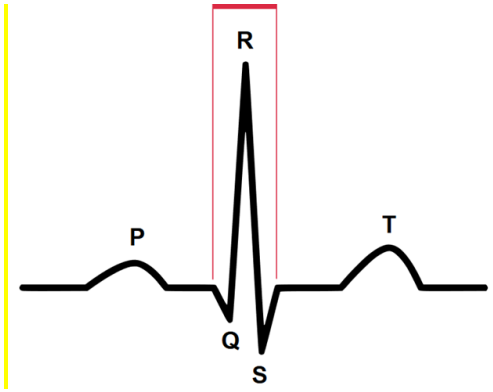
Ukupne veličine električnog potencijala srca se zatim mere iz 12 različitih uglova („vodova"), a snima se u određenom vremenskom periodu (obično 10 sekundi).

3. EKG SIGNAL

Prosečno srce čoveka otkuca otprilike oko sto hiljada puta tokom dana. Mnoge srčane tegobe počinju da se dešavaju tako što poremete par otkucaja srca tokom dana, što ne bi predstavljao veliki zdravstveni problem, ali u budućnosti može da ima fatalne posledice.

Tih par otkucaja (slika 2.), mogu da eksaliraju, i još ako se dodaju drugi faktori poput prekomernog umora i konzumacija određenih supstanci (alkoholnog pića, slatkih sokova..), rezultat svega toga može da dovede do prestanka rada srca. Pogotovo, ako gledamo na specifične slučaje i osobe koje konzumiraju narkotike i druga sredstva koja nanose štetu organizmu.

Zbog toga, da bi izbegli takve incidente, potrebno je što pre na što sigurniji način otkriti da li se zdravstveno stanje osobe pogoršava, da bi se mogle preduzeti određene mere da mu se sačuva život. Osobe koje su u komi, ili im je život ugrožen, su prikačeni na EKG.



Slika 2. *Elektrokardiografski snimak jednog otkucaja*

Na signalu mogu da se uoči razlika u određenim delovima signala, i svaki taj segment predstavlja određen deo rada srca.

P talas predstavlja atrialnu depolarizaciju. Tada krv ulazi u srce, odnosno u pretkomore u PR intervalu.

QRS se odnosi na ventrikularnu depolarizaciju, odnosno kada srce ispumpava krv i šalje je telu. Q talas reaguje na depolarizaciju, R talas je najveći jer se tada dešava kontrakcija. S deo obeležava finalni deo depolarizacije. QRS kompleks je najbitniji.

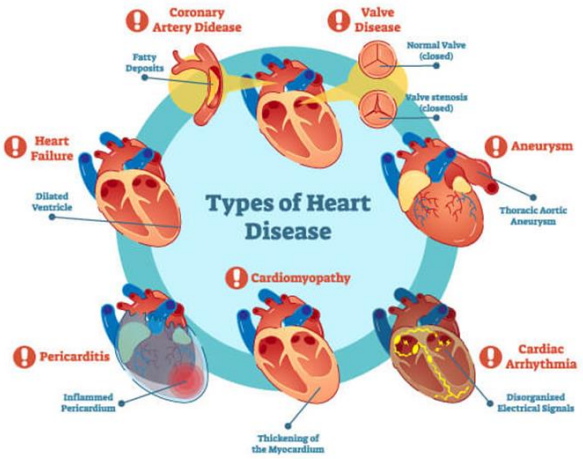
ST interval predstavlja nulti period između polarizacije i depolarizacije ventrikula.

T talas se odnosi na ventrikularnu depolarizaciju.

4. SRČANA OBOLJENJA

Pre svega, da bi se vršila korektna klasifikacija, potrebno je saznati o kakvim se oboljenjima radi i šta sve možemo da očekujemo povodom njih (slika 3.). Prvenstveno, srčana oboljenja koja su navedena na slici ispod se razlikuju po lokaciji na kom oboljenje nastaje, pored čega dolazi i do ispitivanja nivoa ozbiljnosti navedenih oboljenja.

Miokardialna infarkcija je poznat kao srčani napad kada oštećen deo srca napravi prekid protoka krvi na osnovu zagušenja. To zagušenje može da bude uzrok zgrušavanja krvi, što opet može da prourokuje mnogo faktora, od alkoholizma, preko nervoze do prekomerne težine osobe.



Slika 3. *Srčana oboljenja*

Bolest srčanih valvi, kojih ima četiri i locirane su u svakoj srčanoj komori i odgovorne su za odvođenje i dovođenje krvi unutar srca, odnosno predstavljaju kapiju između komora i pretkomora srca.

Valvularna stenoza se može klasifikovati u bolest srčanih valvi, gde ne dolazi do kompletnog širenja valvi i samim time dolazi do slabijeg protoka i otkazivanja srca.

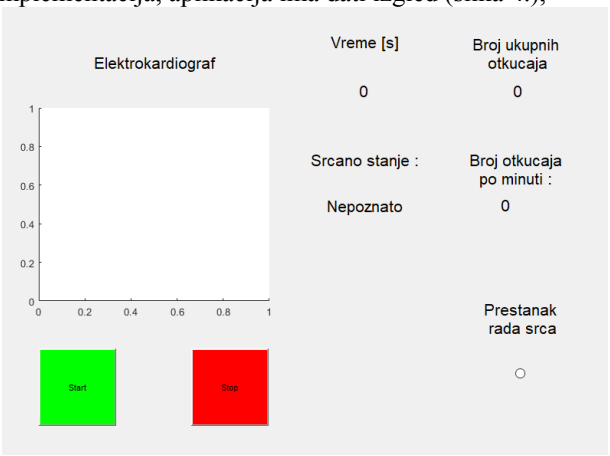
Kardiomiopatija predstavlja fizička povreda srca, gde dolazi da srčano mišićno tkivo slabi, a pošto je to neregenerativni organ poput jetre, počinje da izvršava svoju funkciju nepotpuno (sl. 3). Tada u levoj ventrikuli dolazi do podebljanja srčanog zida što otežava protok krvi a samim time smanjuje aktivnost date osobe i postavlja je u nezgodan položaj. Na slici se može videti zadebljanje srčanog zida, a unutra se može primetiti da je došlo da velike štete srčanog tkiva, koje nema pravilan oblik srca, već je povređen deo tkiva razgranat poput drveta.

Perikardialna bolest napada srčanu maramicu koja okružuje srce uglavnom zbog određenih infekcija i virusa povodom zapaljenja disajnih puteva. Kada osoba dobije upalu pluća (rheumatoid arthritis), ili mu se desi nešto slično, ukoliko ne bude vodio računa, može da se zapaljenje proširi i na druge organe, kao u ovom primeru, na srce.

5. IZRADA APLIKACIJE

Za početak smo pristupili programskom paketu Matlab gde smo prvenstveno kreirali GUI (Graphic User Interface) i započeli smo sa implementacijom izgleda i funkcionalnost naše aplikacije. Za početak imali bi dugme start koje pokreće aplikaciju i dugme stop koje zaustavlja aplikaciju. Dok je aplikacija aktivirana, jedino što možemo da uradimo jeste da zaustavimo aplikaciju ili da izađemo iz aplikacije. Ostale opcije smo onemogućili za korišćenje. Tada smo dodali parametre povodom broja ukupnih otkucaja i vremena. Tada, uz pomoć konstantne evidencije vremena i otkucaja dobili bi otkucaje po minuti na osnovu kojih bi vršili separaciju otkucaja po grupama. Da bi pacijent imao bolji utisak da aplikacija korektno radi i da on može vizuelno da isprati rad njegovog srca,

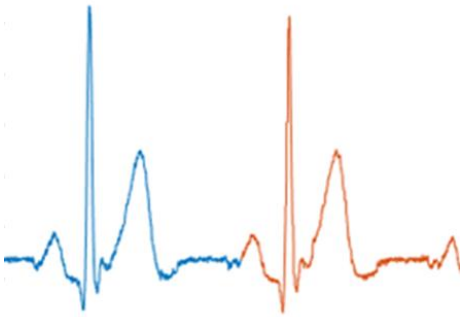
ubačen je grafik, gde mu x osa predstavlja dužinu signala dok mu y osa predstavlja jačinu signala. Uz sve te parametre, ubačeno je i srčano stanje koje je trenutno prisutno.Takođe, postojao bi i detektor za srčane napade, koji bi sa trigerovanjem dugmeta i audia obavestio korisnika i osobe u okolini korisnika. Nakon svih tih implementacija, aplikacija ima dati izgled (slika 4.),



Slika 4. *Aplikacija za detektovanje srčanih anomalija*

6. PROŠIRIVANJE APLIKACIJE

Nakon što smo uspešno ispodešavali i implementirali vizuelan deo aplikacije, potrebno je da se i softverski implementira, odnosno da se definiše šta će tačno da se dešava. Za početak signal koji se dovodi jeste signal sa elektrokardiograma i potrebno je napisati kod koji uspešno obrađuje dati signal. Za početak signal koji dolazi je podeljen u dva paketa, gde bi se prvi paket prikazao na prvoj polovini grafika, a drugi paket bi se prikazao na drugom delu grafika. Kada bi došao nov deo signala, drugi paket prelazi na mesto prvog paketa, a nov paket dolazi na mesto drugog paketa i time dobijamo kontinualan prikaz signala sa elektrokardiograma. Plavi signal predstavlja paket jedan a paket dva predstavlja narandžasti signal (Slika 5.). Na aplikaciji će biti kontinualan prikaz otkucaja srca.



Slika 5. *Prikazivanje kontinualnosti signala preko paketa*

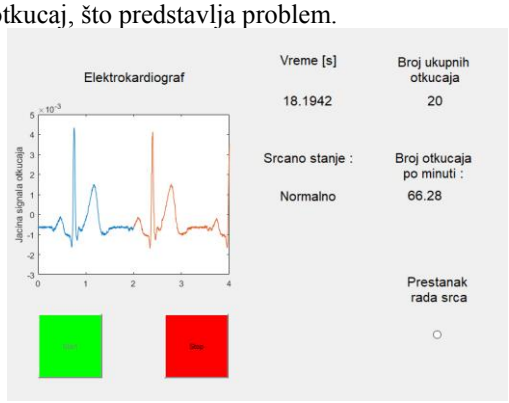
Sledeći korak u programiranju jeste detektovati tačno jedan otkucaj srca, i ne detektovati artefakte i šumove. Na osnovu eksperimentalne faze, na određenoj jačini bi se završavao QRS kompleks koji je najvidljiviji u srčanom ritmu.
Iz tog jednog ritma bi se uzela samo jedna vrednost i na osnovu nje bi se izvršila evidencija u brojaču da se desio jedan otkucaj. Ukoliko je jačina slabija, to ne može da se konstantuje kao otkucaj pošto nema dovoljnu jačinu.

7. DEFINISANJE OPSEGA

Od izuzetnog je značaja postaviti pragove u kodu. Za početak, svi signali blizu nule i iznad vrednosti otkucaja srca su odsečeni i podešeni na neutralnu vrednost 0. Onda smo postavili granice u separaciji ritma srca. Ritam srca koji je normalan se nalazi u opsegu između 60 i 100 otkucaja po minuti. Ukoliko je ritam ubrzan i prelazi preko 100, tada ulazimo u proces tahikardije. Ukoliko srce usporeno radi i padne ispod 60 otkucaja po minuti, tada ulazi u stanje bradikardije.

8. TESTIRANJE APLIKACIJE

Kada je aplikacija startovana pritiskom na dugme start, ona počinje sa izvršavanjem. Vrednost na grafiku dostiže do 0.005, na osnovu koje vrednosti smo uspeli targetirali vrh QRS kompleksa. Ukoliko QRS kompleks ne bude zadovoljavao određenu visinu, ono neće biti registrovano kao otkucaj, što predstavlja problem.



Slika 6. *Jedan od testiranja aplikacije*

Vreme je postavljeno da ispisuje i poslednje 4 decimale zbog tačnosti, a broj otkucaja po minuti sadrži poslednje dve decimale (slika 6). Srčano stanje je normalno i broj ukupnih otkucaja je vidljiv, ali je zato elektrokardiografski snimak prikazan iz dve boje da bi se prikazivao kontinualno.

Odnosno, prvo je obrađen prvi paket signala koji sadrži 1000 vrednosti, i postavljan je u oblast na x osi između vrednosti 2 i 4. Nakon toga, obrađuje se sledećih 1000 vrednosti. Tada, prvi paket signala se postavlja između vrednosti 0 i 2, a drugi paket se postavlja između vrednosti 2 i 4. Odnosno, naredni paket će uvek biti prikazan desno, a prethodni paket levo, na osnovu čega dobijamo konstantno kretanje signala. Takođe, dužina paketa obrade se može promeniti što će uticati na brzinu otkucaja srca i menjanje signala. Ukoliko ima više vrednosti u jednom paketu, prikaz signala će biti sitniji.

9. ZAKLJUČAK

Nakon konstruisanja aplikacije i pisanje njenog koda, gde se na kraju taj kod manuelno testirao i uspešno je prošao sve korake, zaključak je da je aplikacija spremna za dalju esploataciju. Za aplikaciju bi se našla odgovarajuća hardverska postavka koja bi ispunjavala sve zahteve aplikacije sa svim senzorima, memorijama i displejima.

Aplikacija će takođe morati imati i određeno održavanje u slučaju ako se na njoj dese neželjene stvari (kvar, neregularan rad aplikacije..) i takođe da bi se usavršila aplikacija, moguće je i dopisivanje novog koda koji bi bio

nadogradnja. Uz to bi bilo planirano da i uređaj sadrži odgovarajuć lek za pacijenta, da u slučaju mu se desi nešto neočekivano, i hitna pomoć ne bi stigla brzo, da taj lek uspe da pomogne osobi, i čak da mu spasi život.
Osim toga, u planovima budućeg razvoja aplikacije, planira se njeno integrisanje sa istraživačkim sistemom za razvoj stohastičkog elektorkardiografa – elektrokardiografa koji će primeniti koncepte iz ranije razvijenih metoda digitalnog stohastičkog merenja [7-15].

10. REFERENCE

[1] „Heart Attack or Sudden Cardiac Arrest: How Are They Different?". www.heart.org. 30. Jul 2014.
[2] "Your Heart Failure Healthcare Team". www.heart.org.
[3] Marian, A. J.; Roberts, Robert (1. 4. 2001). „The Molecular Genetic Basis for Hypertrophic Cardiomyopathy". Journal of Molecular and Cellular Cardiology.
[4] Mark, Jonathan B. (1998). Atlas of cardiovascular monitoring. New York: Churchill Livingstone.
[5] Moyer, V. A. (2. 10. 2012). „Screening for coronary heart disease with electrocardiography: U.S. Preventive Services Task Force recommendation statement.". Annals of Internal Medicine.
[6] Braunwald E. (ed) (1997), Heart Disease: A Textbook of Cardiovascular Medicine, Fifth Edition, p. 108, Philadelphia. W B Saunders Co.
[7] Vladimir Vujicic, Boris Licina, Dragan Pejic, Platon Sovilj, Aleksandar Radonjic: Stochastic measurement of wind power using a two-bit A/D converter, Measurement Volume 152, February 2020, 107184, ISSN 0263-2241, https://doi.org/10.1016/j.measurement.2019.107184
[8] Pejić D., Naumović-Vuković D., Vujičić B., Radonjić A., Sovilj P., Vujičić V.: Stochastic digital DFT processor and its application to measurement of reactive power and energy, Measurement, 2018, pp. 494-504, ISSN 0263-2241, https://doi.org/10.1016/j.measurement.2018.04.004
[9] 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, https://doi.org/10.1016/j.bspc.2017.07.006
[10] 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
[11] 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
[12] Pejić D., Gazivoda N., Ličina B., Urekar M., Sovilj P., Vujičić B.: A Proposal of a Novel Method for Generating Discrete Analog Uniform Noise, Advances in Electrical and Computer Engineering, 2018, Vol. 18, No 3, pp. 61-66, ISSN 1582–7445, DOI: 10.4316/AECE.2018.03009
[13] 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
[14] 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
[15] 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

Kratka biografija:

Nebojša Mrkaić je rođen 1995. godine u Novom Sadu. Završio je srednju školu Jovan Jovanović Zmaj 2014. godine u Novom Sadu, nakon čega je upisao Fakultet tehničkih nauka u Novom Sadu i diplomirao je 2019. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.39
DOI: https://doi.org/10.24867/12BE47Zelenovic

SOFTVERSKA MIGRACIJA UPOTREBOM AKANA PLATFORME

SOFTWARE MIGRATION USING THE AKANA PLATFORM

Danijela Zelenović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad bavi se predstavljanjem Akana platforme za upravljanje **API**-jima i pokazivanjem njene efikasnosti u oblasti telekomunikacija prilikom softverske migracije podataka i funkcionalnosti sa **V6 box** televizijskog softvera na **Horizon 4**.*

Ključne reči: *Akana, softverska migracija, bezbjednost, API, Horizon 4, Upravljanje API-jima*

Abstract – *This paper deals with the presentation of the Akana platform for **API** management and demonstrating its efficiency in the field of telecommunications during software data migration and functionalituec from **V6 box** television software to **Horizon 4**.*

Keywords: *Akana, software migration, security, API, Horizon 4, API Management*

1. UVOD

Application Programming Interfaces (APIs) se nalaze u samom srcu digitalne revolucije. U ovom radu biće data definicija ***API***-ja, kao i kratak pregled njihovih osobina, namjena i primjena u poslovnom svijetu. Predstavljanjem svih koraka u životnom ciklusu ***API***-ja, dobiće se jasan uvid koje sve korake treba ispuniti kako bi se kreirao stabilan i funkcionalan ***API***, koji je spreman da efikasno odgovori na različite poslovne izazove.

Sumirani pregled platformi za upravljanje ***API***-jima obezbijediće jasan uvid koja od njih je u mogućnosti da obezbijedi odgovarajući nivo sigurnosti i kontrole pristupa, zaštitu od zloupotrebe podataka, stabilnost prilikom skaliranja i integracije sa drugim sistemima, ugrađen sistem sa upravljanje životnim ciklusom ***API***-ja, alat za testiranje, detaljnu analitiku, praćenje, odgovarajuću dokumentaciju za brže i jednostavnije rukovanje prilikom obezbjeđivanja navedenih funkcionalnosti.

Akana, platforma koja je ispunila prethodno navedene ciljeve, široko je rasprostranjena u različitim oblastima primjene čiji je kratak pregled izvršen u radu.

Cilj ovog rada jeste da predstavi Akana platformu i pokaže njenu efikasnost u oblasti telekomunikacija prilikom softverske migracije podataka i funkcionalnosti sa ***V6 box*** televizijskog softvera na ***Horizon 4***.

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

2. PROGRAMSKI INTERFEJS APLIKACIJE

Programski interfejs aplikacije ***(Application Programming Interface - API)*** predstavlja skup funkcija ili procedura za pristup servisima operativnog sistema, softverskim bibliotekama ili drugim sistemima, koje mogu koristiti drugi programi [1]. ***API*** je posrednik koji omogućava komunikaciju između dva sistema, a među kojima je neophodno izvršiti razmjenu podataka ili funkcionalnosti. ***API*** se može posmatrati i kao posrednik koji kontroliše i ograničava pristup nekom softverskom sistemu, ali kontroliše upotrebu podataka koji se razmjenjuju u komunikaciji. ***API***-ji se mogu svrstati u grupu mrežnih servisa ukoliko se komunikacija između dva sistema obavlja putem mreže. Mrežni servis predstavlja kolekciju protokola i standarda otvorenog koda koja se koristi za razmjenu podataka između sistema ili aplikacija, dok je ***API*** softverski interfejs koji omogućava da dva sistema međusobno komuniciraju bez učestvovanja korisnika [2].

3. ŽIVOTNI CIKLUS *API*-JA

Životni ciklus ***API***-ja odvija se u četiri velike faze, koje sadrže određen broj podfaza. Faze i podfaze životnog ciklusa ***API***-ja su:

1. **Kreiranje specifikacije**
   a. Strategija
   b. Dizajn
   c. Simulacija
   d. Povratna informacija
   e. Validacija

   Rezultat je specifikacija ***API***-ja.

2. **Razvoj i priprema za raspoređivanje**
   a. Razvoj
   b. Testiranje

   Rezulatat je ***API*** koji je spreman za uvođenje u upotrebu, odnosno, raspoređivanje i korišćenje kao mrežni servis.

3. **Upravljanje *API*-jima nakon raspoređivanja**
   a. Upravljanje verzijama
   b. Nadgledanje
   c. Analitika
   d. Rješavanje problema
   e. Skaliranje

4. **Povlačenje** - Ukidanje zastarjele verzije.

Podjela životnog ciklusa ***API***-ja na određeni broj faza i podfaza izvršeno je na osnovu literature [3-6].

4. PLATFORME ZA UPRAVLJANJE *API*-JIMA

Izbor platforme/alata za upravljanje ***API***-jima predstavlja veliki izazov za kompanije koje ulažu u neki posao, razvijaju ga ili unapređuju. Povećan nivo sigurnosti i kontrole pristupa, zaštita od zloupotrebe, stabilnost prilikom skaliranja i integracije sa drugim sistemima, ugrađen sistem sa upravljanje životnim ciklusom ***API***-ja, testiranje, detaljnu analitiku, praćenje, odgovarajuća dokumentacija, predstavljaju jedne od osnovnih karakteristika koje platforma za upravljanje ***API***-jima treba da posjeduje. Pored niza navedenih karakteristika koje platforma treba da ispuni da bi zadovoljila poslovne ciljeve, neophodno je da bude pristupačna razvojnom timu za korišćenje, da je obezbjeđena odgovarajuća korisnička podrška od strane dobavljača, ali i da odnos cijene i kvaliteta bude zadovoljavajući kako bi ispunio poslovne ciljeve i zadovoljio veliki broj korisnika. Sumirani pregled nekoliko platformi za upravljanje ***API***-jima dat je u Tabeli 1, koja je formirana na osnovu podataka iz literature [7].

Tabela 1. *Uporedni prikaz platformi za upravljanje **API**-jima*

	Apigee Edge	Kong Enterprise	Postman	3Scale	Akana
Дизајн API-ја	Да	Да	Да	Да	Да
Контрола приступа	Да	Да	Не	Да	Да
Систем за праћење активности	Да	Да	Да	Да	Да
Управљање тестирањем	Не	Не	Да	Не	Да
Контрола саобраћаја	Да	Да	Не	Да	Да
Управљање животним циклусом	Да	Да	Да	Да	Да
Аналитика	Да	Да	Не	Да	Да
Портал за програмере	Да	Да	Не	Да	Да
Заштита од пријетњи	Да	Да	Не	Да	Да
Контрола верзија	Да	Не	Да	Да	Да

5. AKANA

Akana platforma nadoknadila je sve nedostatke prethodno navedenih platformi i alata za upravljanje ***API***-jima. Pogodna je za rad sa malim, srednjim, ali i velikim projektima.

Akana omogućava jednostavno dizajniranje jednostavnih, dobro struktuiranih ***API***-ja grafičkim alatom koji podržava više programskih jezika.

Podržava različite sigurnosne standardne za autorizaciju, autentifikaciju, prevenciju napada i protiv zloupotrebe podataka kako bi zaštitila svoje ***API***-je i očuvala privatnost podataka koje se razmjenjuju u komunikaciji. Upotrebom ***OAuth*** protokola, ***JWT (JSON Web Tokens)*** mehanizma zbog različitih vrsta tokena koji se šalju, ***HTTPS*** protokola, sertifikata i bezbjednosnih polisa kao sigurnosnih mehanizama, prethodno navedene zaštite ***API***-ja su moguće.

Posjedovanje sistema za praćenje aktivnosti u velikoj mjeri olakšava posao rješavanja određenih problema koji se javljaju prilikom podrške klijentima.

Prije puštanja ***API***-ja u upotrebu od strane korisnika, potrebno je izvršiti testiranje kako bi se proverilo da ***API*** funkcioniše kako je očekivano. Platforma posjeduje ***Test Client*** alat koji omogućava jednostavno i detaljno testiranje svih mogućnosti postojećih ***API***-ja u okviru platforme.

Kreiranje nove verzije ***API***-ja moguće je kloniranjem postojeće verzije ili uvoženjem nekog od ***Swagger/RAML/WSDL/WADL*** dokumenata od strane biznis administratora ili administratora ***API***-ja.

Zaštita podataka od zloupotrebe i kontrola saobraćaja je moguća u Akana platformi korišćenjem polisa usklađenosti, operativnih i ***QoS (Quality of Service)*** polisa, definisanjem pravila za dodavanje ***IP (Internet Protocol)*** adresa u bijelu i crnu listu, upotrebom mehanizama za rutiranje, dodavanjem polisa koje dozvoljavaju pristup ***API***-ju sa određenog domena adresa, upotrebom bezbjednosnih polisa koje su zadužene za prevenciju napada kojim se ograničavanja format poruke kako bi se spriječili ***Cross-Site Scripting, SQL (Structured Query Language) Injection*** napadi, itd.

Akana platforma upravlja ***API***-jima od njihovog nastanka, od dizajn faze, pa sve do faze nakon koje je moguće njihovo korišćenje u realnom svijetu. Nakon raspoređivanja ***API***-ja, Akana platforma vrši praćenje, analizu, optimizaciju i poboljšavanje performansi svojih ***API***-ja i na taj način je uključena u svaku fazu životnog ciklusa.

Analitika je od izuzetne važnosti za praćenje performansi ***API***-ja. Akana platforma ima moćan mehanizam za analitiku koji daje uvid u to koliko je prosječno vrijeme kašnjenja zajedno sa informacijama o vrijednosti minimalnog i maksimalnog kašnjenja, koliko je zahtjeva primljeno u toku jednog dana, kao i broj palih zahtjeva, a sve te informacije su od velikog značaja za praćenje i poboljšavanje performansi ***API***-ja.

Akana platforma sadrži veliku zajednicu programera, koji su u međusobnoj komunikaciji i doprinose razvoju i širenju popularnosti platforme. Portal za programere je napravljen kako bi se omogućila ta komunikacija, ali i prodaja i objavljivanje ***API***-ja.

Akana je odgovorna za očuvanje integriteta, dostupnosti i povjerljivosti kao tri osnovne stavke koje treba obezbijediti da bi se osigurala bezbjednost podataka. Kao što je već napomenuto, Akana sistem posjeduje mehanizme za kontrolu pristupa ***API***-jima, a i mehanizme za kontrolu saobraćaja koji se razmjenjuje između potrošača i krajnjih sistema preko Akana sistema. Mehanizmi za kontrolu pristupa štite od neovlašćenog pristupa ***API***-jima i zloupotrebe podataka i funkcionalnosti od strane zlonamjernih korisnika. Pored ***QoS*** polisa postoje i bezbjednosne polise koje vrše ograničavanje pristupa preko ***HTTP*** protokola koji nema ugrađene funkcionalnosti namenjene bezbednosti, za razliku od ***HTTPS*** protokola.

Zaštita od neovlašćenog pristupa ***API***-jima vrši se i upotrebnom različitih vrsta autentifikacije. Ukoliko je ***API*** zaštićenim određenom autentifikacijom, potrošači neće biti u mogućnosti da se povežu sa krajnjim sistemom ukoliko prethodno ne prođu Akana kapiju sa odgovarajućim tokenima za uspješnu autentifikaciju. Najčešće korišćene vrste autentifikacija potrošača ka Akani su:

1. ***OAuth*** autentifikacija - Predstavlja tip autentifikacije koji omogućava potrošačima da se autentifikuju ka Akana ***API***-ju generisanjem ***OAuth*** tokena pozivajući ***OAuth***

server i slanjem dobijenog tokena u zaglavlju zahtjeva prilikom pozivanja ***API***-ja.

2. Osnovna (***Basic***) autentifikacija - Osnovna ***HTTP*** šema autentifikacije omogućava autentifikaciju kodiranjem identifikacione oznake aplikacije i šifre koja je specifična za tu aplikaciju u sljedećem obliku: ***client_id:client_secret*** [8].

3. ***Atmosphere*** autentifikacija - Za ***Atmosphere*** tip autentifikacije neophodno je samo znati identifikacionu oznaku aplikacije.

4. Autentifikacija preko treće strane - Korišćenjem odgovarajućih kredencijala, zahtjeva se pristup drugim servisima koji su u mogućnosti da dobave zahtjevani token, tako da su potrošači u mogućnosti da se autentifikuju korišćenjem ***CIAM (Customer Identity and Access Management), UXP (User Experience Platform), Ping Federate, Google ReCaptcha, ADFS (Active Directory Federation Services)*** tokena.

6. OBLASTI PRIMJENE AKANA PLATFORME

Razvojem ekonomije ***API***-ja, javila se i potreba za stvaranjem stabilnih, sigurnih, bezbjednih, ali i robusnih sistema, kao i sistema sa dobrim performansama izvršavanja. Uspješno kreiran sistem koji upravlja ***API***-jima treba da posjeduje svaku od navedenih karakteristika. ***API***-ji predstavljaju most između dva različita sistema i odgovorni su da komunikacija među njima bude bezbjedna, uspješna i stabilna, tako da kreiranje sistema koji posjeduje sve navedene karakteristike predstavlja veliki izazov u svijetu ***API***-ja. Pored toga, neophodno je omogućiti da je takav sistem i fleksibilan, lak za održavanje, upravljanje, izmjenu i prilagođavanje novim poslovnim izazovima i oblastima primjene. Kako bi sistem zadovoljio potrebe većeg broja krajnjih korisnika, optimizivao svoje procese rada, neophodno je obezbijediti i osobinu skalabilnosti, ali da i dalje zadrži mogućnost očuvanja stabilnosti prilikom integracije sa novim i postojećim sistemima. Akana sistem pruža gore navedene osobine u sljedećim oblastima primjene:

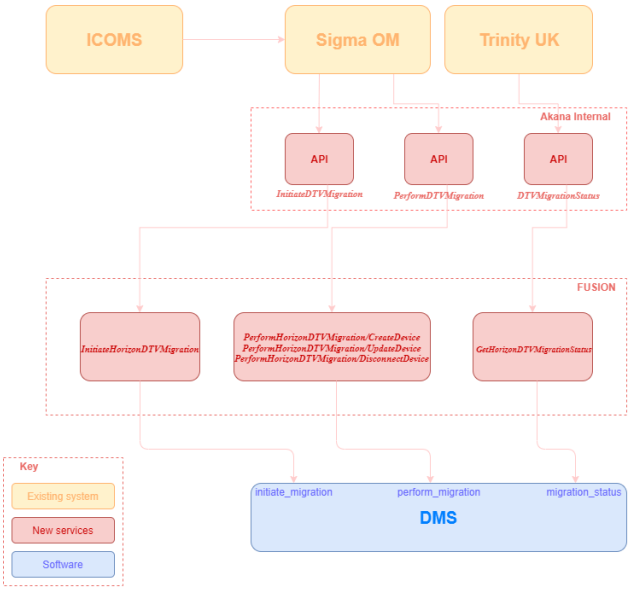
1. Upravljanje ***API***-jima ***(API Management)***
2. Bezbjednost ***API***-ja
3. Digitalna transformacija
4. Digitalno bankarstvo i finansijski servisi
5. Mikroservisi
6. Integracije i medijacije

7. PRIMJER UPOTREBE AKANA PLATFORME

Digitalna revolucija dovela je i do razvoja oblasti telekomunikacija u kojoj je Akana platforma našla široku primjenu. ***Horizon 4 Reference Design Kit (HZN4RDK)*** je nova televizijska platforma za zabavu koja je u Velikoj Britaniji predstavljena kao ***V6 box*** i koristi se u kombinaciji sa ***TiVo (Television In, Video Out)*** softverom. Posjeduje prijemnik signala modernog dizajna, čija je osnovna karakteristika sposobnost veoma brzog prijema i obrade signala sa visoko kvalitetnim ***4K Ultra HD*** prikazom slike. Još jedna od unapređenja koje ova platforma posjeduje u odnosu na ***V6 box*** je to što je ručni daljinski upravljač zamijenjem daljinskim upravljačem sa glasovnim mogućnostima. Sastavni dio unapređene platforme je i mobilna aplikacija koja omogućava gledanje sadržaja platforme u pokretu, nastavljanje gledanja u slučaju prekida, snimanje, ali i skidanje željenog sadržaja koji će korisnicima biti dostupan u bilo kom trenutku, čak i ako ne postoji mobilna ili ***WiFi*** mrežna pokrivenost. Integracija sa ***YouTube*** i ***Netflix*** aplikacijama je, takođe, omogućena i to sa izuzetnim kvalitetom prikaza slike. Informacije o funkcionalnostima koje nova televizijska platforma posjeduje preuzete su iz saopštenja za javnost [9].

TiVo je digitalni video snimač koji omogućava snimanje emisija, serija, filmova po određenim kriterijumima kao što su naslov, žanr, ime glumca, reditelja, ali i gledanje istih [10]. ***Horizon 4*** predstavlja unapređenu verziju ***V6 box*** platforme, koja je napravljena da nadoknadi funkcionalnosti koje je ***V6 box*** koristio od ***TiVo***-a, nadoknadi nedostatke i na taj način zamijeni ***TiVo***. ***TiVo*** omogućava slične funkcionalnosti kao i ***Horizon 4***, samo što je kvalitet prikaza i zvuka mnogo bolji u ***Horizon 4*** softverskoj platformi.

Ukoliko određeni sistemi, čiji korisnici koriste neke od već pomenutih telekomunikacionih usluga, žele da se pretplate na novu, unapređenu platformu za zabavu, potrebno je izvršiti odgovarajuću softversku migraciju, odnosno, prenos podataka, naloga i funkcionalnosti sa jednog softvera na drugi. Akana platforma se koristi za pravljenje ***API***-ja koji će omogućiti iniciranje, izvršavanje i provjeravanje statusa migracije sa ***V6 box*** platforme na ***Horizon 4*** platformu.



Slika 1. *Dijagram rješenja*

Na slici 1 prikazan je dijagram koji prikazuje sisteme koji su uključeni u migraciju. Sitemi koju učestvuju u migraciji su ***ICOMS (Integrated Communications Operations Management system), Sigma OM (Sigma Order Management), Akana, Oracle Fusion Middleware ili skraćeno Fusion, DMS (Device Migration Service)*** i ***Trinity UK***. Ovo rješenje će koristiti ***ICOMS*** sistem da pošalje obavještenje ***Sigma OM*** sistemu o kreiranom nalogu, odnosno, narudžbi za migraciju. ***Sigma OM*** je

aplikacija koja je zadužena za orkestraciju i upravljanje narudžbinama. ***Fusion*** je sistem koji posjeduje servise za iniciranje, izvršavanje i provjeru statusa migracije. ***ICOMS*** je u mogućnosti da direktno pristupi ***Fusion***-ovim servisima, ali zbog sigurnosnih razloga Akana je uključena u proces migracije. ***DMS*** je softversko rješenje koje omogućava migraciju. ***Trinity UK*** je aplikacija koja se koristi za provjeru statusa migracije u ***Horizon 4***.

Akana ***Community Manager (CM)*** i ***Policy Manager (PM)*** komponente se koriste za implementaciju ovog rješenja. ***CM*** je komponenta koja se koristi za implementaciju ***API***-ja, kreiranje dokumentacije, kontrolu pristupa, analitiku, a sadrži i alat za testiranje. Kreiranje ***API***-ja, aplikacija, licenci, ali i njihova izmjena i brisanje, dodavanje, izmjena i brisanje opisa, slika, metoda, dodjeljivanje svih vrsta polisa ***API***-ju vrši se u ***Community Manager*** komponenti. Pretraživanje ***API***-ja, aplikacija i licenci je moguće putem ***CM*** komponente. Okuplja potrošače i razvojni tim na jednom mjestu i omogućava im testiranje aplikacija preko ***API***-ja kojima su dodijeljene. ***Policy Manager*** komponenta, kao što joj i samo ime kaže, upravlja polisama. Omogućava kreiranje svih vrsta polisa, čija aktivacija je neophodna da bi one postale vidljive u ***CM*** komponenti i bile spremne za korišćenje i dodjeljivanje odgovarajućim ***API***-jima i aplikacijama. ***Community Manager*** i ***Policy Manager*** su sinhronizovani, tako da je pretraživanje i pregled ***API***-ja i aplikacija kreiranih u ***CM***-u, moguće i preko ***PM***-a. U ***PM***-u se podešavaju pravila za dodavanje adresa u bijelu i crnu listu, dodavanje, kreiranje, izmjena i brisanje sertifikata iz skladišta, potrebna preslikavanja u procesu, dodavanje autentifikacije ka krajnjem sistemu. Pregled, pretraživanje i izvoženje logova je, takođe, moguće putem ***Policy Manager***-a.

8. ZAKLJUČAK

U radu je data jasna definicija programskih interfejsa aplikacija (***API***-ja), pregled njihovih osobina i funkcionalnosti. Opisana je njihova namjena, uloga i primjena u poslovnom svijetu u okviru različitih oblasti primjene. Omogućena je softverska migracija upotrebom ***API***-ja za iniciranje, izvršavanje i provjeravanje statusa migracije. Izvršeno je kreiranje, opisivanje ***API***-ja, povezivanje sa odgovarajućim potrošačima i krajnjim sistemima, kao i dodjeljivanje odgovarajućih polisa, licenci, vrste autentifikacije, prava pristupa, kako bi se ostvarila bezbjedna komunikacija između sistema koji su učestvovali u njoj i zaštitili podaci koji se prenose. Izvršeno je testiranje kojim je potvrđena funkcionalnost kreiranih ***API***-ja. Izvršen je kratak pregled platformi za upravljanje ***API***-jima koji je omogućio jasan uvid koja od njih je bila u mogućnosti da obezbijedi kvalitet, efikasnost i stabilnost u isto vrijeme. Upotrebom Akana platforme i praćenjem opisanih koraka životnog ciklusa ***API***-ja, kreirali su se stabilni i efikasni ***API***-ji, koji su omogućili uspješno izvršavanje funkcionalnosti za koje su i bili namijenjeni. ***API***-ji su obezbijedili odgovarajući nivo sigurnosti i kontrole pristupa, zaštitu od zloupotrebe podataka, a Akana je platforma koja je održala stabilnost prilikom integracije sa drugim sistemima. Uspješno je nadoknadila nedostatke koje ***Apigee, Kong Enterprise, Postman, 3Scale*** platforme/alati za upravljanje ***API***-jima imaju, a predstavlja i sinonim za stabilan, bezbjedan, robusan sistem sa dobrim performansama izvršavanja, pa je zbog toga zaslužila da bude jedna od najpopularnijih platformi koja se koristi za upravljanje ***API***-jima. Omogućila je upravljanje ***API***-jima od njihovog nastanka, dizajn faze, preko razvoja i raspoređivanja, preko faze u kojoj se vrši nadgledanje, analiza, optimizacija i poboljšavanje performansi, zaključno sa fazom povlačenja. Ponudila je mehanizme za zaštitu ***API***-ja i očuvala je privatnost podataka koji se razmjenjuju u komunikaciji upotrebom različitih mehanizama za autorizaciju, autentifikaciju, prevenciju napada i protiv zloupotrebe podataka. Uz pomoć Akana platforme omogućeno je jednostavno kreiranje ***API***-ja koji su izvršili iniciranje, izvršavanje i provjeravanje statusa migracije sa ***V6 box*** platforme korišćene u kombinaciji sa ***TiVo*** softverom na ***Horizon 4*** platformu zasnovanu na ***RDK***.

9. LITERATURA

[1] https://bbvaopen4u.com/en/actualidad/infographic-what-api (pristupljeno u avgustu 2020.)

[2] C. Ferris and J. Farrell, " What are Web Services?", 2003.

[3] B. Rector, L. Osterman, W. Messmer, " API lifecycle platform and version management", US, 2017.

[4] J. Vester, "RESTful API Lifecycle Management", 2017.

[5] D-F. Yu, C-Y. Chang, H. C. Jiau, K-F. Ssu, " Which API Lifecycle Model is the Best for API Removal Management?", Tainan, Taiwan, 2017.

[6] https://training.mulesoft.com/site/coursePlayer.do?dispatch=show&courseSessionId=ac4377ec-985b-11ea-9f48-0cc47adeb5f8 (pristupljeno u septembru 2020.)

[7] https://www.capterra.com/api-management-software/ (pristupljeno u avgustu 2020.)

[8] J. Reschke, " The 'Basic' HTTP Authentication Scheme", 2015.

[9] https://2zn23x1nwzzj494slw48aylw-wpengine.netdna-ssl.com/wp-content/uploads/2018/09/Press-Release-Liberty-Global-launches-next-generation-entertainment-platform-Horizon-4.pdf (pristupljeno u septembru 2020.)

[10] https://www.tivo.com/ (pristupljeno u septembru 2020.)

Kratka biografija:

Danijela Zelenović rođena je u Zvorniku 1995. god. Završila je gimnaziju „Gimnazija i srednja stručna škola Petar Kočić" u Zvorniku 2014. godine kao nosilac Vukove diplome. Osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu završila je 2018. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Računarstva i automatike - Primenjene računarske nauke i informatika odbranila je 2020.god.
kontakt: danijela.zelenovic@hotmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE48Matovic

ANALIZA SENTIMENTA TEKSTA NA SRPSKOM JEZIKU KORIŠĆENJEM DUBOKOG UČENJA

SENTIMENT ANALYSIS OF TEXT IN SERBIAN LANGUAGE USING DEEP LEARNING

Stevan Matović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Analiza sentimenta je naučno polje koje se bavi analizom mišljenja, stavova i emocija ljudi koji su napisali određeni tekst. Ovakva analiza može se koristiti u svrhe praćenja brendova, poboljšanja korisničke podrške, analize proizvoda, istraživanja tržišta ili u svrhe kreiranja sistema za preporuke. U ovom radu obučeno je i upoređeno više modela mašinskog učenja za zadatku analize sentimenta. Recenzije korišćene kao skup podataka prikupljene su sa jednog od sajtova za dostavu hrane u Srbiji. Recenzije sadrže skup ocena koje će biti korišćene kao pokazatelj emocionalne polarnosti. Trenirana su i upoređena tri modela mašinskog učenja sa različitim vrstama vektorizacije i rezultati su upoređeni sa pristupom transfera učenja. Transfer učenja je metoda dubokog učenja gde se model obučen da reši jedan problem koristi kao polazna tačka pri rešavanju drugog problema. Za transfer učenja korišćen je model dubokog učenja zasnovan na transformerima.*

Ključne reči: *Analiza sentimenta, mašinsko učenje, duboko učenje, transfer učenja*

Abstract – *Sentiment analysis is a scientific field that deals with the analysis of opinions, attitudes and emotions of people who have written a certain text. Such analysis can be used for purposes like brand monitoring, customer service improvement, product analysis, market research, creating recommendation systems etc. In this paper, several machine learning models are trained and compared for the task of sentiment analysis. Reviews used as a dataset were collected from one of the food delivery websites in Serbia. Reviews contain a set of ratings that will be used as an indicator of emotional polarity. Three models of machine learning with different types of vectorization were trained and compared with results of transfer learning approach. Transfer learning is a method of deep learning where a model trained to solve one problem is used as a starting point in solving another problem. Deep learning model based on transformers is used for transfer learning.*

Keywords: *Sentiment analysis, Machine Learning, Deep Learning, Transfer Learning*

1. UVOD

Šta drugi ljudi misle o proizvodu ili temi oduvek je bio važan faktor za ljude pri donošenju odluka. Pre nego što se pojavio internet, mnogi su pitali prijatelje i poznanike za preporuku pri odabiru restorana, mesta koja vredi posetiti, određenog proizvoda itd.

Internet je je omogućio ljudima da saznaju mišljenja i iskustva velikog broja ljudi koji nisu njihovi prijatelji ili poznanici, ali to je takođe dovelo do povećanja broja ljudi koji dele svoje mišljenje sa nepoznatim ljudima preko interneta. Sve ovo je rezultiralo ogromnim količinama nestruktuiranih tekstualnih podataka.

Analiza sentimenta se bavi određivanjem emocionalne polarnosti takvog teksta. Pristupi za analizu sentimenta se razlikuju po metodama pretprocesiranja, kao i po samim algoritmima i tipovima algoritama. Zadatak ovog rada će biti poređenje različitih pristupa nadgledane analize sentimenta za recenzije na srpskom jeziku. Srpski jezik spada u grupu visoko flektivnih i morfološki bogatih jezika, koji koriste puno različitih sufiksa kako bi izrazili različite gramatičke, sintaktičke ili semantičke karakteristike.

Uporedićemo pristup transfera učenja (eng. *transfer learning*) sa tradicionalnim pristupima mašinskog učenja. Transfer učenja je metoda mašinskog učenja gde se model treniran na velikom skupu podataka da reši jedan problem koristi kao polazna tačka pri rešavanju drugog problema. Ovo je popularan pristup u dubokom učenju gde se modeli pretreniraju da reše problem uz korišćenje značajnih komputacionih i vremenskih resursa nad velikim količinama nelabeliranih ili labeliranih podataka da bi se potom prilagodili za druge zadatke (moguće i različite od zadatka za pretreniranje, ali u istom domenu) i na taj način iskoristili prethodno stečeno znanje.

Transfer učenje je imalo mnogo uspeha u domenu mašinske vizije, gde se danas modeli retko treniraju „od nule", već se prilagođavaju već pretrenirani modeli. Transfer učenje se pokazalo kao veoma korisno i u domenu obrade prirodnog jezika.

U ovom radu uporedićemo tradicionalne metode nadgledane analize sentimenta (Naive Bayes, Support Vector Machine, logistička regresija) sa metodom transfera učenja koristeći duboki model reprezentacije jezika BERT [1].

Pod pojmom dubokog učenja (eng. Deep Learning) smatra se primena veštačkih neuronskih mreža (skraćeno neuronskih mreža) koje se sastoje od više slojeva. Duboko učenje koristi sposobnost neuronskih mreža da

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

naprave reprezentacije različitih nivoa kompleksnosti u zavisnosti od dubine sloja. Što je dublji sloj to su reprezentacije kompleksnije, otuda naziv duboko učenje.

Zahvaljujući brzom razvoju u poslednjoj deceniji, neuronske mreže danas predstavljaju jedno od najznačajnijih i najprimenljivijih metoda mašinskog učenja.Bez obzira na veliki broj primena koje neuronske mreže imaju, ipak je važno istaći tipove problema u čijem rešavanju su se one pokazale najbolje - a to su problemi sa velikom količinom podataka koji se nalaze u svojoj sirovoj reprezentaciji. Ovo svakako znači da neuronske mreže nisu rešenje za svaki problem. Male količine podataka veoma lako dovode do overfit-ovanja, dok podaci koji nisu u svojoj sirovoj reprezentaciji već u vektorskom formatu sa već određenim atributima od interesa ne koriste osnovnu prednost neuronskih mreža, a to je svakako sposobnost da same konstruišu kompleksne reprezentacije nad sirovim podacima.

2. METODOLOGIJA I ALATI

U ovom poglavlju predstavljena je metodologija analize sentimenta teksta recenzija na srpkom jeziku koje nose pozitivan ili negativan sentiment. Detaljno će biti predstavljen metod prikupljanja podataka, tehnike preprocesiranja i samo obučavanje i evaluacija različitih modela mašinskog učenja.

2.1. Prikupljanje podataka

Podaci korišćeni u ovom radu prikupljeni su sa jednog od sajtova za dostavu hrane u Srbiji. Namena ovog sajta je povezivanje ljudi sa velikim brojem restorana koji nude uslugu dostave hrane. Nakon naručivanja hrane putem ovog servisa, korisnik ima opciju da unese recenziju za dati restoran. Recenzija može da sadrži naslov, tekst, sliku i set ocena. Korisnik ima opciju da dodeli 4 različite ocene i to za sledeće kategorije: kvalitet hrane, izbor hrane, cenu i uslugu.

Za svaku kategoriju korisnik bira da li će da je oceni sa ocenom od 1 do 5, što znači da svaka recenzija ima minimalno 0 ocena a maksimalno 4. Za prikupljanje podataka korišćen je python programski jezik, biblioteka Beautiful Soup [2] kao i Selenium WebDriver [3].

Ukupno je prikupljeno 75857 recenzija, od kojih je 60807 za restorane u Beogradu a 15050 za restorane u Novom Sadu. Skup podataka neće biti javno dostupan i korišćen je isključivo za eksperimentalne svrhe.

2.2. Pretprocesiranje podataka

Pretprocesiranje podataka je posebno važan korak za zadatke obrade teksta. Tekst se pretvara u savladiv oblik kako bi algoritmi mašinskog učenja mogli bolje da ga obrade. U ovom radu nekoliko različitih metoda pretprocesiranja su primenje a to su:

1. uklanjanja dijakritika
2. izbacivanje stop reči
3. normalizacija emotikona i znakova
4. prebacivanje u mala slova
5. stemovanje reči
6. upweighting
7. vektorizacija

Srpski jezik sadrži nekoliko slova sa dijakritičkim znakovima(č,ć,đ,š,ž). Pri korišćenju srpskog jezika na računaru korisnici često ignorišu dijakritičke znakove, zbog ovog razloga je odlučeno da se karakteri sa dijakritikom svedu na svoju osnovnu formu. Ovo se radi kako ne bi došlo do različite interpretacije tokena koji imaju isto značenje (npr. cevapi i ćevapi).

Tokenizacija je proces u kome se tekst (rečenice, paragrafi, dokumenti) predstavlja kao lista tokena. Token može biti na nivou reči, delova reči ili karaktera. U ovom radu rečenice su podeljene na tokene na nivou reči, tako da će rečenica „Hrana je odlična“ biti predstavljena kao sledeća lista tokena [„Hrana“, „je“, „odlična“].

Stop reči su skup reči koje se često koriste u jeziku. Primeri stop reči na srpskom su „a“, „ali“, „i“, „“ itd. Intuicija iza izbacivanja zaustavnih reči je da se uklanjanjem reči sa niskim informacijama iz teksta algoritam može fokusirati na važne reči.

Emotikoni su dobar pokazatelj emocionalnog polariteta. U [4], tweet-ovi koji se završavaju pozitivnim emotikonima poput „:)“ I „:-)“označeni su kao pozitivni a tweet-ovi koji se završavaju negativnim emotikonima poput „:(“ ili „:-(“ su označeni kao negativni.

U ovom radu emotikoni su zamenjeni rečima sličnog polariteta sentimenta. Na primer „:-))“ je zamenjen rečju „odlican“.

Prebacivanje u mala slova je jedna od najprostijih transformacija a ujedno i veoma efektivna. Sva velika slova se prebacuju u mala kako bi se izbegla različita interpretacija tokena sa istim značenjem (npr. Dostava i dostava).

Upweighting je tehnika gde se za neke reči računa kao da su se pojavile više puta nego što zapravo jesu kako bi se povećao njihov uticaj. U ovom radu ova tehnika se koristila za reči iz naslova recenzije (računate su kao da su se pojavile dva puta umesto jednom).

Kako bi modeli mašinskog učenja bili u stanju da razumeju tekstualne podatke neophodno je tekst predstaviti numeričkim vrednostima (vektorima) - ovaj proces naziva se vektorizacija. U ovom radu korišćeno je više tehnika vektorizacije kako bi se uporedili rezultati.

Prva tehnika je predstavljanje teksta kao set n-grama. N-grami su sekvence susednih tokena dužine N iz datog uzorka teksta. U ovom radu eksperimentisano je sa unigramima, bigramima i trigramima.

Drugi način vektorizacije je tf-idf vektorizacija. Tf-idf vektorizator konvertuje tekst u vektor tf-idf vrednosti množenjem broja ponavljanja tokena u recenziji (eng. term frequency ili tf) sa invertovatnim brojem dokumenata u kojima se token pojavljuje (eng. inverse document frequency ili idf).

Ovo znači da će reč imati veći uticaj ukoliko se pojavljuje više puta u recenziji, ali i ako je reč retka(ne pojavljuje se u puno drugih recenzija).

2.3. Treniranje modela

Nakon što su podaci pretprocesirani spremni su za algoritme mašinkog učenja. Analizom prethodnih radova na temu analize sentimenta i klasifikacije teksta generalno, utvrđeno je koji su algoritmi pokazali dobre performanse za ovaj problem i oni su analizirani i u ovom radu. To su algoritmi:

- Logistička regresija
- Naive Bayes
- SVM (Support Vector Machines)

Na performanse svakog od ovih algoritama utiču tehnike pretprocesiranja, kao i način formiranja finalnog skupa atributa. Upravo zato su svi algoritmi primenjeni na skupu, koji prolazi kroz isti podsistem za pretprocesiranje i formiranje skupa atributa. Optimizacija vrednosti hiperparametara svakog algoritma rađena je postupkom ugnježdene unakrsne validacije.

Osim ovih standardnih pristupa isproban je i model dubokog učenja BERT (Bidirectional Encoder Representations from Transformers).

Ovaj model je dizajniran tako da nauči duboke bidirekcione reprezentacije teksta tokom pretreniranja na velikim količinama nelabeliranih podataka. Rezultat ovog procesa, pretrenirani BERT se potom može prilagoditi (eng. fine-tuning) sa samo jednim dodatnim slojem kako bi rešio širok spektar NLP problema.

Pretreniranje BERT-a je izuzetno skupo, konkretno obuka za BERT Base izvedena je na 16 TPU čipova dok je obuka za BERT Large izvedena na 64 TPU čipa. Svako pretreniranje trajalo je 4 dana. Srećom Google je objavio više BERT-a pretreniranih modela. U ovom radu korišćen je BERT Base Multilingual Cased model pretreniran na na 104 jezika (uključujući srpski). Cased označava da se koristi originalna veličina slova i originalni markeri padeža i akcenta.

Prilagođavanje BERT-a za klasifikaciju sekvenci postiže se dodavanje dodatnog klasifikacionog sloja postojećem pretreniranom modelu. Prvi znak u svakom BERT ulazu je poseban token [CLS]. Ulaz u dodati klasifikacioni sloj je konačno skriveno stanje (izlaz transformera) za ovaj token. Ulazni tekst nije unapred pretprocesiran kao za ostale algoritme u ovom radu.

Model je prilagođen u Google Colab[5] okruženju pomoću Tensor procesne jedinice (TPU) sa sledećim hiperparametrima:

- TRAIN_BATCH_SIZE = 32
- EVAL_BATCH_SIZE = 8
- PREDICT_BATCH_SIZE = 8
- LEARNING_RATE = 2e-5
- NUM_TRAIN_EPOCHS = 3.0
- MAX_SEQ_LENGTH = 512

Svi modeli mašinskog učenja su trenirani nad istim trening skupom i testirani nad istim test skupom podataka. Trening i test skup su dobijeni tako što se originalni skup podelio u odnosu 80/20.

U ovom radu kao metrika za upoređivanje korišćena je mikro F1 mera.

3. REZULTATI I DISKUSIJA

U ovom radu istražena je upotreba unigrama, bigrama, trigrama i tf-idf vektorizacije na tri algoritma: Naive Baies, SVM i Logistička regresija. Ovi rezultati su upoređeni sa rezultatima dobijenim korišćenjem pretreniranog BERT modela. Kao metrika korišćena je mikro F1 mera. Rezultati su prikazani u tabeli 1.

Tabela 1. Rezultati eksperimenata

	No upweighting	Upweighting
Naive Bayes + unigrami	0.93	0.93
Naive Bayes + bigrami	0.91	0.92
Naive Bayes + trigrami	0.86	0.88
Naive Bayes + tf-idf	0.88	0.88
Logistička regresija + unigrami	0.93	0.93
Logistička regresija + bigrami	0.90	0.91
Logistička regresija + trigrami	0.85	0.86
Logistička regresija + tf-idf	0.93	0.93
SVM + unigrami	0.93	0.93
SVM + bigrami	0.9	0.91
SVM + trigrami	0.85	0.86
SVM + tf-idf	0.93	0.93
Bert	0.94	

Iz rezultata se može zaključiti sledeće:

- Bigrami rade lošije od unigrama. Ovo je zbog činjenice da su reprezentacije sa bigramima vrlo retko popunjene i ukupna ocena opada za sve algoritme. Bolje pristup bi mogao biti upotreba unigrama i bigrama zajedno.
- Trigrami su dali vrlo loše rezultate. To je zato što su reprezentacije trigramima još ređe popunjene nego od onih sa bigrama (434.333 ukupnih obeležja u poređenju sa 251.583 obeležja za bigrame).
- Za Naive Baies, tf-idf se loše pokazao, ovo je zbog toga što se koristio Multinomial Naive Baies. Ovaj algoritam je više pogodan za klasifikaciju za diskretne podatke.
- Za SVM i logističku regresiju tf-idf radi podjednako dobro kao unigrami.
- Za sve algoritme upweighting je poboljšao f1 metriku za bigrame i trigrame, to je možda zato

što upweighting dodaje nova obeležja i podaci postaju manje retko popunjeni.
- Upweighting međutim nije uticao na unigrame i tf-idf.
- BERT je nadmašio sve pomenute algoritme, bez prethodne obrade ulaznog teksta. To pokazuje da se pretreniranjem na velikim količinama tekstualnih podataka, duboki modeli obučavaju da shvate jezik na neki način, što im omogućava da prenesu to razumevanja na konkretne probleme. Prilagođavanje ovakvih modela nadmašuje modele koji su trenirani od nule.
- Naivni Baies, SVM i Logistička regresija i dalje daju jako dobre rezultate, za relativno malo vreme obučavanja.

Po današnjim standardima dubokog učenja, količina podataka prikupljena u ovom radu je i dalje veoma mala. Može se pretpostaviti da bi sa većom količinom podataka BERT pokazao još bolje performanse u odnosu na standardne modele mašinskog učenja.

4. ZAKLJUČAK

U ovom radu rešavan je problem analize sentimenta tekstualnih podataka na srpkom jeziku. Analiza sentimenta je naučno polje koje se bavi analizom mišljenja, stavova i emocija ljudi koji su napisali tekst. Tekstualni podaci koji su korišćeni su recenzije koje ostavljaju korisnici na sajtu za dostavu hrane. Detaljno su opisane su metode za prikupljanje podataka kao i za preprocesiranje istih.

Podaci su pretprocesirani nekim od standardnih tehnika pretprocesiranja u polju obrade teksta. Nad pretprocesiranim podacima, obučena su tri modela mašinska učenja - Naive Baies, SVM i logistička regresija. Svaki od ovih algoritama testiran je sa 4 različita tipa vektorizacije: unigrami, bigrami, trigrami i tf-idf. Pored ovoga je testirana i upweighting tehnika (brojanje nekih reči kao da su se pojavile dva puta).

Takođe je prilagođen pretrenirani model dubokog učenja BERT kako bi rešio ovaj zadatak analize sentimenta i upoređeni su rezultati sa prethodno navedenim pristupima. Pokazano je da je prilagođen pretrenirani model dubokog učenja nadmašio rezultate algoritama obučenih od nule čak i nad ovim relativno malim skupom podataka. Ovo pokazuje veliki potencijal tehnike transfera učenja.

Ovaj rad bi se mogao proširiti detaljnijim analizama nad većim skupom podataka, kao i poređenjem sa novijim modelima dubokih reprezentacija jezika.

5. LITERATURA

[1] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
[2]Leonard Richardson 2014, BeautifulSoup4 https://www.crummy. com/software/BeautifulSoup
[3] Jason Huggins, et al, 2004. Selenium, https://www.seleniumhq.org
[4] Go, Alec, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." CS224N project report, Stanford 1.12 (2009): 2009.
[5] https://colab.research.google.com/

Kratka biografija:

Stevan Matović rođen je u Beogradu 1995. god. Master rad na Fakultetu tehničkih nauka iz oblasti Računarstva i automatike – Sistemi za istraživanje i analizu podataka odbranio je 2020.god.

kontakt: stevan.matovic95@gmail.com

Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.9
DOI: https://doi.org/10.24867/12BE49Makaric

ПРИМЕНА АЛГОРИТАМА МАШИНСКОГ УЧЕЊА У ПРЕДИКЦИЈИ СТОПЕ САМОУБИСТАВА У ПОЈЕДИНИМ ДРЖАВАМА СВЕТА

THE USE OF MACHINE LEARNING ALGORITHMS IN THE PREDICTION OF SUICIDE RATES IN SOME COUNTRIES OF THE WORLD

Милица Макарић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Самоубиство је сложена појава која вековима привлачи пажњу разних научника. Сваке године, самоубиство је међу 10 водећих узрока смрти у свету међу људима свих узрасних доби. Као озбиљан здравствени проблем захтева нашу пажњу, премда његова контрола и превенција нису нимало једноставни обзиром да на њих могу утицати разни фактори. Из тог разлога, јављају се разне мере које се предузимају у циљу покушаја превенције. Један од тих покушаја спроводи се коришћењем рачунара, односно машинског учења, и разних алгоритама како би се извршила предикција и омогућило спречавање самоубистава у свету. Управо из тога произилази мотивација за овај рад. У раду је представљена предикција стопе самоубистава у различитим државама света. Предикција је извршена помоћу алгоритама машинског учења. Коришћено је више регресионих алгоритама, као и један класификациони алгоритам.*

Кључне речи: *Предикција, Стопа самоубистава, Алгоритми машинског учења*

Abstract – *Suicide is a complex phenomenon that has attracted the attention of various scientists for centuries. Every year, suicide is among the 10 leading causes of death in the world among people of all ages. As a serious health problem, it requires our attention, although its control and prevention are not at all simple, considering that they can be influenced by various factors. For this reason, various measures are being taken to try to prevent it. One of these attempts is carried out using computers, i.e. machine learning, and various algorithms in order to make predictions and enable the prevention of suicides in the world. This is precisely the motivation for this paper. The paper presents the prediction of suicide rates in different countries of the world. Prediction was performed using machine learning algorithms. Several regression algorithms were used, as well as one classification algorithm.*

Keywords: *Prediction, Suicide rates, Machine learning algorithms*

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Александар Ковачевић, ванр. проф.

1. УВОД

Од почетка људске цивилизације постојала је потреба да се пронађе начин како се супротставити аутодеструктивном понашању и како повећати осећај задовољства животом. Светска здравствена организација самоубиство дефинише као чин намерног прекида сопственог живота [1]. Према класичним психијатријским ставовима, суицид је чин повезан са поремећајем нагона за самоодржањем [2].

Поставља се питање да ли је могуће направити предикцију степена самоубиства у различитим земљама, како би се у што већој мери утицало на њихово спречавање. Управо тим проблемом се бави овај рад. У њему ће бити приказано једно решење за процену предикције броја самоубиства у одређеним земљама. Подаци на основу кога је обучен модел представљају праћење броја самоубистава од 1990. до 2016. године. Поред броја самоубистава, скуп података садржи и податке о величини популације у држави, старосно доба, пол, просечна примања. У бројним студијама о суициду пронађена је повезаност суицида и сезонских варијација. Изражен је јасан утицај годишњих доба на суициде, са пиком у пролеће и лето у односу на остала годишња доба, што потврђује да стопа суицида током времена кореспондира са сезонским варијацијама [3]. Из тог разлога је скуп података проширен податком о просечном броју сунчаних сати у години.

2. ПРЕГЛЕД ПОСТОЈЕЋИХ РЕШЕЊА

Предикција самоубистава представља предмет многобројних истраживања. Научни радови који се баве овом тематиком разликују се по избору алгоритама за машинско учење, као и по избору фактора на основу којих се врши предикција. У наставку су наведени они радови који су имали највећи утицај на овај рад.

У раду [4] је извршена предикција ризика покушаја самоубистава кроз време коришћењем различитих техника машинског учења. У овом раду су за алгоритме предикције одабрани логистичка регресија (енг. *Logistic regression*), и алгоритам случајне шуме (енг. *Random Forest*). Предикција је вршена на основу медицинских података из клиничких здравствених картона у којима су забележени случајеви повреда за које се зна или сумња да су самонанешене. Такође, у обзир су узети и здравствени картони у којима нису забележени подаци о покушајима самоубиства. Модел је показао добре резултате са следећим вредностима

коришћених евалуционих параметара: АУЦ = 0.84, прецизност = 0.79, одзив = 0.95, и успео је да тачност погађања побољша са 720 дана на 7 дана пре покушаја самоубиства.

Рад [5] се бави предикцијом идеја појединаца о самоубиству на основу социо-демографских, физичких и психолошких обележја. Предикција се врши употребом *Random Forest* алгоритма. Као неки од најбитнијих параметара су издвојени: депресија, ниво стреса у свакодневном животу, пол, старост. Модел предвиђања постигао је добре перформансе (AUC = 0,85) са следећим вредностима валидационих параметара: *accuracy* = 0,821, *sensitivity* = 0,836 и *specificity* = 0,807.

Рад [6] истражује тезу да на суицидно понашање утиче сунчева светлост. За истраживање су коришћени подаци о свим самоубиствима у Аустрији у периоду од 1970-2010. године. Подаци о просечном броју сунчаних сату у току дана добијени су са 86 репрезентативних метеоролошких станица. Као резултат овог истраживања изведен је закључак да постоји позитивна корелација између броја самоубистава и броја сунчаних сати на дан самоубиства, као и до 10 дана пре самоубиства.

3. ПРИКУПЉАЊЕ И ПРИПРЕМА ПОДАТАКА

У овом поглављу је детаљно описан скуп података који је коришћен у пројекту. Наведени су полазни скупови података, као и све трансформације које су над њима примењене како би настао скуп података који је даље коришћен у раду. Такође, представљене су и статистичке анализе које су извршене над поменутим подацима.

3.1. Редукција сувишних обележја

Ово поглавље описује поступак уклањања сувишних обележја. Иницијални скуп података представљао је скуп података преузет са веб странице која садржи велики број различитих скупова података. У питању је сајт *kaggle.com* [7]. У питању је скуп података који поред социо-економских информација, садржи и информације о броју самоубистава за 101 државу. Информације се односе на период од 1985. до 2016. године. Анализом овог скупа података је утврђено да за атрибут *индекс хуманог развоја* велики број редова има недостајуће вредности. Из тог разлога је ова колона изузета из скупа података. Даљом анализом је утврђено да постоји још неколико колона које не доприносе даљем коришћењу скупа података. Једна од тих колона јесте *назив генерације*. Наиме, у скупу података постоји колона *старосна група*, и за сваку старосну групу постоји одговарајући назив генерације. Како су ове две колоне у потпуној корелацији, донета је одлука да се колона *назив генерације* уклони из скупа података. Колона назив државе са годином је такође искључена из даљег разматрања. Вредности које ова колона садржи представљају конкатениране, односно спојене вредности колона *назив државе* и *година*. Из тога се лако закључује да колона назив државе са годином представља редундантан податак, те је то разлог њеног уклањања. Анализом скупа података је донета одлука да се и колона *бруто домаћи производ по глави становника* уклони. Наиме, вредности које ова колона садржи се могу добити дељењем вредности колоне *бруто домаћи производ* са вредностима колоне *број становника државе*.

3.2. Спајање скупова података

На веб страници Економске комисије за Европу (*UNECE*) је пронађен скуп података који садржи информације о просечним месечним примањима држава кроз године [8]. Конкретно, овај скуп података садржи податке о просечној месечној заради за 56 различитих држава у периоду од 1990. до 2017. године. Обзиром да први скуп података садржи информације о 101 држави, а други скуп о 56 држава, приликом спајања ова два скупа података је утврђено да пресек ова два скупа чине 44 различите државе. Поред тога, ова два скупа се односе и на различит опсег година. Због тога је урађен пресек, те се финални скуп података односи на податке између 1990. и 2016. године.

Како је пронађен податак да је највећи број самоубистава забележен у периоду пролеће-лето [3], у разматрање је узет и скуп података о броју сунчаних сати по години у различитим градовима широм света. Адекватни скуп са подацима о сунчаним сатима у току године по државама није пронађен, те су подаци ручно преузети и обрађени 12 са веб странице wikipedia.org [9]. Дати подаци су били груписани по континентима, и приказани за сваки месец у години. Из тог разлога је, након преузимања података, извршено сабирање броја сунчаних сати по месецу за град сваке државе која постоји у циљном скупу података. На тај начин је добијен податак о броју сунчаних сати за сваку државу циљног скупа података.

Овако формирани скуп података је подељен на тренинг и тест подскупове у односу 75% - 25%. Подела је извршена тако да се у тренинг скупу налазе подаци од 1990. до 2008. године, док су у тест скупу подаци везани за период од 2009. до 2016. године. Овако формиран тренинг скуп података је употребљен за обучавање модела како би се извршила предикција стопе самоубиства.

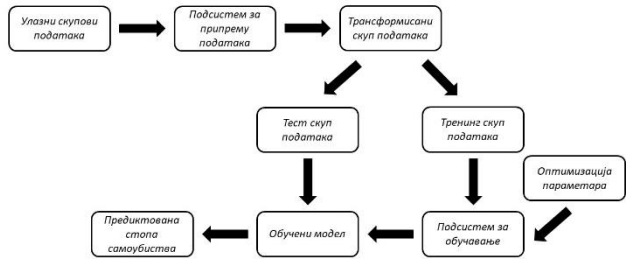
3.3. Трансформације над подацима

Обзиром да су вредности атрибута држава (*country*), пол (*sex*) и година (*age*) у оригиналном скупу података били у текстуалном облику, над овим подацима је било неопходно применити трансформацију у бројеве. Конкретно, назив сваке државе је замењен индексом те државе у скупу података. На тај начин су надаље државе биле представљене бројевима од 0 до 43. Атрибути *sex* и *age* су такође измењени на сличан начин као и атрибут *country*. Вредности атрибута који представља пол су након трансформације садржали само вредности *0* и *1*, уместо вредности *female* и *male*. Атрибут *age* који је садржао 6 текстуалних вредности за 6 различитих старосних група, након промене је презентован вредностима *0-5*.

4. МЕТОДОЛОГИЈА И АЛАТИ

У овом поглављу је представљена методологија која је примењена за предикцију стопе самоубистава над

претходно описаним скупом података. Први део ове целине је задужен за приказ структуре система, заједно са свим подсистемима које обухвата. Обзиром да је у претходном поглављу детаљно описан први подсистем, у овом поглављу ће акценат бити стављен на други подсистем целокупног система – подсистем за обучавање. Подсистем за обучавање се састоји из различитих алгоритама машинског учења. У оквиру овог система је за предикцију стопе самоубистава одабрано неколико различитих регресионих модела, као и један класификациони модел. На слици 1 се налази скица структуре читавог система.



Слика 1. *Шематски приказ структуре система*

4.1. Регресиони модели подсистема за обучавање

Централни део истраживања овог рада фокусиран је на испитивање перформанси различитих алгоритама машинског учења приликом 22 предикције стопе самоубистава. Анализом радова који се баве сличном тематиком као и овај рад, утврђено је који су алгоритми оптимални за овај проблем, и они су и анализирани у наставку.

4.1.1. Линеарна регресија

Линеарна регресија представља један од најједноставнијих и најчешће коришћених модела машинског учења. Односи се на сваки приступ моделовања између једне или више зависних променљивих (означених са *Y*), и једне или више независних променљивих (означених са *X*), на такав начин да модел линеарно зависи од непознатих параметара процењених из података. Обзиром да је у овом раду циљ предикција, линеарна регресија се користи за подешавање предиктивног модела према разматраном скупу података вредности *Y* и *X*. Након развоја оваквог модела, уколико је дата вредност за *X* без припадајуће вредности за *Y*, модел се може употребити за предвиђање вредности *Y*.

4.1.2. Support Vector Regression

Метод потпорних вектора (енг. *Support Vector Machines*) је алгоритам који је првобитно осмишљен за решавање класификационих проблема. Тек касније је овај алгоритам адаптиран за регресионе проблеме. У том случају се користи назив регресија потпорних вектора (енг. *Support Vector Regression*). Алгоритам SVM је заснован на једноставној идеји, а то је да се дефинише хиперраван која раздваја податке који припадају различитим класама. Кључна разлика између метода потпорних вектора за класификацију и за регресију је та што се у случају регресије не траже тачна предвиђања, као што је у линеарно раздвојивом случају код класификације био захтев. У пројекту који је описан у овом раду је за алгоритам *SVR* извршена оптимизација два параметра: параметар C (мера којом се пенализује грешка) и избор кернел функције. У овом примеру се као најпогоднија показала *RBF* кернел функција. Емпиријски је утврђено да оптимална вредност параметра C за претходно описани скуп података износи 1000.

4.1.3. Gradient Boosted Tree

Алгоритам *Gradient Boosted Tree* се у српском језику може пронаћи под називом *алгоритам појачавања градијената*. Главна идеја код *boosting* алгоритама јесте идеја да се слаб предиктор може модификовати да постане бољи. Слаб предиктор или слаб ученик се дефинише као онај чији је учинак бар мало бољи од случајне шансе. *Појачавање* је ансамбл техника у којој се предиктори не праве независно, већ узастопно. Ова техника користи логику у којој наредни предиктори уче на грешкама претходних предиктора. Будући да нови предиктори уче на грешкама које су починили претходни предиктори, потребно је мање итерација да би се приближили стварним предвиђањима. Као слаб предиктор се користе стабла одлучивања (енг. *decision trees*). У примеру за предикцију стопе самоубиства који је обрађиван у овом раду је за алгоритам *Gradient Boosted Tree* извршена оптимизација неколико параметара: *n_estimators* (број стабала одлучивања) = 100, *lerning_rate* (стопа учења) = 0.4 и *max_depth* (максимална дубина стабла одлучивања) = 4.

4.1.4. Extreme Gradient Boosting

Extreme Gradient Boosting или скраћено *XGBoost* алгоритам је алгоритам који у последње доба доминира примењеним машинским учењем. Овај алгоритам представља имплементацију стабала одлучивања појачаних градијената дизајнираних за рачунарску брзину и перформансе модела. *Extreme Gradient Boosting* је једна од имплементација *Gradient Boosting* концепта, али оно што *XGBoost* чини јединственим јесте то што користи уређенију формализацију модела за контролу прекомерног уклапања, што му даје боље перформансе. Модел алгоритма *XGBoost* који је коришћен у овом раду је оптимизован. Конкретно оптимизовани су параметри: *n_estimators*=200, *learning_rate*=0.16 и *max_depth*=4.

4.1.5. Random Forest

Алгоритам *Random Forest* се у нашем језику преводи као *алгоритам случајних шума*. То је алгоритам надгледаног учења који се користи и за регресију и за класификацију. Случајна шума је метод учења који делује конструисањем вишеструких стабала одлучивања на узорцима података. Овај метод креира велики број стабала одлучивања у време тренирања, у сврху елиминисања шума и аутлајера (енг. *outlier*). Од креираног скупа стабала одлучивања, метод случајне шуме добија предвиђање од сваког стабла појединачно, и на крају бира најбоље решење гласањем. За потребе овог истраживања, алгоритам Random Forest је оптимизован. Оптимизација је урађена за следеће параметре: *n_estimators*=20, *max_depth*=50 и *min_samples_split*=4.

4.2. Класификациони модел подсистема за обучавање

У сврху класификационог модела за предикцију стопе самоубистава којом се бави овај рад, одабран је алгоритам *Random Forest*. Како је његов метод рада описан у претходном поглављу, ово поглавље ће бити усмерено на приказ оптимизационих параметара који су употребљени у класификационом моделу. Оптимизовани су исти параметри као и у случају регресије, и њихове оптимизоване вредности износе: *n_estimators*=20, *max_depth*=15, *min_samples_split*=6.

5. ЕКСПЕРИМЕНТАЛНА ЕВАЛУАЦИЈА

Као мера евалуације перформанси регресионих модела, коришћена је R^2 мера, као и корен средње вредности квадрата грешке (енг. *Root Mean Square Error – RMSE*). У табели 1 су приказане вредности ових параметара које су добијене евалуацијом регресионих модела.

Табела 1. *Резултати регресије на тест скупу*

	R^2	*RMSE*
Linear Regression	0.65	493.4
Support Vector Regression	0.26	704.37
Gradient Boosted Tree	0.97	135.03
XGBoost	0.98	128.46
Random Forest	0.95	179.8

На основу података приказаних у табели 1, закључује се да је за проблем описан у овом раду најбоље резултате постигао алгоритам *XGBoost*. Коефицијент детерминације за овај алгоритам износи 0.98, што је врло близу максималној вредности. Такође, овај алгоритам је постигао и најмању вредност за *RMSE*. Веома добре резултате се постигли и алгоритми *Gradient Boosted Tree* и *Random Forest*, те се и они могу сматрати погодним алгоритмима за овакав проблем.

За проблем класификације је као мера евалуације коришћена F_1 мера, као и параметри прецизност (енг. *precision*) и одзив (енг. *recall*). У табели 2 су приказани резултати евалуације класификационог проблема.

Табела 2. *Резултати класификације на тест скупу*

	F_1	*precision*	*recall*
Random Forest	0.748	0.752	0.75

Као што се из претходне табеле може уочити, вредности сва три разматрана параметра имају приближно исту вредност која износи *0.75*. Обзиром да максимална вредност за ове параметре износи *1*, може се закључити да су резултати добијени класификацијом у великој мери прихватљиви.

6. ЗАКЉУЧАК

У раду је приказан систем за предикцију стопе самоубиства појединих држава света на основу аллгоритама машинског учења. Приликом предикције је коришћено више различитих регресионих модела, као и један класификациони модел. Приликом предикције су, осим основних података о државама, коришћени и одређени специфични фактори попут просечних месечних примања грађана и броја сунчаних сати у години. Решење овог проблема довело би до знања који то фактори доводе индивидуу до ситуације да почини самоубиство, што би могло помоћи у превенцији таквих ситуација.

Скуп података који је коришћен у овом раду садржи податке о 44 различите државе у периоду од 1990. до 2016. године. Овај скуп је подељен на тренинг и тест подскупове у односу 75% - 25%. Након тога је путем регресије и класификације обучено више модела на тренинг скупу података. Најбољи резултати су постигнути приликом употребе регресионог модела *XGBoost*. У том случају је постигнута R^2 вредност од 0.98. Даљи правци развоја овог решења обухватају добављање података о другим државама, добављање података о годинама пре 1990. и након 2016. године, као и коришћење других алгоритама машинског учења за предикцију.

7. ЛИТЕРАТУРА

[1] World Health Organization, „Preventing suicide: A global imperative“, Geneva, 2014.

[2] Д. Марчинко, „Модели разумјевања суицидалнога понашања“, Медицинска наклада Загреб, 2011.

[3] Ч. Милић, „Сезонске варијације – фактор ризика за настанак суицида“, Медицински преглед Нови Сад, 2010.

[4] C. G. Walsh, J. D. Ribeiro, J. C. Franklin, „Predicting Risk of Suicide Attempts Over Time Through Machine Learning“, Florida State University, Florida, 2017.

[5] S. Ryu, H. Lee, D. K. Lee, K. Park, „Use of a Machine Learning Algorithm to Predict Individuals with Suicide Ideation in the General Population“, Republic of Korea, 2018.

[6] B. Vyssoki, N. D. Kapusta, N. Praschak-Riede, G. Dorffner, M. Willeit, „Direct Effect of Sunshine on Suicide“, Austria, 2014.

[7] https://www.kaggle.com/russellyates88/suicide-rates-overview1985-to-2016 (приступљено у октобру 2020.)

[8] „Gross Average Monthly Wages by Country and Year”, [Online] Available: https://w3.unece.org (приступљено у октобру 2020.)

[9] https://en.wikipedia.org/wiki/List_of_cities_by_-sunshine_duration (приступљено у октобру 2020.)

Кратка биографија:

Милица Макарић рођена је у Новом Саду 1996. године. Факултет техничких наука у Новом Саду, смер Рачунарство и аутоматика, уписала је 2015. године. Основне академске студије је завршила 2019. године, након чега је уписала мастер академске студије на Факултету техничких наука, смер Рачунарство и аутоматика.
контакт: makaric.milica@gmail.com

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.3.049
DOI: https://doi.org/10.24867/12BE50Kostovski

BIOMEDICINSKI ULOŽAK ZA CIPELE ZA KOREKCIJU HODA KOD DECE

BIOMEDICAL INSOLE FOR WALKING CORRECTION AMONG CHILDREN

Aleksandar Kostovski, *Fakultet tehničkih nauka, Novi Sad*

Oblast - ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Kako bi se sprečile deformacije hoda u kasnijem dobu, u ovome radu predstavljen je „Biomedicinski uređaj za cipele za korekciju hoda kod dece“ kako bi se popravila statika čitavog tela i smanjio pritisak u svim zglobovima iznad stopala (skočni, koleno i kukovi). U prvom delu rada predstavljena je ideja kako obezbediti uređaj koji će dete usmeravati da pravilno hoda, dok se drugi deo rada sastoji od proof – of - concepta samog uređaja.*

Ključne riječi : *FSR senzor, IOT, Mikrokontroler*

Abstract – *In order to prevent gait deformities at a later age, this paper presents a “Biomedical insole for walking correction among children” in order to improve the statics of the whole body and reduce the pressure in all joints above the feet (ankles, kness and hips). The first part of the paper presents the idea of how to provide a device that will guide the child to walk properly, while the second part of the paper consist of a proof – of - concept of the device itself.*

Keywords : *FSR sensor, IOT, Microcontroler*

1. UVOD

Deformacija stopala kod dece se javlja u dva slučaja. Jedan broj dece se rađa sa deformitetima stopala, a kod drugih je on stečen u nekoj od razvojnih faza. Izražene deformacije se uočavaju odmah po rođenju i uz primenu fizikalne terapije bivaju uspešno korigovane. Deformitet može da bude prisutan samo na jednoj ili na obe noge. Ukoliko se ništa ne preduzme ili se ne istraje u primeni vežbi, dete će prohodati sa stopalima u nepravilnom položaju što će dovesti do loše šeme hoda koja ostaje za čitav život. Isto će se dogoditi ukoliko do pojave deformiteta dođe kasnije. Sve ovo ima za posledicu narušenu statiku čitavog tela, svih zglobova iznad stopala (skočnih, kolena i kukova) kao i kičmenog stuba [1].

U samom ulošku je predviđeno da se nalaze FSR senzori, koji mere kolikim silama dete deluje stopalom na njih. Unutar uloška se nalazi i mikrokontroler koji prikuplja podatke sa FSR senzora i pomoću Bluetootha te podatke o silama šalje aplikaciji na mobilnom telefonu. U aplikaciji postoji mogućnost da se unese trenutna masa deteta i da se na osnovu baze podataka (koja se nalazi u sklopu aplikacije) odredi koji opseg sila je prihvatljiv za dete. Aplikacija nudi mogućnost da beleži koliko je pravilnih i nepravilnih koraka dete realizovalo.

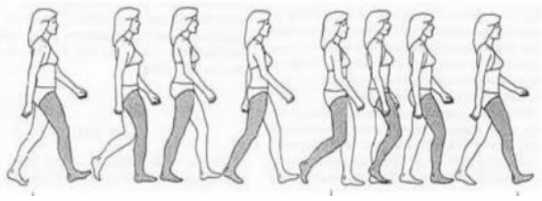
NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Marjan Urekar.

Na taj način se daje mogućnost roditeljima i lekarima da prate napredak na dnevnom, nedeljnom i mesečnom nivou. Ukoliko dete ne napravi dva pravilna koraka zaredom, telefon emituje glasovnu poruku roditelja kako opominje dete da se potrudi da popravi hod. Cilj jeste da dete stekne rutinu kako pravilno da hoda kako bi ispravilo svoje držanje. Što manje opomena bude bilo, to je znak da se dete trudi i da ostvaruje napredak.

2. TEHNIČKO REŠENJE SISTEMA

U samom ulošku je predviđeno da se nalazi otpornik osetljiv na silu pritiska (Force – sensing resistor – FSR), čija je uloga da meri silu kojom dete svojim stopalom deluje (vrši pritisak) na podlogu (slika 1). Što je veće delovanje sile na osetljivi deo, smanjuje se otpornost senzora [2]. Pravilno hodanje treba da bude kao na slici 2.

Sl. 1. *FSR otpornik osetljiv na pritisak [2]*

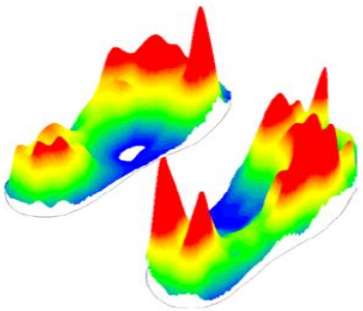


Sl. 2. *Proces pravilnog hodanja [3]*

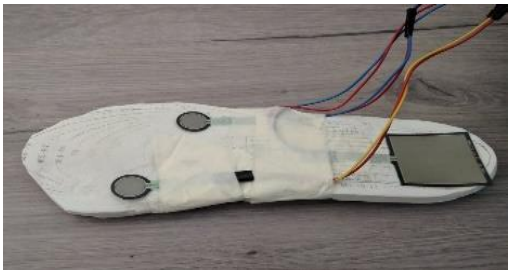
Pritisak stopala prilikom kretanja se raspoređuje na 3 dela: centralni deo pete, spoljašnji deo stopala (metatarzalna kost) i deo do nožnog palca. Značajan deo tereta nosi prednja noga, a najveći pritisak je smešten u centralnom delu pete.

Medijalni (unutrašnji) deo preuzima veoma mali deo tereta zbog koštane strukture stopala [4] i ne predstavlja nam veliki značaj za ispitivanje. Na slici 3 nalazi se raspored pritiska stopala prilikom hoda, gde crvena boja predstavlja deo gde je pritisak najveći.

Postoji veliki izbor FSR senzora i potrebno je pronaći odgovarajući. U radu koji je objavila Juliette van der Pas na sajtu Arduino Project Hub [5], koriste se FSR 406, koji je postavljen na peti i FSR 402, koji je postavljen na spoljašnji deo stopala i deo do prstiju (slika 4).



Sl. 3. *Raspodela pritiska stopala prilikom hoda [4]*



Sl. 4. *Uložak u kojem su postavljena 3 FSR senzora [5]*

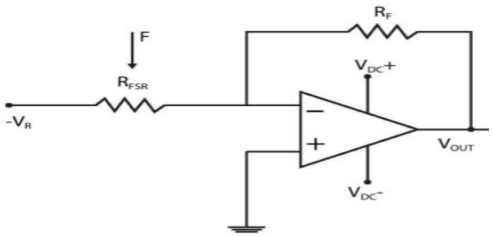
3. SLANJE REZULTATA MERENJA I PRINCIP OBRAĐIVANJA REZULTATA

Kako rezultati merenja i očitavanja nisu moguća na samim FSR senzorima, svi krajevi senzora se dovode na poseban pretvarački blok.

U pomenutom bloku se sile pretvaraju u napone i mikrokontroler pomoću svojih analognih ulaza očitava napone. Pretvarački blokovi, koji su malopre spomenuti, služe kako bi rešili problem koji se javlja kod ovih FSR senzora.

Utvrđeno je da otpornost ne opada linearno sa povećanjem sile i taj problem se rešava pomoću šeme na slici 5. Kolo se sastoji od operacionog pojačavača na koji smo priključili FSR i dovodimo napon. Delovanjem sile na senzor, otpornost senzora se smanjuje i struja teče kroz kolo [6].

Na izlazu se dobija napon V_{out} koji se šalje dalje mikrokontroleru. Na slici 6 je prikazan grafik gde se jasno uočava koliko se otpornost linearno smanjuje sa povećanjem sile [7].

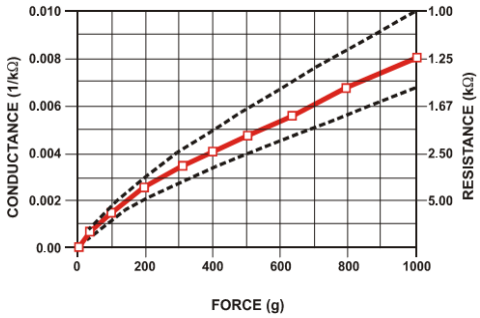


Sl. 5. R_{FSR} povezan sa operacionim pojačavačem [6]

$$I = \frac{V_r}{R_{FSR}} = V_r \cdot \frac{1}{R_{FSR}} = V_r \cdot G_{FSR} = k \cdot F \qquad (1)$$

$$V_{out} = R_F \cdot I = R_F \cdot k \cdot F = k_2 \cdot F \qquad (2)$$

Jednačina (2) jasno ukazuje da se izlazni napon povećava srazmerno sili pritiska.



Sl. 6. *Grafik uticaja sile na linearno smanjenje ili povećanje*

4. UTICAJ I4.0 TEHNOLOGIJE

Današnja tehnologija daje mogućnost da pomoću aplikacije na mobilnom telefonu korisnik upravlja uređajima u svom domu ili da nadgleda proces rada. Pomenuta tehnologija nosi naziv **Internet of Things** i služi da "opameti" hardver i promeni sve sfere našeg života. [8].

Upravo FSR senzori mere sile kojima dete deluje na uloške, ti podaci se šalju mikrokontroleru i pomoću Bluetooth mreže šalju podatke na mobilni telefon koje dete mora imati uz sebe.

Pri pokretanju aplikacije postavlja se pitanje: „Kolika je masa deteta?". Nakon unosa mase, aplikacija bira opseg koji je pogodan za tu masu i taj podatak se preko Bluetooth-a šalje mikrokontroleru. Zbog izuzetno velike količine podataka (Big Data) koje senzori prikupljaju u realnom vremenu, oni se skladište i obrađuju na *cloudu.* Pošto se deca brzo razvijaju, aplikacija na svakih sedam dana postavlja pitanje o trenutnoj masi deteta kako bi se opseg prilagodio novoj masi.

U aplikaciji postoji opcija gde se nalaze grafički prikazi koliko je pravilnih i nepravilnih koraka dete napravilo u toku jednog dana na svakih sat vremena (sl. 7) i ukupan broj pravilnih i nepravilnih koraka za nedelju dana (sl. 8).

Sl. 7. *Ukupan broj pravilnih i nepravilnih koraka na svakih sat vremena*

Sl. 8. *Grafički prikaz pravilnih i nepravilnih koraka za 7 dana*

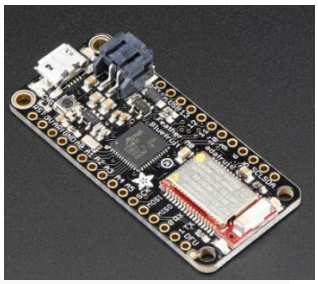
Merenje počinje onog momenta kada senzor detektuje prvi pritisak stopala na uložak i kada se pokrene aplikacija. Na slici 7 nalazi se ukupan broj pravilnih i nepravilnih koraka deteta na svakih sat vremena. Merenje je počelo u 6:15, pa su konačni rezultati pravilnih i nepravilnih koraka je predstavljeni u 7:15. Prednost ove aplikacije je mogućnost da roditelji i lekar sa svog mobilnog telefona mogu prijaviti na nalog deteta. Na taj način je omogućen detaljan uvid u napredak ili nazadovanje.
U aplikaciji postoji i posebna opcija gde se beleže isključivo nepravilni koraci. Cilj je da se dobiju podaci na kom delu stopala je najveće odstupanje prilikom hoda. Prikazana je Tabela 1 kada je dete napravilo nepravilan korak. Tabela se sastoji od tri parametra:

1. izmerene sile na stopalu
2. vremena kada se nepravilan korak desio
3. trenutne mase deteta.

Odstupanje	Peta	Spoljni deo	Deo do prstiju	Vreme [s]	Masa [kg]
Sila [N]	317	367	387	13:23:12	40

Tabela 1. Prikaz rezultata detetovog hoda u aplikaciji

Crvenom bojom je obeleženo na kom delu stopala je dete ostvarilo odstupanje, kao i vreme kada se desilo.
Prikupljanje podataka sa senzora i slanje tih podataka pomoću Bluetooth mreže, omogućeno je odgovarajućim mikrokontrolerom. Za potrebe Pametnog uloška koristi se Adafruit Feather M0 Bluefruit LE mikrokontroler. Reč je o najnovijem Bluetooth Low Energy protokolu niske potrošnje u frekventnom opsegu od 2.4 GHz. To je jedini bežični protokol koji može da se koristi sa iOS operativnim sistemom, a podržavaju ga i svi moderni pametni telefoni.. Korisniku je omogućeno da ima potpunu kontrolu nad uređajem, uključujući mogućnost definisanja i upravljanja karakteristikama uređaja [9].



Sl. 9. Adafruit Feather M0 Bluefruit LE mikrokontroler [9]

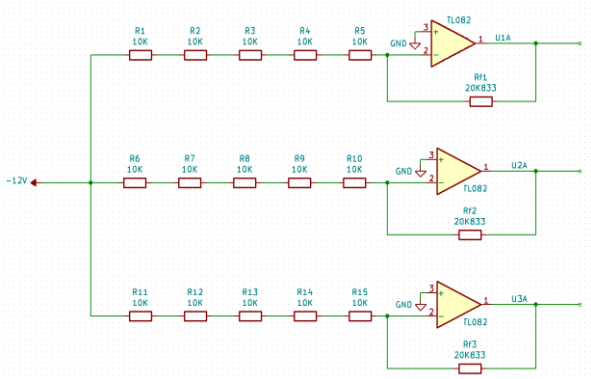
Sve prethodno pomenuto nema smisla ukoliko ne bi postojao način kako da se dete primora da hoda pravilno. Roditelj treba da instalira aplikaciju i pomoću bluetooth - a uspostavi komunikaciju sa uloškom. Tada u sklopu aplikacije postoji opcija da se snimi glasovna poruka takvog sadržaja da skrene detetu pažnju da ne hoda pravilno. Poruka bi se automatski aktivirala posle svaka 2 nepravilna koraka.
Jedan od problema može da bude činjenica da dete ne želi da pristane da nosi uložak. Ukoliko dete 7 dana ne reaguje na upozorenja koja dobija i svesno pravi greške, tada je roditelj u obavezi da kontaktira lekara i da ga obavesti o nastaloj situaciji i da se detetu dodeli termin kada će morati pod nadzorom pedijatara da radi vežbe radi unapređenja hoda.

5. PROOF OF CONCEPT

Na samom startu eksperimenta došlo je problema oko nabavke FSR senzora. Prvo rešenje problema je da se umesto FSR senzora postave otpornici od 10 KΩ. U dogovoru sa mentorom Marjanom Urekarom, pet redno vezanih otpornika od 10 KΩ (ukupno 50 KΩ u jednoj grani) bi bilo dovoljno za masu do 50 kg.
Eksperiment je podeljen u 5 segmenata:

1. U prvom merenju korišćen je otpornik od 10 KΩ za masu od 10 Kg,
2. U drugom merenju korišćena su dva otpornika od 10 KΩ (ukupno 20 KΩ) za masu od 20 Kg,
3. U trećem merenju korišćena su tri otpornika od 10 KΩ (ukupno 30 KΩ) za masu od 30 Kg,
4. U četvrtom merenju korišćena su četiri otpornika od 10 KΩ (ukupno 40 KΩ) za masu od 40 Kg,
5. U petom merenju korišćeno je pet otpornika od 10 KΩ (ukupno 50 KΩ) za masu od 50 Kg.

Merenje je bilo podeljeno u pet segmenata kako bi se izvršile simulacije za različite mase deteta. Na slici 10 nalazi se električna šema kada je potrebno merenje za 50 kg. Tri grane sa otpornicima menjaju 3 FSR senzora koji bi bili postavljeni u idealnom slučaju (slika 10).



Sl. 10. Električna šema kada je potrebno merenje za 50 kg

Povezivanje je ostvareno na ploči *Analog System Lab Kit PRO*. Na ulaz ploče dovodi se napon od -12 V, dok se na izlazu invertujućeg operacionog pojačavača očekuje napon od 5 V. Očitavanje napona na izlazu je omogućeno pomoću univerzalnog mernog instrumenta.
Kako bi se dobio željeni napon na izlazu, potrebno je podesiti otpornost na R_F otporniku. U tabeli 2 predstavljeni su rezultati očekivanih vrednosti R_F otpornosti, kao i dobijene vrednosti prilikom povezivanja. Otpornost R_F je podešena na dekadnim kutijama otpornosti.

R_{FSR} [KΩ]	$R_{F\ OČEKIVANO}$ [KΩ]	$R_{F\ DOBIJENO}$ [KΩ]
10	4.066	4.016
20	8.333	8.053
30	12.500	11.840
40	16.667	15.240
50	20.833	19.250

Tabela 2. Prikaz očekivane vrednosti R_F otpornika i dobijene vrednosti R_F otpornika

6. ZAKLJUČAK

Ovaj rad nudi rešenje kako da se unapredi hod deteta i kako podstaći dete da to realizuje. Internet of Things, kao deo Industrije 4.0 nudi mogućnost slanja rezultata i monitoring. Za tu potrebu koriste se FSR senzori, mikrokontroler i Bluetooth mreža koja šalje rezultate na aplikaciju mobilnog telefona. Prednost „Biomedicinskog uloška za cipele“ u odnosu na prethodne uređaje jeste mogućnost da se bluetooth sistem nalazi u samom ulošku. Glavni razlog jeste da se spreči uništenje wireless senzorskog sistema ukoliko dete slučajno udari nogom o neki predmet.

Svakako je dodatna prednost i aplikacija na mobilnom telefonu koja detaljno prikazuje napredak hoda. Postoji prostora za napredak, pre svega odabir FSR senzora koji ima opseg sila za masu do 100 kg. Kao moguće rešenje može da se upotrebi *100 kg Thin Film Pressure Sensor for Arduino*. Jedan od problema svakako jeste „Sa koliko godina dete može da nosi uložak?“. U novinskom članku „Gait abnormalities in children“ [10] postoji objašnjenje da prve naznake bolesti stopala nastaju već u trećoj godini života.

U Scholastic novinskom članku [11] se nalazi objašnjenje da dete postaje svesno tek u sedmoj godini. Najveći problem nastaje sa decom između treće i sedme godine, kako im prilagoditi uložak i kako ih naterati da slušaju upozorenja roditelja. Audio poruka sa glasom roditelja je početna ideja kako dete motivisati da se trudi i da koristi Pametan uložak.

Kako bi mikrokontroler mogao da prepozna koji korak je ispravan, potrebne su referentne vrednosti. Treba imati u vidu da ne mogu sva deca imati podjednaku masu, pa su samim tim i referentne vrednosti drugačije. Zato je potrebno da se sprovede ispitivanje dece u rasponu od treće do osamnaeste godine, a da su pritom i različitih konstitucija.

7. LITERATURA

[1] https://www.centarsm.co.rs/saveti-primaknuto-stopalo.html.

[2] https://solarbotics.com/product/50803/.

[3] N. Igić, Kineziološka analiza hoda, Novi Sad: Medicinski fakultet Novi Sad, Katedra za medicinsku rehabilitaciju, Kineziologija, 2015.

[4] J. Pauk, K. Daunoraviciene, M. Ihnatouski, J. Griskevicius and J. V. Raso, "Analysis of the plantar pressure distribution in children with foot deformities," Januar 2010..

[5] J. v. d. Pas, "A DIY Smart Insole to Check Your Pressure Distribution," Arduino Project Hub, 2018.

[6] V. Aleksovski, Force resistive sensor, Novi Sad: Fakultet tehničkih nauka, 2014.

[7] www.sparkfun.com/datasheets/Sensors/Pressure/fsrguide.pdf.

[8] P. Sovilj, Internet of Things, Novi Sad: Fakultet tehničkih nauka.

[9] https://www.adafruit.com/product/2995.

[10] D. H. Willacy, "Gait abnormalities in children," *Patient,* 2019..

[11] P. D. Dorfman, "Age of reason," *Scholastic*

Kratka biografija

Aleksandar Kostovski rođen je u Novom Sadu 1994. godine. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu, na katedri za električna merenja 2020. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 007.5
DOI: https://doi.org/10.24867/12BE51Popadic

PAMETNO BROJILO U KONCEPTU INDUSTRIJE 4.0

SMART ELECTRICITY METER IN INDUSTRY 4.0

Igor Popadić, *Fakultet tehničkih nauka, Novi Sad*

Oblast-ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je dat opis realizacije rješenja povezivanja digitalnog brojila sa web serverom. Dat je opis načina rada digitalnog brojila DIRIS A-10, njegove mogućnosti, a takođe i opis realizacije web servera, prvo povezivanjem tastera koji simulira rad brojila, a na kraju i povezivanjem brojila.*

Ključne riječi: *Digitalno brojilo, IoT, Merenja*

Abstract – *This paper describes the realization of the solution of connecting a digital meter to web server. A description of how the DIRIS A-10 digital meter works, it's capabilities, and also a description of the realization of a web server, first by connecting a button that simulates the operation of the meter, and finally by connecting the meter.*

Keywords: *Digital electricity meter, IoT, Measurements*

1. UVOD

Pojava koncepta IoT i sve brži razvoj interneta uslovio je i sve veći razvoj pametnih uređaja koji se mogu povezati na internet i kojima možemo pristupiti pomoću našeg telefona. Sa ovim promjenama došlo je i do pojave pametnog brojila za električnu energiju. Pomoću ovoga brojila korisnik može da vidi koliko je struje potrošio, kad je najviše troši i da shodno tome kontroliše svoju potrošnju i svoje potrebe.

Sa ovim obična digitalna brojila koliko god bila bolja u odnosu na stara, robusna analogna brojila, ipak polako gube na vrijednosti. Elektrodistribucija ide ka tome da ne šalje ljude na teren, već da na web serveru vide potrošnju svakog korisnika.

Tu se javlja problem pravovremenog prikaza podataka na serveru. Vodeći se ovim problemom i željom da korisnicima digtalnog brojila na vizuelno ljepši način prikažemo potrošnju električne energije, kroz ovaj rad je dat primjer pravljenja web servera i povezivanja digitalnog brojila na njega.

2. OPIS PROBLEMA

Problem sa kojim se susrećemo u ovome projektu je što digitalno brojilo DIRIS A-10 ima samo analogni izlaz.
Slanjem analognog signala ne može se prikazati broj koji pokazuje potrošnju električne energije u datom vremenskom periodu.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Marjan Urekar.

Primarni cilj projekta je naći način kako da se takvo brojilo poveže na web server gdje će u realnom vremenu biti prikazana potrošnja električne energije. I tu dolazi do problema šta uraditi sa tim signalom i kako na osnovu njega na ekranu računara pokazati ono što se nalazi na ekranu digitalnog brojila.

3. DIRIS A-10

DIRIS A-10 je izabrano trofazno digitalno brojilo za ovaj rad. To je modularno višenamjensko brojilo za mjerenje električnih vrijednosti u niskonaponskim mrežama. Omogućava prikaz svih električnih parametara, kao i otkrivanje varijacija u temperaturi zahvaljujući svojoj unutrašnjoj funkciji mjerenja temperature.

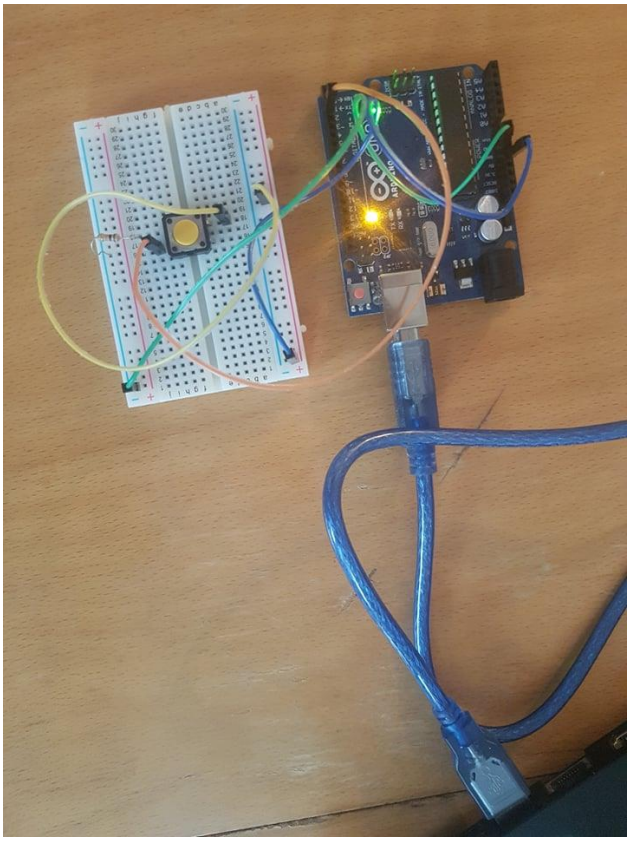
Slika 1. *Izgled digitalnog brojila DIRIS A-10*

Komunikaciju vrši preko RS485 protokola. Na slici 1. je prikazan izgled prednjeg panela, gdje možemo primjetiti 5 tastera pomoću kojih se mijenja da li će na ekranu biti prikazana struja, napon, temperatura, aktivna i reaktivna snaga.

3. HARDVER

Kao što je naglašeno postoji problem kako analogni signal dovesti do računara. Zbog nemogućnosti pristupa laboratoriji i brojilu, odlučeno je brojilo zamijeniti sa tasterom. Analogni signal koji se pošalje prilikom pritiska tastera simulira rad brojila i analogni signal koji bi brojilo slalo. Odlučeno je da se taster postavi na protobord i da se poveže sa arduinom pa onda sa računarom. Za realizaciju ovoga

kola bio je potreban jedan taster, otpornik od 220 Ω, kratkospojnici, protobord i Arduino Uno. Na slici 2. možemo da vidimo kako izgleda hardverski dio ovoga projekta, tj. kako izgleda kolo koje simulira rad brojila.



Slika 2. *Hardverski dio*

Arduino ima ulogu da prepozna kad je taster pritisnut i tu dolazimo do prelaska na softversko rješavanje problema.

4. SOFTVER

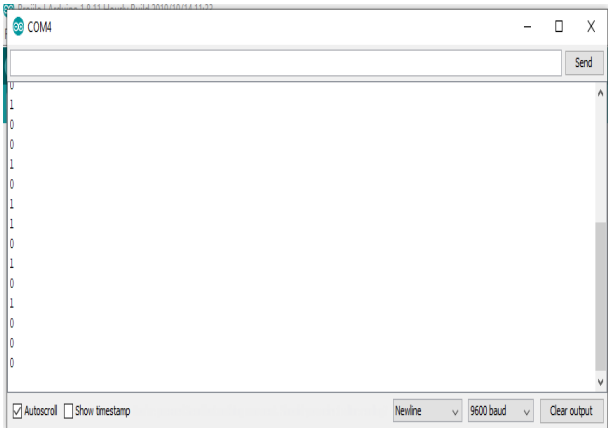
Prvi korak u rješavanju softverskog dijela problema ovoga projekta je bio napisati kod po kome će arduino da prepoznaje kada je taster pritisnut. Na slici 3. dat je prikaz koda koji to obavlja.

Brojilo

```
int switchButton = 2;
void setup() {
  Serial.begin(9600);
  pinMode(switchButton, INPUT);
}
void loop() {
  int buttonState = digitalRead(switchButton);
  if(buttonState == HIGH){
    Serial.println("1");
  }else{
    Serial.println("0");

  }
  delay(500);
}
```

Slika 3. *Arduino kod*

Prvi dio koda arduinu govori na kom je pinu povezan taster. Onda se u petlji provjerava da li je taster pritisnut, tj. da li je stanje tastera HIGH i ako jeste ispisuje se 1 u prozoru serijskog porta. Ako stanje tastera nije HIGH, tj. taster nije pritisnut ispisuje se 0. Na slici 4. možemo vidjeti kako izgleda ispis nula i jedinica.



Slika 4. *Ispis nula i jedinica u prozoru serijskog porta*

Sledeći korak u radu ovoga projekta je bio pravljenje web stranice na kojoj će se podaci prikazati. Tu se javlja problem i kako pokupiti nule i jednice što ih arduino šalje i onda kako ih iskoristiti. Stranica je napravljenja kao localhost stranica na adresi 3000. Zatim se pomoću funkcije serialport hvataju nule i jedinice koje šalje arduino koji je povezan u ovome slučaju na port 4 i te nule i jedinice se ispisuju u konzoli. Na slici 5. je dat prikaz koda.

index.js

c: > Users > Korisnik > Desktop > Brojilo > index.js > ...

```
var http = require('http');
var express = require('express');
var app = express();
var server = http.createServer(app);
var io = require('socket.io')(server);

const SerialPort = require('serialport');
const Readline = require('@serialport/parser-readline');
const port = new SerialPort("COM4", { baudRate: 9600 });
const parser = new Readline();
port.pipe(parser);

parser.on('data', line => console.log(`> ${line}`));
app.set('view engine', 'ejs');
app.use(express.static(__dirname + '/public'));
app.get('/', function (req, res) {
    res.render('index')
    });

    port.on('open', function(){
        console.log('serial port opened');
    });
    io.on('connection', function(socket){
        console.log('socket.io connection');
        parser.on('data', function(data){
            data = data.trim();
            console.log(data);
            socket.emit('data', data);
        });
});
server.listen(3000, function(){
    console.log('Listening on port 3000...');
});
```

Slika 5. *Kod za stranicu*

Nakon što je napravljena web stranica sledeći problem koji se pojavio je kako prikazati potrošnju. Da bi to riješili napravljena su dva prozora na web stranici. U prvom se ispisuje broj impulsa i taj broj se povećava za 1 sa svakim novim pritiskom tastera.

U drugom prozoru se ispisuje potrošnja električne energije u imp/Ws. U kodu možemo vidjeti da se konstanta množi sa impulsima.

Na stranicu je još dodat i grafik koji se iscrtava u real time-u sa svakim povećanjem potrošnje. Na slici 6. i 7. je dat prikaz tog koda.

```
<html>
    <head>
        <title>Get data</title>
        <div id="chart"></div>
    </head>
    <body style="background-color:lightgrey">
        <h1>Arduino data</h1>
        <p>The button is <span id="button-state"></span></p>
        <input type="text" id="in" value="0"/>  <p>imp</p>
        <input type="text" id="inc" value="0"/> <p>imp/WS</p>
        <script src="/socket.io/socket.io.js"></script>
        <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
      <script>
          var socket = io();
            var i = 0 ;
            var buttonState=document.getElementById('button-state');
            socket.on("data", function(data){
                if(data === "1" && i>= 0){
                    buttonState.innerHTML = "pressed";
                    i += 1;
                }else{
                    buttonState.innerHTML = "not pressed";
                }
                document.getElementById('in').value = i;
            });
```

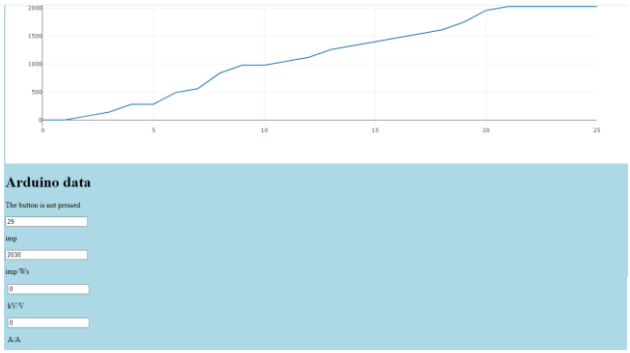
Slika 6. *Prvi dio koda*

```
            var k = 0 ;
              var buttonState=document.getElementById('button-state');
              socket.on("data", function(data){
               if(data === "1" && i>= 0){
                buttonState.innerHTML = "pressed";
                k = i*70;
                }else{
                  buttonState.innerHTML = "not pressed";
                  }
                    document.getElementById('inc').value = k;
        });

            function getData() {
               return k;
                  }

                  Plotly.plot('chart',[{
                        y :[getData()],
                           type:'line'
                        }]);

                        setInterval(function() {
                        Plotly.extendTraces('chart', { y: [[getData()]] }, [0])
                                                        }, 2000);

    </script>
  </body>
</html>
```

Slika 7. *Drugi dio koda*

5. TESTIRANJE SISTEMA

Kada je kod pokrenut i taster pritisnut vidjeli smo da kolo obavlja posao. Na slici 8. dat je prikaz web stranice.

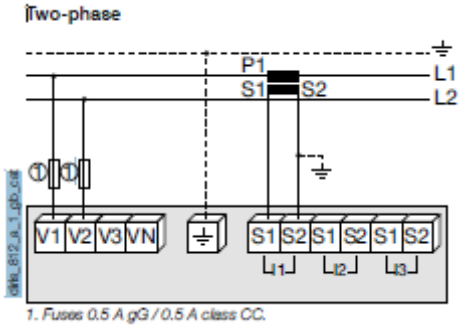


Slika 8. *Prikaz izgleda web stranice*

Vidimo kako izleda iscrtan grafik i vidimo kako izgleda prikaz energije i impulsa. Y osa grafika pokazuje potrošnju električne energije koja je izražena u imp/Ws. Grafik se konstantno ispisuje, a krivulja raste sa svakim novim pritiskom tastera. Sledeći korak je povezivanje brojila umjesto tastera.

6. ZAKLJUČAK

Na slici 9. vidimo primjer dvofaznog načina povezivanja brojilo.



Slika 9. *Primjer dvofaznog brojila*

Realizovani sistem u kome se brojilo povezuje na web server nudi mogućnost korisniku da isprati potrošnju električne energije u realnom vremenu. Grafički prikaz podataka, pored prikaza potrošnje i impulsa u brojevima, nudi mogućnost realnog prikaza kako potrošnja raste i da prema tome korisnik preuzme određene akcije povodom smanjenja potrošnje.

Sa realizacijom sajta i hardverskog rješenja, ograničenosti povezivanja digitalnog brojila koje ima izlaz samo za impulse u pogledu povezivanja na internet su otklonjene. Brojilo sada može da se iskoristi u konceptu Industrije 4.0. Sistem odlikuje stabilan rad.

Dodatkom prozora za konstantu naponskih i strujnih transformatora omogućeno je da se provjeri rad brojila pomoću sajta.

Sledeći korak bi bio dodavanje dodatnih opcija na sajt kao što su prikaz struje, napona i temperature, tj. svih opcija koje prikazuje brojilo na svom ekranu. Sve te veličine bi bile prikazane grafički. Sajt bi se mogao unaprijediti tako da mu se može pristupiti i sa telefona. Tada sajt ne bi bio korišten samo za brojilo Diris A-10, već svaki korisnik može personalizovati sajt i koristiti ga za više brojila i vidjeti potrošnju za svaki uređaj u domu. Takođe bi se moglo dodati par dodatnih senzora na samo brojilo da se unaprijede njegove perfomanse.

7. LITERATURA

[1] Indira Knight, “Connecting Arduino to the Web” , 2018

[2] Cuno Pfister, „Getting started with internet of things“, *O’Reilly,* 2011.

[3] Peter Waher, „Learning Internet of Things“, *Packt publishing*, 2015.

[4] Paul Horowitz, Winfield Hill, „The Art of Electronics“, 2015

Kratka biografija

Igor Popadić rođen je u Trebinju 1996. godine. Diplomirao je na Fakultetu tehničkih nauka u Novom Sadu, na katedri za električna mjerenja 2019. godine. Master rad iz oblasti mjerno-informacionih sistema, odbranio je 2020. godine.

Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE53Lovcevic

VIZUALIZACIJA RADA AKCELEROMETRA POMOĆU QT I ANDROID APLIKACIJE UPOTREBOM KLIJENT-SERVER ARHITEKTURE

ACCELEROMETER VISUALIZATION USING QT AND ANDROID APPLICATION AND CLIENT-SERVER ARCHITECTURE

Milan Lovčević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U okviru ovog rada prikazane su realizacije Qt i Android aplikacija koje omogućavaju prikupljanje podataka merenja akcelerometra ADXL345. Opisani su i način skladištenja podataka u SQLite bazu podataka i vizualizacija u realnom vremenu, primenom klijent-server arhitekture.*

Ključne reči: *Akcelerometar, Raspberry Pi, Qt, Android, klijent-server arhitektura*

Abstract – *Within this paper, the realizations of Qt and Android applications are presented, which enable the collection of measurement data of the ADXL345 accelerometer. The method of data storage in SQLite database and real-time visualization, using client-server architecture, are also described.*

Keywords: *Accelerometer, Raspberry Pi, Qt, Android, client-server architecture.*

1. UVOD

Jedna od reči koja se poslednjih par godina sve intenzivnije koristi u indrustriji jeste termin *IoT*, što predstavlja skraćenicu od „*Internet of Things*" ili kako se na našem jeziku prevodi Internet stvari ili Internet integrisanih uređaja.

Ovaj pojam se odnosi na mrežu fizičkih uređaja, vozila, zgrada i drugih predmete koji u sebi sadrže ugrađen softver, senzor i pristup internetu preko čega ti objekti mogu da prikupljaju i razmenjuju podatke. *IoT* često uključuje obradu velikih količina podataka prikupljenih radi dobijanja korisnih informacija, koje mogu dosta da pomognu prilikom donošenja nekih strateških odluka.

Na primer, u nekim zemljama su postavljeni merači količine padavina tokom cele godine, zatim se prikupljeni podaci analiziraju i koriste kako bi zemlja bolje reagovala prilikom velikih padavina ili poplava [1].

U daljem tekstu biće dat jedan od primera upotrebe *IoT* tehnologije, korišćenjem ADXL345 senzora, zatim TCP/IP protokola zasnovanim na krijent-server arhitekturi, zaduženim za razmenu podataka putem interneta, kao i Qt i *Android Studio* aplikacija, pomoću kojih je kreiran grafički korisnički interfejs.

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

2. OPIS PODSISTEMA UREĐAJA

Komponenete koje su korišćene u realizaciji ovog master rada su Raspberry Pi 3 model B+ (skraćeno RPi) i ADXL345 model akcelerometra. *Android Studio* i *Qt Creator* su korišćeni za realizaciju server aplikacije vidljive korisniku i na kraju, za razmenu podataka između Raspberry Pi i jedne od server aplikacija korišćena je klijent-server arhitektura.

Akcelerometar je komponenta koja se koristi za merenje vibracija ili ubrzanja kretanja objekata. Svoju primenu je pronašao u mnogim granama industrije, kao što je automobilska industrija (npr. pri sudaru, za aktiviranje vazdušnih jastuka), robotika (npr. za određivanje pozicije objekta), u avio industriji, itd. Akcelerometar koji je korišćen za realizaciju ovog projekta je ADXL345, koji predstavlja akcelerometar male snage i visoke rezolucije (13-bita). Poseduje mogućnost merenja statičkog ubrzanja (gravitaciona sila), kao i dinamičkog ubrzanje, koje predstavlja rezultat kretanja ili vibriranja. Zbog svoje visoke rezolucije ima mogućnost merenja promene nagiba manjeg od 1.0° [2, 3].

Qt predstavlja besplatni softver za kreiranje grafičkog korisničkog interfejsa kao i višeplatformskih aplikacija koje rade na različitim softverskim i hardverskim platformama kao što je Windows, macOS, Android ili na embeded sistemima, sa malo ili bez izmena u osnovi koda i bez ograničavanja funkcionalnosti ili brzine aplikacije. Signali i slotovi predstavljaju jednu od najvažniji Qt karakteristika koji omogućavaju komunikaciju između Qt objekata [4, 5].

Android Studio predstavlja razvojno okruženje koje je namenjeno za razvoj Android mobilnih aplikacija, zasnovano je na *IntelliJ IDEA* softveru. Glavne komponente koje se koriste u razvoju Android aplikacije su aktivnosti i servisi [5, 6].

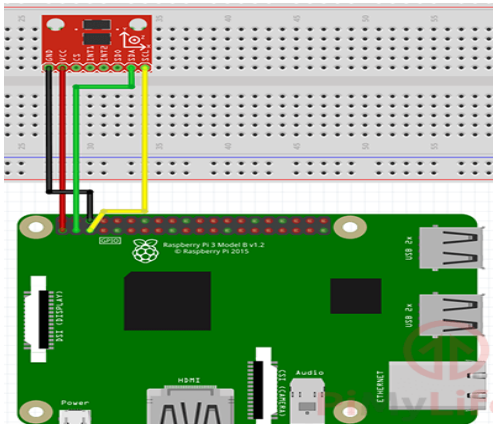
Klijent-server arhitektura predstavlja komunikaciju između dve aplikacije koje se nalaze na odvojenim uređajima . Aplikacija koja se zove klijent ima zadatak da inicira komunikaciju sa drugom aplikacijom koja se zove server. Dok server ne mora da zna nikakvu informaciju o klijentu, klijent mora da poseduje osnovne informacije o serveru, kao što je IP adresa i port na koji može da se poveže. Da bi se uspostavila komunikacija, potrebno je da server i klijent kreiraju sopstveni *socket,* nakon uspostavljanje komunikacije moguća je međusobna razmena podataka (*full duplex*). Velika prednost klijent-server arhitekture je što dozvoljava razmenu podataka sa

velike udaljenosti, pošto akcelerometar ne mora fizički biti povezan sa uređajem na kojem je server aplikacija implementirana, odnosno podaci se prenose putem interneta ili lokalne mreže [4].

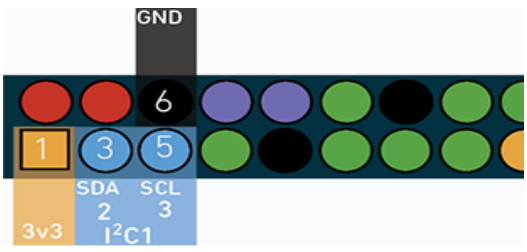
Kao što je bilo rečeno, implementacija projekta je zasnovana na klijent-server arhitekturi koja razdvaja rad na dve celine. Prvi deo predstavlja povezivanje Raspberry Pi mikroračunara sa odgovarajućim akcelereometrom i kreiranje aplikacije u Python-u, koja će predstavljati klijenta. Drugi deo predstavlja kreiranje Qt i Android server aplikacije, čija će funkcija biti da nakon što klijent inicira komunikaciju počne sa prikupljanjem podataka poslatih od klijenta, zatim da ih vizualizuje (grafički prikaz) i skladišti u bazu podataka. Za skladištenje podataka u bazu korišćena je *SQLite* biblioteka.

3. REALIZACIJA RPI KLIJENT APLIKACIJE

Klijent aplikacija je realizovan u Python-u i nalazi se na Raspberry Pi uredjaju koji je direktno povezan sa akcelerometrom pomoću GPIO pinova, kao što je prikazano na slici 1. Komunikacija između RPi i akcelerometra je sihrona serijska komunikacija zasnovana na I2C (eng. *Inter-Integrated Circuit*) serijskom interfejsu, koji predstavlja jedan od najčešće korišćenih načina komunikacije između uređaja na kratkim razdaljinama (rastojanje treba da bude manje od 3 metra). Da bi se ostvario ovaj vid komunikacije potrebne su nam pored napona i uzemljenja samo još dve žice, čiji položaj na RPi ploči je dat na slici 2.



Slika 1. *RPI i akcelerometar povezivanje [7]*



Slika 2. *Prikaz korišćenih pinova [7]*

Kako bi klijent uspeo da ostvari komunikaciju sa serverom potrebno je da zna IP adresu servera i port na koji može da se poveže. U Python skripti to je definisano upotrebom dve promenjive „HOST“ i „PORT“, gde se promenjivoj „HOST“ dodeljuje IP adresa servera, a promenjivoj „PORT” broj porta servera na koji klijent planira da se poveže. Sa druge strane, potrebno je takođe pripremiti komunikaciju sa akcelerometrom, za tu svrhu koristimo „*busio*” i „*board*” biblioteke. Prva biblioteka u sebi sadrži mnoštvo drugih biblioteka koje omogućavaju rukovanje sa različitim serijskim protokolima. Prilikom realizacije ovog projketa ona je iskorišćena kako bi se ostvarila komunikacija sa akcelerometrom putem I2C serijskog interfejsa. Druga biblioteka je potrebna kako bi se na lakši način odredili položaj SDA i SCL pinova na RPi ploči.

Instanciranje različitih klasa, priprema pinova, definisanje porta i IP adrese je potrebno uraditi samo jednom, na početku izvršavanja programa. Nakon uspešnog povezivanja sa serverom, klijent prvo čita podatke sa senzora, pakuje ih i šalje serveru. Ovaj proces je postavljen u beskonačnu petlju, sa pauzom od jedne sekunde između slanja dva uzastopna podatka. Da bismo prestali sa slanjem podataka potrebno je prekinuti konekciju sa serverom.

4. IMPLEMENTACIJA BAZE PODATAKA

Baza koja je zadužena za skladištenje podataka primljenih od klijenta je realizovana upotrebom aplikacionog formata podataka koji se zove *SQlite.* Identična implementacija baze podataka, upotrebom *SQlite* biblioteke je urađena za potrebe obe server aplikacije. Podaci koji se čuvaju u bazi su *x*, *y* i *z* kordinate, koje server dobija do klijenta, kao i vremenski trenutak i datum kada je ta vrednost zabeležena. Primer baze podataka je dat na slici 3.

X_POSITION	Y_POSITION	Z_POSITION	DATE	TIME
5.0602314	7.1784677999...	5.766310199...	09.08.2020	10:56:18
5.0210048	7.1784677999...	5.6486304	09.08.2020	10:56:19
5.0602314	7.1784677999...	5.6094038	09.08.2020	10:56:20
5.0210048	7.2176943999...	5.687856999...	09.08.2020	10:56:21
5.0602314	7.1392411999...	5.727083599...	09.08.2020	10:56:22

Slika 3. *Baza podataka*

Da bi se kreirala baza potrebno je navesti putanju gde će se ta baza čuvati, tip baze podataka, naziv tabele i nazive kolona koje treba da se nalaze u tabeli. Baza podataka se kreira samo u slučaju ako na navedenoj putanji ne postoji baza sa identičnim imenom, ukoliko postoji, proces kreiranja se preskače. Prilikom izlaska iz aplikacije sama baza podataka kao i podaci koji se nalaze u njoj se ne brišu, tako da prilikom narednog pokretanja aplikacije, proces kreiranja baze podataka se preskače i nastavlje se upis podataka u tabelu bez gubitka prethodno unetih podataka.

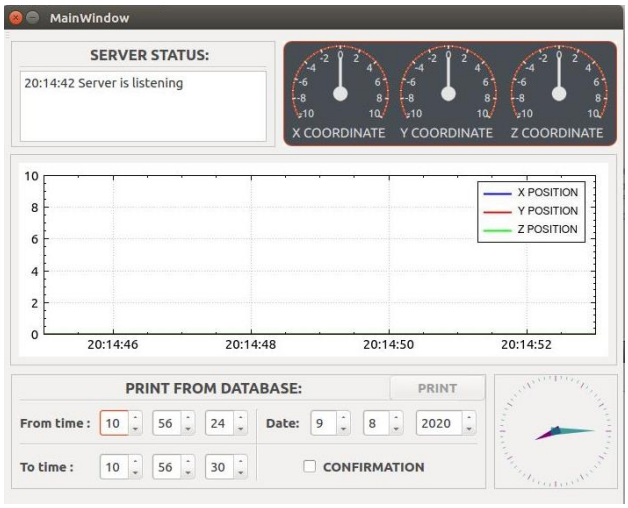
5. REALIZACIJA QT SERVER APLIKACIJE

Server koji je realizovan za potrebe Qt aplikacije pripada onom delu funkcionalnih blokova čije se izvršavanje dešava u pozadini aplikacije, odnosno nije vidljivo korisniku. Jedine informacije koje korisnik može da dobije u vezi servera su prikazane u polju za ispis teksa (eng. *textbox*), koji se nalazi u gornjem levom uglu aplikacije i to su samo osnovne informacije o trenutnom

statusu servera i klijenta. U polju za ispis teksta moguće je dobiti tri različite poruke, uz sve tri poruke ispisuje se i vreme kada se određena akcija desila:

- *„Server is listening“*, koja označava da je server aktivan i da čeka na klijenta.
- *„Client connected“*, klijent se konektovao i počinje sa slanjem podataka.
- *„Client disconnected“*, komunikacija između servera i klijenta je prekinuta.

Qt sever je realizovan upotrebom *QTcpServer* klase, koja obezbeđuje osnovne TCP server funkcionalnosti. Prilikom pokretanja aplikacije server se automatski aktivira i čeka da se neki klijent poveže, što takođe potvrđuje i poruka u polju za ispis teksta. Slika 4 predstavlja izgled aplikacije realizovanog u Qt-u.



Slika 4. *Izgled Qt aplikacije*

Vizualizacija rada akceleromtera u realnom vremenu je postignuta upotrebom *„QcustomPlot”* biblioteke. Glavni deo programa zauzima grafik koji služi za vizuelni prikaz podataka u realnom vremenu koje je server primio od klijenta. Grafik se sastoji od dve ose *x* i *y*, *y* osa je iskorišćena za ispis vremenskog trenutka, u 24-oro časovnom režimu, u kojem su pristigle vrednosti zabeležene, dok *x* osa se koristi za prikaz vrednosti koje je klijent poslao. Početni opseg x ose je od 0 do 10 ali on može dinamički da se menja. Npr. ako klijent pošalje podatak čija je vrednost manja od 0 ili veća od 10, grafik će proširiti svoj opseg kako bi mogao da prikaže i tu vrednost.

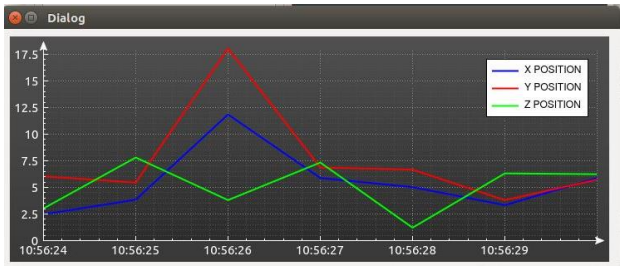
Kazaljke u gornjem desnog uglu su realizovane upotrebom QML jezika. QML je deklarativni jezik dizajniran da opiše korisnički interfejs programa, odnosno kako korisnički interfejs treba da izgleda i kako treba da se ponaša.

Pošto klijent svake sekunde šalje po tri podatka serveru, u gornjem desnom uglu programa nalaze se tri kazaljke, gde svaka od njih treba da prikaže po jednu od te tri vrednosti.

Opseg sve tri kazaljke je od -10 do 10, i ne može dinamički da se menja, odnosno ako klijent pošalje vrednosti koja je veća od 10, kazaljka će pokazati na vrednost 10 ili u slučaju da je potrebno prikazati vrednosti manju od -10 kazaljka će biti postavljena na vrednosti -10.

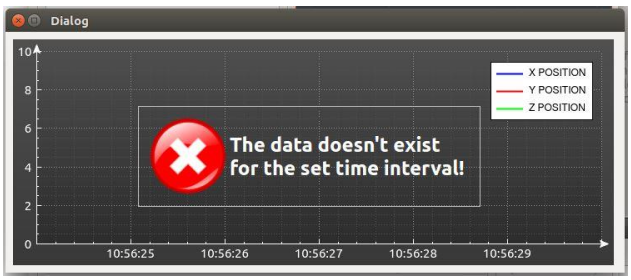
U donjem desnom uglu aplikacije nalazi se sat, koji je preuzet iz primera koji se dobijaju uz Qt instalaciju. Njegov doprinos ovoj aplikaciji je samo estecki.

Pošto se podaci koje je klijent poslao čuvaju u bazu podataka u tabelarnoj formi, koja često može da bude prilično naporna prilikom analize merenja akcelerometra, u samu aplikaciju ugrađen je i mehanizam koji nam omogućuva da podatke pročitamo iz baze za neki zadati vremenski interval i da ih iscrtamo na grafiku, kako bismo olakšali proces analize. Mogućnost iscrtavanja podataka iz baze je ostvarena pomoću opcija koje zauzimaju donji deo aplikacije. Kako bismo grafički prikazali podatke iz baze podataka za neki željeni interval vremena prvo je potrebno da unesemo početno i krajnje vreme, odnosno od kojeg do kojeg trenutka želimo da prikažemo podatke i datum. Nakon što se unese željeni vremenski interval i označi polje za potvrdu (eng. *checkbox)* dobija se prozor kao na slici 5.



Slika 5. *Grafički prikaz podataka za zadati vremenski interval*

Postoji i druga mogućnost, da je korisnik pravilno uneo vremenski interval ali da za taj interval podaci ne postoje u bazi podataka, u tom slučaju otvoriće se novi prozor sa grafikom ali umesto iscrtanih podataka biće ispisana poruka da podaci ne postoje za zadati vremenski interval, kao što je prikazano na slici 6.

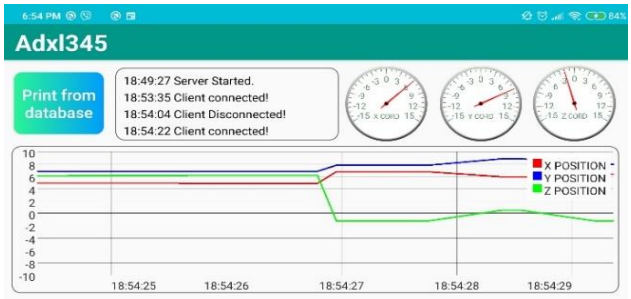


Slika 6. *Poruka koja se ispisuje kada podaci ne postoje za zadati vremenski interval*

6. REALIZACIJA ANDROID APLIKACIJE

Kao što je bio slučaj i kod prethodno objašnjene server aplikacije i ova aplikacija ima polje za ispis teksta u kojem se ispisuju samo osnovne informacija o trenutnom stanju klijenta i servera. Prilikom pokretanja aplikacije server se automatski aktivira i čeka da se neki klijent poveže, što takođe potvrđuje i poruka u polju za ispis teksta. Pošto postoji mogućnost da se poruke u polju za ispis teksa nagomilaju i da neke budu slučajno prepisane, kako bi se izbeglo to i kako bi korisnik u svakom trenutku imao pristup svim informacijama koje su bile ispisane u polju od trenutka pokretanja aplikacije, ugrađen je mehanizam pomeranja na gore i dole (eng. *scroll up and*

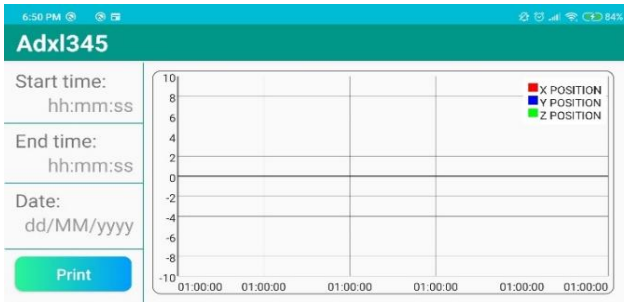
down) kroz polje. Na slici 7 je prikazan izgled server aplikacije realizovane u *Android Studio* programu.



Slika 7. *Izgled Android aplikacije*

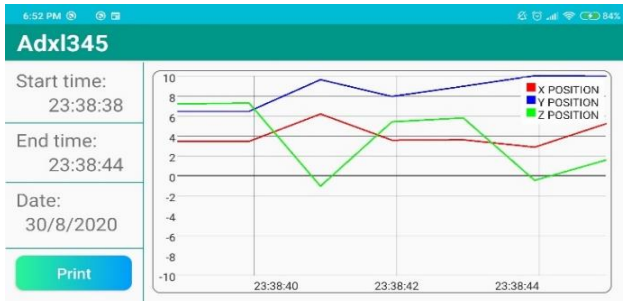
Najveći deo aplikacije zauzima grafik koji u realnom vremenu ispisuje podatke dobijene od klijenta. Za razliku od Qt aplikacije, početni opseg ovog grafika je od -10 do 10 i može dinamički da se proširi ako postoji potreba za tim. Gornji deo prozora aplikacije uglavnom ima istu funkciju kao i u aplikaciji realizovanoj u Qt-u. Tri kazaljke prikazuju vrednosti dobijene od klijenta za sve tri kordinate, a polje za ispis teksta ima zadatak da nas informiše o trenutnom statusu servera i klijenta.

Najveća razlika između ove aplikacije i one realizovane u Qt-u je u dugmetu „*Print from database*" koje kada se pritisne otvara identičan prozor kao na slici 8.



Slika 8. *Prozor za ispis podataka iz baze podataka*

Kako bi se iscrtali podaci za željeni interval vremena potrebno je uneti početno, krajnje vreme i datum, odnosno potrebno je pritisnuti na svako od tri polja koja se nalaze na levoj strani prozora. Nakon što su svi neophodni podaci uneseni potrebno je pritisnuti dugme „*Print*". Ukoliko je korisnik pravilno uneo parametre i podaci postoje u bazi za zadati interval vremena, oni će se iscrtati na grafiku kao što je prikazano na slici 9. U slučaju da podaci ne postoje u bazi grafik će ostati prazan.



Slika 9. *Ispis podataka za zadati vremenski interval*

Najveća prednosti ove aplikacije u odnosu na aplikaciju realizovanu u Qt-u je mogućnost zumiranja (eng. *Zoom*) i pomeranja levo i desno kroz grafik, što u velikoj meri olakšava analizu podataka za veći vremenski interval.

7. ZAKLJUČAK

U prethodnim poglavljima objašnjena je jedna od mogućih upotreba *IoT* tehnologije, koja ima za zadatak da vrednosti merenja senzora vizualizuje i skladišti u bazu podataka. Za prikupljanje podataka iskorišćen je ADXL345 model akcelerometra koji je povezan sa Raspberry Pi mikroračunarom, koji zajedno predstavljaju jednu stranu prilikom razmene podataka. Drugu stranu čini uređaj koji poseduje instaliranu Android ili Qt server aplikaciju, koje su realizovane upotrebom C++, Java, QML i XML programskog jezika.

Komunikacija između mikroračunara i aplikacija postignuta je upotrebom TCP/IP protokola koji je zasnovan na klijent-server arhitekturi. Mogućnosti za proširenje opsega funkcionalnosti i unapređenja projekta su velike. Projekat dalje može da se razvija u smeru upotrebe više od jednog klijenta istovremeno ili upotrebom različitih senzora.

8. LITERATURA

[1] W.-M. Lee, „Introduction to IoT Using the Raspberry Pi", Code magazine. https://www.codemag.com/article/1607071/Introduction-to-IoT-Using-the-Raspberry-Pi (pristupljeno u oktobru 2020.)

[2] Analog Devices, „Digital Accelerometer". https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf (pristupljeno u avgustu 2020.)

[3] Automatika, „Akcelerometri". https://www.automatika.rs/baza-znanja/senzori/akcelerometri.html (pristupljeno u avgustu 2020.)

[4] I. Mezei, „Računarska elektronika (deo skripti)", Fakultet tehničkih nauka, Novi Sad, 2018. https://www.elektronika.ftn.uns.ac.rs/ (pristupljeno u septembru 2020.)

[5] https://www.rt-rk.uns.ac.rs/sites/default/files/materijali/lab/Vezba02%20-%20Upoznavanje%20sa%20Android%20Studio%20okru%C5%BEenjem.pdf (pristupljeno u avgustu 2020.)

[6] http://vtsnis.edu.rs/wp-content/plugins/vts-predmeti/uploads/MOS%20Predavanje%205%202018.pdf (pristupljeno u julu 2020.)

[7] Emmet, „Raspberry Pi Accelerometer using the ADXL345", Pi My Life Up, 24 May 2019. https://pimylifeup.com/raspberry-pi-accelerometer-adxl345/ (pristupljeno u septembru 2020.)

Kratka biografija:

Milan Lovčević rođen je u Sremskoj Mitrovici 1995. god. Završio je osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu, na Departmanu za energetiku, elektroniku i telekomunikacije, 2019 godine. Master rad iz oblasti Embeded sistemi i algoritmi odbranio je 2020.godine.

U realizaciji Zbornika radova Fakulteta tehničkih nauka u toku 2020. godine učestvovali su sledeći recenzenti:

Aco Antić
Aleksandar Erdeljan
Aleksandar Kovačević
Aleksandar Kupusinac
Aleksandar Ristić
Bato Kamberović
Biljana Njegovan
Bogdan Kuzmanović
Bojan Batinić
Bojan Lalić
Bojan Tepavčević
Bojana Beronja
Branislav Atlagić
Branislav Nerandžić
Branka Nakomčić
Branko Milosavljević
Branko Škorić
Damir Đaković
Danijela Ćirić
Danijela Gračanin
Danijela Lalić
Darko Čapko
Darko Marčetić
Darko Reba
Dejan Ecet
Dejan Jerkan
Dejan Ubavin
Dejana Nedučin
Dragan Ivanović
Dragan Jovanović
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
Igor Peško
Ivan Beker
Igor Maraš
Ivan Mezei
Ivan Todorović
Ivana Katić
Ivana Kovačić
Ivana Maraš
Ivana Miškeljin
Jasmina Dražić
Jelena Atanacković
Jeličić
Jelena Borocki
Jelena Demko Rihter
Jelena Radonić
Jelena Slivka
Jelena Spajić
Jovan Petrović
Jovanka Pantović
Laslo Nađ
Lazar Kovačević
Leposava Grubić Nešić
Livija Cvetičanin
Ljiljana Vukajlov
Ljiljana Cvetković
Ljubica Duđak
Maja Turk Sekulić
Marinko Maslarić
Marko Marković
Marko Todorov
Marko Vekić
Maša Bukurov
Matija Stipić
Milan Čeliković
Milan Mirković
Milan Rapajić
Milan Segedinac
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
Miroslava Radeka
Mirko Borisov
Miro Govedarica
Miroslav Hajduković
Miroslav Kljajić
Miroslav Popović
Miroslav Zarić
Mitar Jocanović
Mitar Đogo
Mladen Kovačević
Mladen Tomić
Mladen Radišić
Nebojša Brkljač
Neda Milić Keresteš
Nemanja Stanisavljević
Nemanja Sremčev
Nikola Đurić
Nikola Jorgovanović
Nikola Radaković
Ninoslav Zuber
Ognjen Lužanin
Pavel Kovač
Peđa Atanasković
Petar Malešev
Platon Sovilj
Predrag Šiđanin
Radivoje Dinulović
Radomir Kojić
Radovan Štulić
Relja Strezoski
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 Bačkalić
Svetlana Nikoličić
Tanja Kočetov
Tatjana Lončar - Turukalo
Uroš Nedeljković
Valentina Basarić
Velimir Čongradec
Veran Vasić
Veselin Perović
Višnja Žugić
Vladimir Katić
Vladimir Mučenski
Vladimir Strezoski
Vlado Delić
Vlastimir Radonjanin
Vojin Ilić
Vuk Bogdanović
Zdravko Tešić
Zoran Anišić
Zoran Brujić
Zoran Čepić
Zoran Jeličić
Zoran Mitrović
Zoran Papić
Željen Trpovski
Željko Jakšić