



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



ЗБОРНИК РАДОВА ФАКУЛТЕТА ТЕХНИЧКИХ НАУКА

Едиција: Техничке науке - зборници

Година: XXXI

Број: 13/2016

Нови Сад

Едиција: „Техничке науке – Зборници“

Година: XXXI

Свеска: 13

Издавач: Факултет техничких наука Нови Сад

Главни и одговорни уредник: проф. др Раде Дорословачки, декан Факултета техничких Наука у Новом Саду

Уредништво:

Проф. др Раде Дорословачки

Проф. др Драгиша Вилотић

Проф. др Срђан Колаковић

Проф. др Владимир Катић

Проф. др Драган Шешилија

Проф. др Миодраг Хаџистевић

Проф. др Растислав Шостаков

Доц. др Мирослав Кљајић

Доц. др Бојан Лалић

Доц. др Дејан Убавин

Проф. др Никола Јорговановић

Доц. др Борис Думнић

Проф. др Дарко Реба

Проф. др Ђорђе Лађиновић

Проф. др Драган Јовановић

Проф. др Мила Стојаковић

Проф. др Драган Спасић

Проф. др Драгољуб Новаковић

Редакција:

Проф. др Владимир Катић, главни
уредник

Проф. др Жељен Трповски, технички
уредник

Проф. др Драган Шешилија

Проф. др Драгољуб Новаковић

Др Иван Пинћур

Бисерка Милетић

Језичка редакција:

Бисерка Милетић, лектор

Софија Рацков, коректор

Марина Катић, преводилац

Издавачки савет:

Савет за библиотечку и издавачку делатност ФТН,
проф. др Радош Радивојевић, 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)

Двомесечно

ISSN 0350-428X

COBISS.SR-ID 58627591

ПРЕДГОВОР

Поштовани читаоци,

Пред вама је тринаеста 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 дипломских-мастер 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 дипломских-мастер радова студената ФТН-а у облику кратког рада у „Зборнику радова Факултета техничких наука“. Поред студената дипломских-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 веб сајту ФТН-а (www.ftn.uns.ac.rs) и штампаном, који је пред вама. Обе верзије публикују се више пута годишње у оквиру промоције дипломираних инжењера-мастера.

У овом броју штампани су радови студената мастер студија, сада већ мастера, који су радове бранили у периоду од 28.09.2016. до 27.10.2016. год., а који се промовишу 07.04.2017. 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.

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 13., објављени су радови из области:

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

У свесци са редним бројем 14. објављени су радови из области:

- саобраћаја,
- графичког инжењерства и дизајна,
- архитектуре,
- инжењерског менаџмента,
- инжењерства заштите животне средине,
- математике у техници,
- геодезије и геоматике,
- управљања ризиком од катастрофалних догађаја и пожара,
- инжењерства информационих система и
- сценске архитектуре и дизајна.

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

- | | |
|---|-----------|
| 1. Ljubiša Trivković, Sebastian Baloš,
PROJEKTOVANJE TEHNOLOGIJE ZAVARIVANJA I ISPITIVANJE ZAVARENIH
SPOJEVA KONSTRUKCIONOG I NERĐAJUJEG ČELIKA | 2451-2454 |
| 2. Dušan Pejović,
BIOGAS POSTROJENJE ZA MALO SEOSKO POLJOPRIVREDNO DOMAĆINSTVO..... | 2455-2458 |

Radovi iz oblasti: Elektrotehnika i računarstvo

- | | |
|---|-----------|
| 1. Dejan Varga, Dejan Nemec,
PRIMENA GLOBALNOG SISTEMA ZA POZICIONIRANJE (GPS) U POLJOPRIVREDI | 2459-2462 |
| 2. Vladimir Šobić, Branko Milosavljević, Gordana Milosavljević,
INTEGRACIJA SA ELEKTRONSKIM SERVISIMA MINISTARSTVA FINANSIJA | 2463-2466 |
| 3. Stefan Predojević, Dragan Milićević,
ANALIZA POGONSKIH ASPEKATA RADA POSTROJENJA NA ČVRSTU BIOMASU | 2467-2470 |
| 4. Andrea Erdeljan,
IMPLEMENTACIJA MEMORIJSKOG BUILT-IN SELF TEST KONTOLERA | 2471-2474 |
| 5. Veljko Boljanović,
PRIMENA KOMPRIMUJUĆEG ODABIRANJA ZA DETEKCIJU KORISNIKA U
BEŽIČNIM KOMUNIKACIJAMA | 2475-2478 |
| 6. Vladimir Vrbaški,
IP JEZGRO ZA OBRADU PIKSELA I NJEGOVA IMPLEMENTACIJA NA
ZEDBOARD PLATFORMI | 2479-2482 |
| 7. Aleksandar Tankosić,
IZGRADNJA I ZAŠTITA DISTRIBUTIVNE 110/20 KV I 20/0,4 KV
TRANSFORMATORSKIH STANICA | 2483-2486 |
| 8. Miloš Dobrosavljević,
REALIZACIJA PROGRAMSKE PODRŠKE ZA KOMUNIKACIJU IZMEĐU DSP DEMO
PLATFORME I PERSONALNOG RAČUNARA | 2487-2490 |
| 9. Viktor Udovički, Dragan Milićević,
ANALIZA ISPITNE STANICE ZA TESTIRANJE CEVI PRITISKOM..... | 2491-2494 |

	STRANA
10. Davor Rukavina, KVANTIFIKACIJA SIGNALA MR PERFUZIJE	2495-2498
11. Vladimir Kosjer, Željko Trpovski, Dejan Nemec, PROJEKTOVANJE I ANALIZA BAZNIH STANICA U LTE MREŽAMA	2499-2503
12. Siniša Gak, Dejan Vukobratović, ALGORITAM ZA PRONALAZENJE MINIMALNIH BLOKIRAJUĆIH SKUPOVA ZAŠTITNIH KODOVA SA RETKIM MATRICAMA PROVERE PARNOSTI	2504-2506
13. Dejan Križan, DIZAJN I VERIFIKACIJA CSI-3 IP MODULA	2507-2510
14. Душица Алексић, МОДУЛАРНА WEB АПЛИКАЦИЈА НА .NET ПЛАТФОРМИ	2511-2514
15. Nedeljko Stojaković, MODELOVANJE DISTRIBUTIVNIH ELEKTROENERGETSKIH SISTEMA UPOTREBOM MICROSOFT GRAPH ENGINE KOMPONENTE	2515-2518
16. Душко Алексић, IONIC ОКВИР ЗА РАЗВОЈ ХИБРИДНИХ МОБИЛНИХ АПЛИКАЦИЈА	2519-2522
17. Aleksandar Kerekeš, Veran Vasić, Đura Oros, ZAŠTITA VISOKONAPONSKIH ASINHRONIH MOTORA	2523-2526
18. Kristian Haška, Nikola Vojnović, PRORAČUN SIMETRIČNIH TOKOVA SNAGA DISTRIBUTIVNIH MREŽA PRIMENOM POSTUPKA SUMIRANJA STRUJA I KOREKCIJA NAPONA	2527-2530
19. Aleksandar Svorcan, VIŠEKRITERIJUMSKA PODELA MODELA PODATAKA DISTRIBUTIVNE MREŽE	2531-2534
20. Novak Boškov, SISTEM ZA PERZISTENCIJU PODATAKA U PROGRAMSKOM JEZIKU <i>CLOJURE</i>	2535-2538
21. Žolt Varga, ANDROID АПЛИКАЦИЈА ЗА АКВИЗИЦИЈУ SIGNALA VIBRACIJA I NAKNADNA OBRADA U SVRHU DETEKTOVANJA KVARA MOTORA	2539-2542
22. Aleksandra Aleksić, Milan Vidaković, SIMULACIJA SAOBRAĆAJA UPOTREBOM AGENTSKE TEHNOLOGIJE I SARL PROGRAMSKOG JEZIKA	2543-2546
23. Jelena Keravica, Milan Vidaković, SERVISNO-ORIJENTISANI DIZAJN U RUBY PROGRAMSKOM JEZIKU	2547-2550
24. Marina Bulat, PRIMENA MONTE KARLO METODE ZA ODREĐIVANJE MERNE NESIGURNOSTI	2551-2554
25. Marko Milić, PODELA GRAFOVA ALGORITMOM SUPERKORENA	2555-2558
26. Miloš Nedić, RAČUNARSKI MERNI SISTEM ZASNOVAN NA PIC MIKROKONTROLERU I ETHERNET KOMUNIKACIJI	2559-2562
27. Filip Fesl, IMPLEMENTACIJA IMPORTA ŠEME RELACIONE BAZE PODATAKA U OKVIRU KROKI ALATA	2563-2566
28. Mirjana Laušev, IMPLEMENTACIJA ALGORITMA SINTEZE KORIŠĆENJEM PRISTUPA RAZVOJU SOFTVERA ZASNOVANOM NA MODELIMA	2567-2570
29. Viktor Šujić, DSL I GENERATOR KODA ZA SPECIFIKACIJU KOMUNIKACIJE IZMEĐU SERVERA I HARDVERSKIH UREĐAJA	2571-2574
30. Nemanja Blagojević, ПРОШИРЕЊЕ КРОКИ АЛАТА НОВИМ ТИПОВИМА ФОРМИ	2575-2578

	STRANA
31. Bojana Zoranić, GENAN – DSL ZA OPIS KORISNIČKOG INTERFEJSA WEB APLIKACIJA UPOTREBOM TEXTX METAJEZIKA	2579-2582
32. Ivan Jašo, ANDROID APLIKACIJA ZA PLAĆANJE STUDENTSKIH TROŠKOVA POSREDSTVOM PAYPAL SISTEMA	2583-2586
33. Petar Gostović, Vladimir Rajs, AUTOMATIZACIJA MOTOCIKLA I UPRAVLJANJE PUTEM ANDROID PAMETNOG UREĐAJA	2587-2590
34. Lazar Nikolić, PROŠIRIVA PLATFORMA ZA GENERISANJE APLIKACIJA IZ TEKSTUALNIH DSL MODELA	2591-2595
35. Горана Гојић, CUDA БАЗИРАНА ПАРАЛЕЛИЗАЦИЈА МНОЖЕЊА СИМЕТРИЧНИХ МАТРИЦА И ВЕКТОРА	2596-2599
36. Šándor Klebečko, IMPLEMENTACIJA KORISNIČKOG INTERFEJSA FENEK ALATA	2600-2603
37. Saša Krajina, AUTOMATSKO KONFIGURISANJE DISTRIBUIRANOG DMS SCADA SISTEMA	2604-2607

Radovi iz oblasti: Građevinarstvo

1. Snežana Utvić, MATEMATIČKO MODELIRANJE OBARANJA NIVOVA PODZEMNE VODE NA LOKACIJI NOVE GLAVNE CRPNE STANICE U NOVOM SADU (NGC1) PRIMENOM GMS MODELA	2608-2611
2. Aleksandar Narandžić, MATEMATIČKO MODELIRANJE PLAVLJENJA KOD RAČINOVACA I RAJEVOG SELA PRIMENOM 2D MODELA U HEC-RAS-u	2612-2615
3. Milan Marinković, Bojan Matić, RAZVOJ GEOGRAFSKOG INFORMACIONOG SISTEMA ZA SAOBRAĆAJNICE LOKALNIH SAMOUPRAVA	2616-2619
4. Slavko Tomić, PROCENA STANJA I ENERGETSKA SANACIJA STAMBENE ZGRADE U ULICI ALEKSE ŠANTIĆA U NOVOM SADU	2620-2623
5. Jelena Bačić, PROCENA STANJA, SANACIJA I POBOLJŠANJE ENERGETSKE EFIKASNOSTI ZGRADE GIMNAZIJE „SVETOZAR MARKOVIĆ“ U NOVOM SADU	2624-2627
6. Miloš Tišma, Matija Stipić, HIDRAULIČKA ANALIZA I FUNKCIONISANJE VAKUUMSKE KANALIZACIJE NA PRIMERU NASELJA GOSPOĐINCI	2628-2631
7. Aleksandar Dušanić, PROJEKAT KONSTRUKCIJE STAMBENE ZGRADE P+4+PK PREMA EVROKODU I ANALIZA EFEKATA TEORIJE DRUGOG REDA	2632-2635
8. Suzana Juškić, UPRAVLJANJE TROŠKOVIMA GRAĐENJA ABC METODOM	2636-2639
9. Aleksandra Lajko, NUMERIČKI MODEL ZA PROVERU SIGURNOSTI ODBRAMBENIH NASIPA NA SUFOZIJU I FLUIDIZACIJU	2640-2643
10. Biljana Bugarinović, HIDRAULIČKA ANALIZA PRIMENE RACIONALNE METODE I SWMM-a NA ODVOĐENJE ATMOSFERSKIH VODA NASELJA ĐURĐEVO	2644-2647

	STRANA
11. Viktor Keresteš, Matija Stipić, HIDRAULIČKA ANALIZA RAZVOJA KANALIZACIONE MREŽE NASELJA VRBAS I SLIVA CENTRALNOG POSTROJENJA ZA PRERADU OTPADNIH VODA	2648-2651
12. Никола Рајић, Ђорђе Лађиновић, НЕЛИНЕАРНА СТАТИЧКА СЕИЗМИЧКА АНАЛИЗА ВИШЕСПРАТНИХ АБ ОКВИРА	2652-2655

Radovi iz oblasti: Mehatronika

1. Ivan Sušić, KUĆNA AUTOMATIZACIJA ZASNOVANA NA RASPBRY PI	2656-2659
--	-----------

PROJEKTOVANJE TEHNOLOGIJE ZAVARIVANJA I ISPITIVANJE ZAVARENIH SPOJEVA KONSTRUKCIONOG I NERĐAJUJEG ČELIKA**WELDING TECHNOLOGY DESIGN AND TESTING OF STRUCTURAL AND STAINLESS STEEL WELDS**Ljubiša Trivković, Sebastian Baloš, *Fakultet tehničkih nauka, Novi Sad***Oblast - MAŠINSTVO**

Kratak sadržaj – Pojavom novih i usavršavanjem postojećih postupaka zavarivanja dolazi se do potrebe da se i kontrola i ispitivanje zavarenih spojeva unaprijedi. Ispitivanje zavarenih spojeva metodama koje uključuju razaranje je vrlo primjenljiv i efikasan način ispitivanja. Ispitivanje sa razaranjem podrazumjeva makro ispitivanje, ispitivanje zatezanjem, savijanjem, energije udara i tvrdoće. U radu je prikazan jedan primjer ispitivanja sa razaranjem sučeonog spoja dobijenog MAG i MIG postupcima zavarivanja.

Abstract – Inventing new and improving existing methods of welding leads to the need to improvements in weld testing as well. Testing of welds by methods involving the destruction is very applicable and efficient way of testing. Destructive testing comprises of macro testing, tensile, bend, impact energy and hardness testing. In this work, an example of destructive testing of a weld obtained with MAG and MIG processes is shown.

Ključne reči: zavarivanje, postupci zavarivanja, klasifikacija zavarenih spojeva.

1. UVOD

Zavarivanje i srodne tehnike (lemljenje, lijepljenje, navarivanje i metalizacija) su proizvodne tehnologije, pri kojima se uložnom energijom spajaju dva ili više materijala u neraskidivu vezu, odnosno, nanosi se sloj istorodnog ili raznorodnog materijala na osnovni materijal. S toga se tehnike spajanja materijala uvrštavaju u visokotehnološke (eng. *high-tech*) procese koji su važni u skoro svim granama industrije.

Danas skoro da se ne može zamisliti niti jedna proizvodnja gotovih proizvoda koja bi svoju djelatnost lako obavljala bez tehnike i tehnologije zavarivanja. Pored klasičnog mašinstva ove tehnike su prisutne i u građevini, energetici, elektrotehnici, procesnoj tehnici, livarstvu, automobilskoj, vojnoj i avio industriji, brodogradnji, mehatronici, kao i u preciznoj, ukrasnoj i zubnoj tehnici [1].

NAPOMENA:

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

1.2 Postupci zavarivanja

Postupci zavarivanja koji se najčešće koriste su:

- Gasno zavarivanje
- Ručno elektrolučno zavarivanje
- MIG zavarivanje
- MAG zavarivanje
- TIG zavarivanje
- Elektrootporno zavarivanje

2. EKSPERIMENTALNI RAD**2.1. Zavarivanje uzoraka**

U eksperimentalnom radu su zavarena dva ispitna uzorka, od dva različita materijala. U oba slučaja je riječ o čeliku (konstrukcioni i nerđajući), kako bi se posmatralo njihovo ponašanje u toku zavarivanja i uporedili rezultati mehaničkih ispitivanja. Dimenzije ploča su 150x350 mm. Mehaničke osobine i hemijski sastav ovih čelika su dati u sledećim tabelama:

Tabela 1. *Mehaničke osobine čelika S355JR*

R_{eH} [MPa]	R_m [MPa]	A [%]	KV [J]
≥ 355	510 – 680	≥ 18	≥ 27 (20°C)

Tabela 2. *Hemijski sastav čelika S355JR, najmanje vrednosti*

C	Si	Mn	P	S	Ni
0,24	0,55	1,6	0,035	0,035	0,012

Tabela 3. *Mehaničke osobine čelika X5CrNi18-10*

R_{p02} [MPa]	R_m [MPa]	A [%]	KV [J]
≥ 355	510 – 680	> 35	≥ 27 (20°C)

Tabela 4. *Hemijski sastav čelika X5CrNi18-10*

C	Si	Mn	P	S	Cr	Ni
<0,07	<1	<2	<0,045	<0,015	17,0–19,5	8–10,5

2.1.1 Zavarivanje konstrukcionog čelika

Izbor aparata za zavarivanje u ovom slučaju se sveo na aparat marke FRONIUS TPS 2700 (MIG/MAG). Kao zaštitni gas istopljenog metala je upotrebljena mješavina FEROMIX C18- 82%-18%:Ar-CO₂ (stand. SRPS EN ISO 14175: M21), proizvođača MESSER. Za dodatni materijal je izabrana žica prečnika ϕ 1.2mm, G3Si1 (stand. SRPS EN ISO 14341-A: G42 4M G3Si1), proizvođača Elektroda Jesenice. Parametri zavarivanja su prikazani u tabeli 5.

Tabela 5. Parametri zavarivanja čelika S355JR

R.b.	Postupak zavarivanja	Dodatni materijal ϕ [mm]	Jačina struje [A]	Napon zavarivanja [V]
1.	135	1,2	125	16
2.	135	1,2	180	18,2
3.	135	1,2	200	23,6
4.	135	1,2	200	18,5
5.	135	1,2	200	22
R.b.	Vrsta toka/polaritet	Brzina zavarivanja [cm/min]	Brzina dod. žice [m/min]	Unos energije [kJ/cm]
1.	DC+	17,5	3,2	5,5
2.	DC+	23,8	5,5	6,6
3.	DC+	20	6,7	11,3
4.	DC+	29,9	5,6	5,9
5.	DC+	28	6,4	7,5

2.1.2 Zavarivanje nerđajućeg čelika

Za zavarivanje ovog čelika je korišten isti aparat *FRONIUS TPS 2700*. Razlika od predhodnog zavarivanja se ogleda u zaštitnom gasu. Na aparat je povezana boca sa mješavinom gasa INOXLINE C2-97%-2.5%: Ar-CO₂ (stand. SRPS EN ISO 14175: M12), proizvođača *MESSER*. Korišćena žica prečnika ϕ 1.2mm, ER 316 Lsi (SRPS EN ISO 14343-A: G19 12 3Lsi), je od istog proizvođača kao i predhodna, *Elektroda Jesenice*. Parametri zavarivanja su prikazani u tabeli 6.

Tabela 6. Parametri zavarivanja čelika X5CrNi18-10

R.b.	Postupak zavarivanja	Dodatni materijal ϕ [mm]	Jačina struje [A]	Napon zavarivanja [V]
1.	135	1.2	150	15
2.	135	1.2	200	21.3
3.	135	1.2	200	21.7
R.b.	Vrsta toka/polaritet	Brzina zavarivanja [cm/min]	Brzina dod. žice [m/min]	Unos energije [kJ/cm]
1.	DC+	24.6	4.3	4.4
2.	DC+	30	7	6.8
3.	DC+	24	7	8.7

2.2. Ispitivanje bez razaranja

Kako standard nalaže, potrebna je kontrola prije početka zavarivanja, u toku i na kraju zavarivanja. Pošto su ispoštovane predhodne dvije, ostalo je izvršiti kontrolu zavarenih uzoraka.

2.2.1 Vizuelna kontrola

Najjeftinija i najjednostavnija kontrola jeste VT (eng. Visual Testing). Izvodi se prema standardima:

- **SRPS EN ISO 6520-1**- Zavarivanje i srodni postupci. Klasifikacija geometrijskih nepravilnosti u metalnim materijalima. Deo 1: Zavarivanje topljenjem [3]
- **SRPS EN ISO 5817**- Zavarivanje – Spojevi zavareni topljenjem u čeliku, niklu, titanijumu i njihovim legurama (zavarivanje elektronskim i laserskim snopom isključeno) – Nivoi kvaliteta za nesavršenosti [4]

Dimenzionom kontrolom je prekontrolisano nadvišenje šava, kao i njegova širina, uređajem “*Kembridž-metar*”, Slika 2.



Slika 2. Kembridž-metar

2.2.2 Ispitivanje penetrantima

Pošto su male prsline nevidljive golim okom, odnosno ne mogu se otkriti vizuelnom kontrolom, na ispitne uzorke je nanijet tečni penetrant, Slika 3, proizvođača *MAGNAFLUX*, u cilju indikacije.



Slika 3. Set za ispitivanje tečnim penetrantima

2.2.3 Radiografsko ispitivanje

Kako bi bili sigurni da je šav homogen, odnosno da nema zapreminskih grešaka, ploče je potrebno ispitati jednom od zapreminskih metoda. U startu je isključeno ispitivanje magnetnim česticama zbog nerđajućeg čelika, koji je nemagnetičan. U ovom slučaju, kontrola po kontruri je izvršena radiografijom (RT). Nakon zračenja u zaštićenoj laboratoriji su dobijeni radiografski filmovi.

2.3. Ispitivanje sa razaranjem

Ispitivanje uzoraka zatezanjem, savijanjem, energijom udara, makro ispitivanja i ispitivanja tvrdoće su izvršena u laboratoriji za mehanička ispitivanja na Institutu za varilstvo u Ljubljani, prema važećim standardima.

2.3.1 Makro ispitivanje zavora

Makro uzorci su nakon isjecanja i manualnog šmirglanja površine uzorka (uz hlađenje) na papirima različite granulacije (80, 120, 220, 320, 500, 800), tretirani sredstvom za nagrizanje *ADLER* (25ml destilovana voda, 50ml hlorovodonična kiselina, 15g gvožđe (III) hlorid, 3g amonijum tetrahlorosuprat II) i zatim posmatrani.

2.3.2 Ispitivanje zatezanjem

Uređaj na kome je vršeno ispitivanje je hidraulična kidalica *ZD40*, raspoloživog opterećenja 40 t. Epruvete se postavljaju u samostežuće čeljusti i na njih se djeluje silom, Slika 4. Temperatura laboratorije je iznosila 25 °C.



Slika 4. Hidraulična kraljica ZD40

2.3.3 Ispitivanje savijanjem

Pripremljeni uzorci za bočno savijanje su ispitani na istoj mašini kao i zatezanje, ZD40, Slika 5. Temperatura ispitivanja je iznosila 25°C.



Slika 5. Ispitivanje savijanjem

2.3.4 Ispitivanje energije udara

Za ovu vrstu ispitivanja, upotrebljen je uređaj, koji oslobađa rad od maksimalno 300J, Slika 6. Temperatura u laboratoriji je iznosila nepromjenljivih 25°C. Uzorci su po isjecanju potopljeni u alkohol i skladišteni u rashladni uređaj, kako bi bili ohlađeni na temperaturu od -20°C. Razlog toga jeste šire područje važnosti kvalifikacije zavarivanja.

2.3.5 Ispitivanje tvrdoće

Po standardu SRPS EN ISO 15614-1[2], tvrdoća se ne mjeri na materijalima grupe 8. Kako je nerđajući čelik X5CrNi18-10 iz te grupe, na njemu nije rađena Vickers metoda. Makro uzorak od konstrukcionog čelika je poslužio kao epruveta za mjerenje tvrdoće HV10.

3. REZULTATI I DISKUSIJA

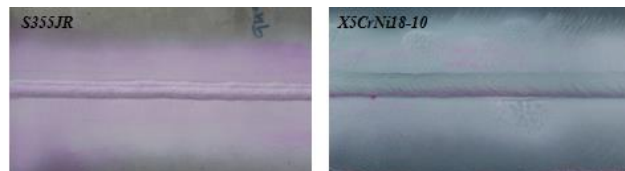
3.1 Rezultati vizuelne kontrole

Objekti ploče su površinski pregledane i nisu uočene greške i nepravilnosti koje su definisane standardom SRPS EN ISO 6520-1 [3]. Rezultati mjerenja su pokazali da je nadvišenje šava kod konstrukcionog čelika iznosilo 3 mm za širinu šava 17 mm, a kod nerđajućeg čelika 2 mm za

širinu šava 14 mm, što je prihvatljivo za nivo kvaliteta "B" po standardu SRPS EN ISO 5817[4].

3.2 Rezultati penetrantske kontrole

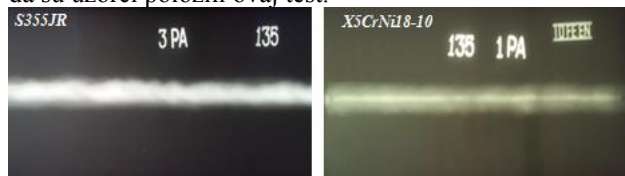
Nakon ispoštovanog postupka i redosleda nanošenja penetranta, na površini nije uočena indikacija, pa se smatra da su uzorci zavareni bez površinske greške, Slika 8.



Slika 8. Ploče ispitane penetrantima.

3.3 Rezultati radiografskog ispitivanja

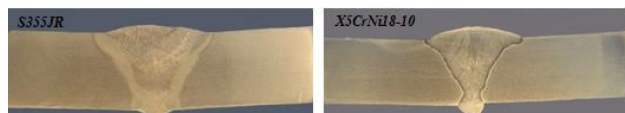
Razvijanjem i analiziranjem radiografskih filmova je dokazano da ne postoje zapreminske greške na ispitivanim pločama, Slika 9. Šav je homogen i smatra se da su uzorci položili ovaj test.



Slika 9. Radiografski snimci zavarenih uzoraka

3.4 Rezultati makroskopskog ispitivanja

Fotografije makro uzoraka su predstavljeni na slici 10. Kao što se vidi, ne postoje nepravilnosti i greške.



Slika 10. Makro uzorak: a) S355JR; b) X5CrNi18-10

3.5 Rezultati ispitivanja zatezanjem

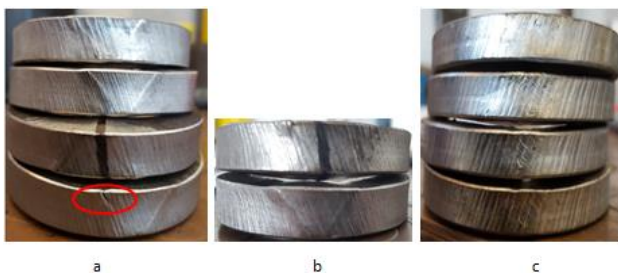
Prilikom ispitivanja zatezanjem, dobijeni rezultati su pokazali da i konstrukcioni i nerđajući čelik ispunjavaju zadate veličine, što se moglo i očekivati, jer se na makro uzorku nisu pojavile greške koje bi bile inicijator pada zatezne čvrstoće, Tabela 7.

3.6 Rezultati ispitivanja savijanjem

Pri savijanju nerđajućeg čelika nije došlo do pojave prslina, dok se kod konstrukcionog pojavila prslina na jednoj epruveti, pa stoga je ispitivan još jedan par epruveta koje zadovoljavaju po kvalitetu, Slika 11

Tabela 7. Rezultati ispitivanja zatezanjem

Materijal	S_0 [mm]	F_m [kN]	R_m [N/mm ²]	Izdužen je %	Mjesto preloma
S355JR	296.75	172	580	26	OM
	295.66	170.5	577	27.7	OM
X5CrNi 18-10	239.90	153.5	640	26	OM
	239.92	152	634	27.7	OM



Slika 11 Savijene epruvete:

a) uzorci čelika S355JR; b) ponovljeno ispitivanje čelika S355JR; c) uzorci čelika X5CrNi18-10

3.7 Rezultati ispitivanja energije udara

Razlike u energiji udara u zoni uticaja toplote i metalu šava su relativno male, pa se ovo ispitivanje smatra zadovoljavajuće. Tabela 8 prikazuje rezultate za čelik S355JR, a tabela 9 za čelik X5CrNi18-10.

Tabela 8. Rezultati ispitivanja žilavosti čelika S355JR

Oznaka	Širina uzorka [mm]	Visina uzorka [mm]	Površina uzorka [mm ²]	Energija udara KV ₂ [J/cm ²]	Položaj zarez
T1	10.032	7.970	79.95	78	VHT 1/1
T2	10.030	7.997	80.20	106	VHT 1/1
T3	10.030	7.995	80.19	117	VHT 1/1
Z1	10.025	7.929	79.49	82	VWT 0/1
Z2	10.023	8.001	80.19	90	VWT 0/1
Z3	10.020	7.925	79.43	108	VWT 0/1

Tabela 9. Rezultati ispitivanja žilavosti čelika X5CrNi18-10

Oznaka	Širina uzorka [mm]	Visina uzorka [mm]	Površina uzorka [mm ²]	Energija udara KV ₂ [J/cm ²]	Položaj zarez
T1	7.408	8.000	59.26	219	VHT 1/1
T2	7.448	7.968	59.35	256	VHT 1/1
T3	7.470	7.984	59.64	189	VHT 1/1
Z1	7.481	8.012	59.94	147	VWT 0/1
Z2	7.452	8.008	59.68	146	VWT 0/1
Z3	7.441	7.982	59.39	136	VWT 0/1

3.8 Ispitivanje tvrdoće

Da je izbjegnuto zakaljenje ZUT-a potvrđuje i izmjerena tvrdoća na konstrukcionom čeliku S355JR. Standard SRPS EN ISO 15614-1 [2] tačno definiše maksimalnu veličinu tvrdoće po Vickersu i ona iznosi 380HV. Kako je u ovom slučaju najveća izmjerena tvrdoća ≈300HV, jasno je da je u skladu sa zahtjevom, Tabela 10.

Tabela 10. Rezultati mjerenja tvrdoće

		Mjesto utiska		HV
OM	Bez uticaja toplote	1	OM površina	164,170,174
		2	OM sredina	161,162,166
		3	OM površina	162,167,168
		4	OM sredina	158,162,165
OM	Uticaj toplote	5	ZUT tjeme	267,292,260
		6	ZUT korijen	220,208,222
		7	ZUT tjeme	303,303,263
		8	ZUT korijen	226,205,199
ŠAV		11	Krovni sloj	208,203,222
		13	Koreni sloj	197,196,196

4. ZAKLJUČAK

Iz rezultata ispitivanja može se zaključiti:

- Ispitivanjem bez razaranja je dokazano da je šav bez površinskih, geometrijskih i zapreminskih greški.
- Ispitivanjem uzoraka na zatezanje su postignuti očekivani rezultati. Kod konstrukcionog čelika se lom javio u osnovnom materijalu a kod nerđajućeg u metalu šava, što je očekivano.
- Ispitivanjem uzoraka na savijanje je dokazano da nema pojave prslina.
- Ispitivanjem tvrdoće kod konstrukcionog čelika je pokazano da je tvrdoća po površinskom presjeku zavarenog spoja vrlo slična i ispod standardom propisanih vrednosti.
- Ispitivanjem energije udara su dobijeni rezultati koji ne pokazuju veliko odstupanje žilavosti u ZUT-u i šavu.

5. LITERATURA

- [1] Janez Tušek – „Varjenje in sorodne tehnike spajanja materialov v neločljivo zvezo”, Ljubljana, 2014 g.
- [2] SRPS EN ISO 15614-1-Ispitivanje procedure zavarivanja (čelika, nikla i legura nikla)
- [3] SRPS EN ISO 6520-1- Zavarivanje i srodni postupci . Klasifikacija geometrijskih nepravilnosti u metalnim materijalima . Deo 1: Zavarivanje topljenjem
- [4] SRPS EN ISO 5817- Zavarivanje – Spojevi zavareni topljenjem u čeliku, niklu, titanijumu i njihovim legurama (zavarivanje elektronskim i laserskim snopom isključeno) – Nivoi kvaliteta za nesavršenosti

Kratka biografija:



Ljubiša Trivković rođen u Loznici 1992. god. Master rad na Fakultetu Tehničkih Nauka iz oblasti proizvodnog mašinstva, odnosno tehnologije zavarivanja.



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

**BIOGAS POSTROJENJE ZA MALO SEOSKO POLJOPRIVREDNO DOMAĆINSTVO
 BIOGAS PLANT FOR THE SMALL RURAL AGRICULTURAL FARM**Dušan Pejović, *Fakultet tehničkih nauka, Novi Sad***Oblast – MAŠINSTVO**

Kratak sadržaj – Ovaj rad opisuje faze u dobijanju biogasa, procedure, postrojenja, izvore i produkte ovog procesa. U radu je takođe izvršena analiza biogas postrojenja na maloj seoskoj farmi sa 10 krava. Projektovano postrojenje je na bazi indijskog digestora sa temperaturnom kontrolom, automatskim punjenjem i pražnjenjem i odsumporavanjem. Investicija za jedno ovakvo postrojenje bi bila oko 8500€. Pokazano je da je farma isuviše malog kapaciteta da bi ovakva investicija bila energetski isplativa.

Abstract – This work describes the stages in obtaining biogas, procedures, facilities, resources and products of this process. Analysis of biogas plant on a small rural farm with 10 cows is also shown. Plant design is based on indian digester with temperature control, automatic filling and emptying and desulphurization. Investment for such digester and additional equipment would be around € 8,500. It is shown that observed farm is of too small capacity for such investment to be viable.

Ključne reči: Biogorivo, biogas, fermentacija, digestor, bioenergija.

1. UVOD

Biogoriva su poslednjih decenija prepoznata kao potencijalno kvalitetna zamena za fosilna goriva. Iako je teško naći zadovoljavajuću definiciju biogoriva, izraz biogorivo se često koristi da se opišu pozitivne karakteristike zamene tradicionalnih goriva nekim novim energentima. U suštini, biogorivo je nov obnovljiv izvor energije, koji nastaje redukcijom i preradom biomase, a čija se energija dobija fiksacijom ugljenika, tj. redukcijom ugljenika iz vazduha u organska jedinjenja.

2. BIOGAS

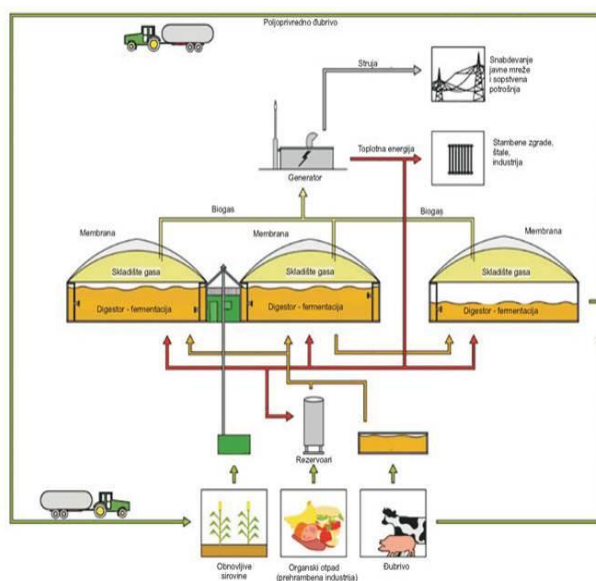
Biogas je mešavina metana (CH_4) i ugljen-dioksida, koji nastaje pri razgradnji organskih materija pod anaerobnim uslovima. U pitanju je veoma kvalitetno gorivo koje može da zameni fosilna goriva. U današnje vreme evidentan je stalni pad rezervi fosilnih goriva, rast ukupnih troškova, ugrožavanje životne sredine nepravilnim odlaganjem smeća, tako da pronalaženje rešenja za tretman biološkog otpada i organskih materija postaje veoma bitno pitanje.

Proizvodnja i primena biogasa ima pozitivne efekte u očuvanju životne sredine, tj. biogas ne doprinosi globalnom zagrevanju i stvaranju efekta staklene bašte. Pored dodatnih prihoda, ostatak nakon fermentacije (biootpad) sadrži elemente kao što su azot, fosfor i kalijum, koji služe kao izvršno đubrivo. Omogućeni su zatvoreni ciklus hranjivih materija i fleksibilnost u korišćenju različitih vrsta biomase. Takođe, smanjeni su neprijatni mirisi i prisustvo insekata, a povećana je veterinarska sigurnost.

Biogas je proizvod anaerobnog truljenja biomase i sadrži u proseku oko 60% metana, 35% ugljen dioksida i 5% smeše vodonika (H_2), azota (N_2), amonijaka (NH_3), sumpor-vodonika (H_2S), ugljen-monoksida (CO), kiseonika (O_2) i vodene pare (H_2O). Ovakav sastav biogasa može se koristiti za dobijanje toplotne ili električne energije, ili njihove kombinacije. Toplotna moć biogasa srazmerna je količini metana i kreće se u granicama 25-26MJ/Nm³ [1].

Postoji širok spektar materija za proizvodnju biogasa, kao što su: tečno i čvrsto stajsko đubrivo, posebno priključen biološki otpad iz stambenih jedinica, obnovljive materije kao što su kukuruzna silaža, semenke koje se ne koriste za hranu, kanalizacioni mulj i masti, korišćena maziva, trava, biološki otpad iz klanica, pivara, destilerija, proizvodnje vina, prerade voća, mlekarar, šećerana, industrije celuloze, i slično.

Struktura postrojenja u kojem se odvija proces proizvodnje biogasa prikazana je na Slici 1.



Slika 1. Biogas postrojenje

NAPOMENA:

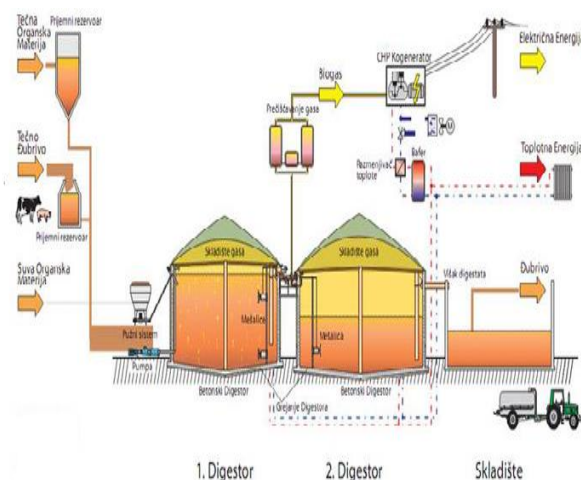
Ovaj rad proistekao je iz master rada čiji mentor je bila prof. dr Marija Vičević.

Proizvodnja biogasa se odvija u zagrejanim, zatvorenim reaktorima bez prisustva vazduha (digestorima) kako bi se stvorili uslovi za fermentaciju organskog materijala. U digestoru se moraju obezbediti odgovarajuća temperatura i dobro mešanje.

Postoji više tipova digestora, koji se razlikuju po veličini, konstrukciji, načinu odvijanja procesa u njima, materijala od koga su sastavljeni, mogućnosti da skladište ili ne skladište proizvedeni biogas, sa mogućim dogrevanjem ili nedogrevanjem supstrata, i drugi.

2.1. Proizvodnja biogasa iz stajnjaka domaćih životinja

Procenjuje se da bi u narednih 10 godina bilo moguće da se u Srbiji ostvare investicije u pet biogas postrojenja na farmama svinja i pet biogas postrojenja na farmama goveda. Na ovaj način bio bi prerađen stajnjak od oko 11.000 uslovnih grla goveda i 8.500 grla svinja u količini od oko 370.000m³, na godišnjem nivou. Doprinos u očuvanju životne sredine ogledao bi se u sprečavanju emisije od oko 5.000.000m³ metana i 20-60 puta manje ugljen-dioksida [2]. Ove farme bi kao supstrat koristile silažni kukuruz, otpatke iz klanica, šećerana i prehrambene industrije. Preduslov je usvajanje pravnih i tehničkih propisa o proizvodnji i korišćenju biogasa u državi, koji treba da budu usklađeni sa pravilima i propisima koji važe u EU. Slikom 2. dat je šematski prikaz postrojenja za dobijanje biogasa iz stajnjaka domaćih životinja.



Slika 2. Šematski prikaz procesa proizvodnje biogasa iz stajnjaka domaćih životinja

2.2. Energija i snaga postrojenja za proizvodnju i korišćenje biogasa

Toplotna energija koja se dobija sagorevanjem biogasa u odgovarajućim ložištima određuje se na osnovu izvora biogasa. Za biogas koji je proizveden iz stajnjaka domaćih životinja ona se određuje izrazom [3]:

$$Q = m \cdot H_d \cdot f_{sj} \cdot N \cdot \eta_s \cdot t \quad [MJ] \quad (1)$$

dok se za biogas proizveden fermentacijom poljoprivrednih kultura i otpadaka određuje po formuli [3]:

$$Q = m \cdot H_d \cdot M \cdot \eta_s \cdot t \quad [MJ] \quad (2)$$

Na drugoj strani, snaga biogasa proizvedenog iz stajnjaka domaćih životinja ili iz poljoprivrednih kultura i otpadaka izračunava se kao [3]:

$$P = \frac{Q}{243600} [MW] = \frac{Q}{86,4} [kW] \approx 11,6 \cdot Q [W] \quad (3)$$

pri čemu su:

H_d – donja toplotna moć biogasa [MJ/m³],

m – dnevni prinos biogasa [m³/kg],

f_{sj} – faktor stočne jedinice [SJ/kom],

M – masa OSM poljoprivrednih kultura i otpadaka [kg],

N – broj jedinica životinja [kom],

η_s – efikasnost sistema i

t – broj dana.

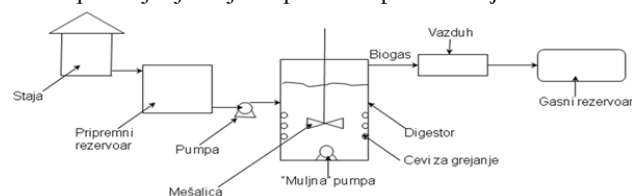
3. TEHNIČKI PRORAČUN

U tehničkom proračunu prikazano je domaćinstvo koje poseduje savremenu štalu od 10 krava. Proračun se zasniva na 10 krava prosečne težine 500kg. Dnevna količina izmeta, isparljivih čvrstih materija, suve materije, uz dodatak vode i raspoloživi kompost prikazani su u Tabeli 1.

Tabela 1. Početni parametri

Životinja	Ukupno žive mase [kg]	Dnevno izmeta po životinji [kg]	Ukupno izmeta, M_G [kg]	Isparljive čvrste materije VS [kg]	Suva materija SM [kg]	Dodatak vode [l]	Substrat, M_S [kg]
Krava	5000	40	400	45	58,1	340	740

Šema postrojenja koje se predlaže prikazana je na Slici 3.



Slika 3. Digestor i dodatna oprema

3.1. Proračun digestora

Vertikalni cilindrični rezervoar od betona će se koristiti kao digestor (BR, bioreaktor) zbog jednostavne izvedbe i povoljnog odnosa troškovi/efikasnost. Hidrauličko vreme zadržavanja supstrata u digestoru može se izabrati kao prosečno za ovaj tip biomase i iznosi $t_{BR} = 30$ dana, a stepen sigurnosti se može izabrati kao $f_{VBR} = 1,25$ da bi se naći u reaktoru. Odnos između visine i prečnika digestora treba da bude $H_{BR}/D_{BR} \approx 1/2$ [4]. Gustina tečnog stajnjaka ρ_G može da se pretpostavi da je približno jednaka gustini vode $\rho_w \approx \rho_G$. U digestoru će se koristiti propelerska mešalica (kao dodatna oprema) koju pokreće motor zaronjen u supstrat, a mešalica se može podešavati po visini.

Zapremina digestora može se izračunati kao:

$$V_{BR} = M_S \cdot \frac{t_{BR}}{\rho_G} \cdot f_{VBR} = 740 \frac{kg}{dan} \cdot \frac{30 dan}{1000 \frac{kg}{m^3}} \cdot 1,25 = 27,75 m^3$$

gde su:

V_{BR} - zapremina digestora (bioreaktora),
 M_S - količina substrata - zbir količine izmeta i dodate količine vode (Tabela 1),
 t_{BR} - hidrauličko vreme zadržavanja,
 ρ_G - gustina tečnog stajnjaka,
 f_{VBR} - stepen sigurnosti.

Prečnik digestora D_{BR} računa se pomoću poznate radne zapremine digestora V_{BR} i pretpostavljene visine (dubine) digestora H_{BR} i iznosi 4,2m.

3.2. Proračun pripremnog rezervoara

Pripremini rezervoar (PT, *engl. preparation tank*) je cilindričnog oblika sagrađen takođe od betona. U pripremnom rezervoaru će se čuvati samo dnevna proizvodnja stajnjaka i to najduže 10 dana ukoliko se vrše čišćenje i održavanje digestora.

Da ne bi došlo do izlivanja tečnog đubriva iz rezervoara, može se uzeti faktor sigurnosti $f_{VPT}=1,25$. Odnos između visine i prečnika rezervoara mora biti $H_{PT}/D_{PT} \approx 2$ [4].

Dimenzije rezervoara mogu se proračunati kao i u slučaju digestora, pa se prvo može odrediti zapremina rezervoara.

$$V_{PT} = M_G \cdot \frac{t_{PT}}{\rho_G} \cdot f_{VPT} = 400 \frac{kg}{dan} \cdot \frac{10 dan}{1000 \frac{kg}{m^3}} \cdot 1,25 = 5 m^3$$

gde su:

V_{PT} - zapremina pripremnog rezervoara,
 M_G - ukupna količina izmeta (Tabela 13),
 t_{PT} - vreme zadržavanja izmeta u rezervoaru,
 ρ_G - gustina tečnog stajnjaka,
 f_{VPT} - stepen sigurnosti.

Prečnik rezervoara D_{PT} se može zatim odrediti pomoću sada poznate zapremine rezervoara V_{PT} i pretpostavljene visine (dubine) rezervoara H_{PT} i iznosi 1,5m.

3.3. Proračun grejnih cevi

Za fermentaciju u digestoru može se izabrati mezofilni opseg, pa se može izabrati temperatura $t_{BR} = 50^\circ C$. Najniža spoljna temperatura zimi je $t_A = -20^\circ C$. Specifična toplota supstrata je (kao i za gustinu, može se pretpostaviti ista kao vode) $c_S = c_W = 4,2 kJ/kg^\circ C$ i mora se zagrejati sa (pretpostavljenih) $20^\circ C$ na $50^\circ C$, tako da se dobija razlika temperatura $\Delta t_S = 30^\circ C$.

Zidovi digestora će biti izolovani stiroporom debljine $s_{BR} = 0,1m$. Toplotna provodljivost kroz stiropor je $\lambda_S = 0,05 W/m^\circ C$ [5]. Iako substrat ne zauzima ceo prostor digestora, tj. nema kontakta sa krovom, pri izračunavanju gubitaka toplote uzima se u obzir kompletan zid digestora; gubici toplote kroz krov se zanemaruju jer je krov u kontaktu sa gasom i/ili vazduhom unutar i van njega.

Koeficijent prenosa toplote tečnog mešanog supstrata se može pretpostaviti kao $(\alpha_{BR})_i = 4000 W/m^2^\circ C$ [4] (unutrašnji koeficijent prenosa toplote), a (spoljašnji) koeficijent prenosa toplote zemlje izvan digestora je $(\alpha_{BR})_a = 400 W/m^2^\circ C$ [4]. Kada se sve to uzme u obzir može se izračunati ukupni koeficijent prolaza toplote:

$$k_{BR} = \frac{1}{\left(\frac{1}{(\alpha_{BR})_i} + \frac{s_{BR}}{\lambda_S} + \frac{1}{(\alpha_{BR})_a}\right)} = \frac{1}{\left(\frac{1}{4000} + \frac{0,1}{0,05} + \frac{1}{400}\right)} = 0,5 W/m^2 \cdot ^\circ C$$

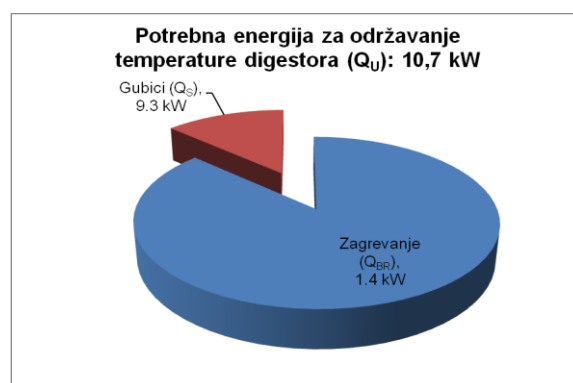
Snaga koja je potrebna za zagrevanje substrata može se izračunati kao:

$$Q_S = M_S \cdot c_S \cdot \Delta t_S = 740 \frac{kg}{dan} \cdot 4,2 \frac{kJ}{kg} \cdot 30K = 9,3 kW$$

Površina digestora koja sprovodi toplotu (A_{BR}) može se zračunati kao $40m^2$, a maksimalna razlika temperatura (Δt_{BR}) kao $70^\circ C$, pa su toplotni gubici digestora:

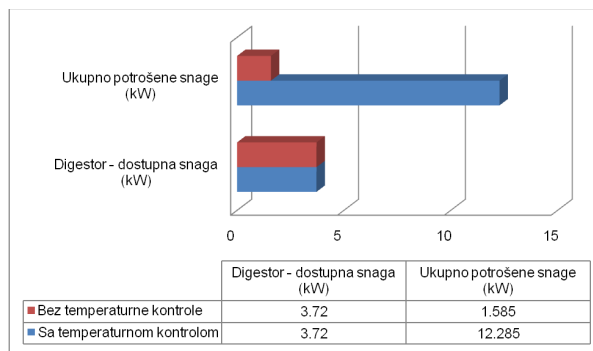
$$Q_{BR} = k_{BR} \cdot A_{BR} \cdot \Delta t_{BR} = 0,5 \frac{W}{m^2} \cdot ^\circ C \cdot 40m^2 \cdot 70^\circ C = 1,4 kW$$

Potrebna električna snaga za zagrevanje substrata može se dobiti kao zbir toplote koja je potrebna za zagrevanje substrata i toplotnih gubitaka i iznosi 10,7kW (Slika 3).



Slika 3. Potrebna energija za održavanje temperature digestora

Energetska vrednost $1m^3$ biogasa se navodi u literaturi kao $22 MJ/m^3$, tj. $E_{BG} = 6,1 kWh/m^3$ [6], pa se prema tome od ovog postrojenja može očekivati oko 89,3 kWh energije po danu, ili preračunato u snagu, 3,72 kW po danu. Iz proračuna potrebne uložene snage za digestor i dodatnu opremu (mešalica 0,085 kW, pumpa 1 kW, kompresor 0,5 kW), ukupna snaga potrebna za predloženo postrojenje je 12,285 kW, pa se može zaključiti da je ovakvo poluautomatsko postrojenje za malu farmu od 10 krava neisplativo jer bi se trošilo više energije od proizvedene. Jedna opcija bi bila odustajanje od temperaturne kontrole kao najvećeg potrošača (Slika 4).



Slika 4. Prikaz energetske isplativosti postrojenja sa i bez temperaturne kontrole

Bez temperaturene kontrole bi ovakvo postrojenje bilo energetski isplativo.

4. EKONOMSKA ANALIZA

Troškovi građevinskih radova svode se na proračun troškova izgradnje digestora i pomoćne opreme. Potrebna zapremina betona koja je potrebna za izgradnju digestora i pripremnog rezervoara računa se na osnovu površine valjka poznatih dimenzija bez jedne osnove i debljine betona.

Cena betona je 65 €/m³. Kada se to uzme u obzir dobija se ukupna cena betona za izradu digestora i pripremnog rezervoara od €910 (Beton komerc-Priboj).

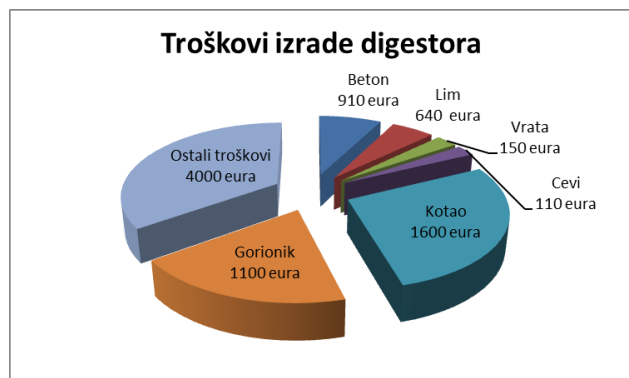
Za krovnu konstrukciju može da se koristi lim debljine 3mm. Površina koju je potrebno prekriti iznosi 31 m². Cena takvog lima, na našem tržištu, iznosi 20€/m², pa ukupni troškovi lima za pokrivanje digestora iznose €640 (Atenic komerc, Čačak).

Metalna vrata za digestor, razmatrajući srednju cenu, koštaju oko €150 (Atenic komerc, Čačak). Dimenzije vrata su 1x1m.

Cevi za grejanje supstrata, odnosno za pasterizaciju, koje su planirane da se koriste su prečnika φ30mm. Dužina potrebne cevi iznosi 37m. Troškovi cevi su prema tome €110 (Atenic komerc, Čačak).

Kotao od 50kW, potreban za zagrevanje supstrata u digestoru, košta oko €1600 (Radijator, Kraljevo), a gorionik €1100 (Citerm, Beograd). Ostali troškovi izgradnje (nepovratni ventili, fitinzi, radna snaga, pumpe i dr.) iznose oko €4000.

Na osnovu proračuna, iznosi ukupnih sredstava za izradu digestora, planiranih dimenzija i karakteristika, iznosi €8510, sa udelom troškova koji je prikazan Slikom 5.



Slika 5. Troškovi izrade digestora

Pošto je u prethodnoj analizi već pokazano da bi ovakvo postrojenje bilo energetski neisplativo, ukoliko bi se, kao što je prethodno već predloženo, odustalo od temperaturene kontrole, troškovi izrade ovakvog delimično kontrolisanog postrojenja bi se smanjili za oko €1700.

5. ZAKLJUČAK

Na primeru malog poljoprivrednog gazdinstva sa 10 krava, izvršen je proračun očekivane količine supstrata, dimenzionisan digestor, pripremini rezervoar, predložena dodatna oprema, izvršena analiza energetskog bilansa i dati troškovi. Proračunata zapremina digestora je

27,75m³, a pripremnog rezervoara 5m³. Količina supstrata na dnevnom nivou je 740 kg. Od ovakvog postrojenja može se očekivati oko 23m³ biogasa po toni supstrata. Prema tome, može se očekivati dnevna količina biogasa od 17,02 m³/dan, odnosno, s obzirom da se očekuje oko 86,2% ove dnevne količine, realni prinos biogasa bio bi 14,64 m³/dan.

Izračunata uložena snaga za predloženi digestor i dodatnu opremu je 12,3kW, pa se došlo do zaključka da je ovakvo poluautomatsko postrojenje za malu farmu od 10 krava neisplativo, jer bi se trošilo više energije od proizvedene. Predloženo je odustajanje od temperaturene kontrole, što bi smanjilo potrošnju na 1,6kW i na taj način bi ovakvo postrojenje bilo energetski isplativo.

Na osnovu proračuna, iznosi ukupnih sredstava za izradu digestora, planiranih dimenzija i karakteristika, iznosi €8510, uključujući temperaturenu kontrolu i automatizaciju protoka. Zbog prethodno naglašene energetske neisplativosti, predloženo je postrojenje bez temperaturene kontrole, čime bi se troškovi izrade ovakvog delimično kontrolisanog postrojenja smanjili na oko €6800.

Po izvršenim analizama i proračunima, u poljoprivrednom gazdinstvu koje raspolaže takvim stočnim fondom koje može da proizvede 200-250m³ biogasa mesečno može da se napravi mesečna ušteda od 7.000-9.000 dinara. U ovom slučaju proizvodnja biogasa iznosi oko 500m³ mesečno, što bi iznosilo oko 14.000 dinara uštede na mesečnom nivou. Uzimajući u obzir troškove uloženog i uštede koji se dobijaju instaliranjem postrojenja za biogas, ova investicija bi mogla biti isplaćena za 5 godina.

6. LITERATURA

1. A. Stefanović, „Alternativna goriva motora SUS. Tehnički aspekti, mogućnosti i korišćenja: Tečna i gasovita alternativna goriva“, Mašinski fakultet Univerziteta u Nišu, 2009.
2. IPA projekat br.: 2007CB16IPO006-2011-2-19, „Kako napraviti sopstveno biogas postrojenje“, 2013, Dostupno na: http://www.tf.ni.ac.rs/IPA_Bul_Ser/pdf%2013.pdf, preuzeto: 02.09.2016. godine.
3. M. Vićević, „Bioenergetska goriva“, Fakultet tehničkih nauka. Novi Sad, 2014.
4. D. Deublein and A. Steinhäuser, „Biogas from Waste and Renewable Resources. An Introduction“, WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim, 2008.
5. K. Fodor, „Provođenje toplote kroz zid“, Diplomski rad, Prirodno-matematički fakultet, Univerzitet u Novom Sadu, 2011.
6. C. Banks, „Anaerobic digestion and energy“, University of Southampton, 2016.

Kratka biografija:



Dušan Pejović rođen je u Užicu 1987. god. Master rad na Fakultetu tehničkih nauka iz oblasti Mašinstvo – Energetika i procesna tehnika odbranio je 2016. god.

PRIMENA GLOBALNOG SISTEMA ZA POZICIONIRANJE (GPS) U POLJOPRIVREDI APPLICATION OF GLOBAL POSITIONING SYSTEM IN AGRICULTURE

Dejan Varga, Dejan Nemec, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu su prikazane osnove globalnog sistema za pozicioniranje (GPS), istorijski razvoj, tehnički opis sistema, sa posebnim osvrtom na primenu u važnoj industrijskoj grani – poljoprivredi. Pored toga, izvršena je procena ekonomske opravdanosti ove primene.

Abstract – In this thesis the bases of a global positioning system (GPS), development through the history are shown. Technical description of the system is given, with special emphasis on application to an important industrial branch – agriculture. The economic feasibility of these applications is also assessed.

Ključne reči: GPS, Poljoprivreda, Ekonomska evaluacija

1. UVOD

GPS (Globalni sistem za pozicioniranje, engl. Global Positioning System) predstavlja, za sada, jedini potpuno funkcionalan globalni satelitski sistem za navigaciju (GNSS, engl. Global Navigation Satellite System).

GPS servis omogućava:

- 24-časovni servis na bilo kojoj tački planete (uz izvesna ograničenja po pitanju dostupnosti signala)
- veoma precizno određivanje pozicije u tri dimenzije (presudnu ulogu ima kvalitet prijemnika)
- određivanje tačne brzine kretanja i vremena
- pristup neograničenom broju korisnika

U ovom radu izvršena je analiza GPS sistema, pregled njegovog razvoja kroz istoriju i analiza primene u veoma važnoj industrijskoj grani: poljoprivredi.

2. ISTORIJSKI RAZVOJ

Razvoj satelitske tehnologije sredinom XX veka bio je presudan činilac u razvoju i realizaciji satelitskog navigacionog sistema. Ispitivanja prvog satelita Sputnik 1, lansirano 4. oktobra 1957. godine, rezultovala su otkrićem ključnog navigacionog koncepta. Uočeno je da se na osnovu Doplerovog efekta može odrediti pozicija satelita sa poznate lokacije na zemlji. Inverzno tome, koristeći Doplerov efekat se može odrediti pozicija na zemlji ukoliko je poznata pozicija satelita koji emituje signal.

22. decembra 1973. je zvanično objavljen početak realizacije koncepta GPS sistema. Realizacija je obuhvatala tri faze: ispitivanje, razvoj i implementaciju. Za vreme faze ispitivanja, napravljeni su i lansirani prvi Blok I sateliti i prototip kontrolne komponente, kao i različiti tipovi prijemnika.

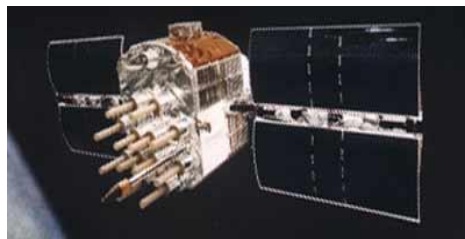
U razvojnoj fazi od 1979. dostignuta je početna konstelacija satelita daljim lansiranjem Blok I satelita i počela je proizvodnja Blok II satelita.

16. septembra 1983. godine, nakon što je Rusija oborila zalutali korejski avion, predsednik SAD odobrio je upotrebu GPS sistema u civilne svrhe. To je početak dvojne uloge GPS sistema.

U fazi implementacije, započetoj 1985. godine, kompletirana je konstelacija od 24 Blok II i IIA (A je oznaka za advanced, unapređen) satelita. Potpuna operativna sposobnost sistema proglašena je 17. jula 1995. godine.

Pošto je projektovani vek trajanja prvih GPS satelita u orbiti bio oko 7,5 godina, u cilju održavanja sistema operativnim, neophodno je konstantno unapređivati sistem i slati novije generacije satelita u orbitu. Trenutno se u orbiti nalazi 31 GPS satelit, od kojih 12 Blok IIR satelita, 7 Blok IIR(M) satelita i 12 Blok IIF satelita. Poslednji Blok IIF satelit je lansirano u martu 2016. godine.

Projektovani vek trajanja poslednjih, Blok IIF satelita je povećan na 12 godina. U fazi izrade je trenutno i najnovija generacija GPS satelita, Blok III sateliti koji bi uskoro trebali da uđu u operativnu upotrebu [1]. Na slikama 1. i 2. prikazane su prva i treća generacija Blok satelita.



Slika 1. Blok I satelit



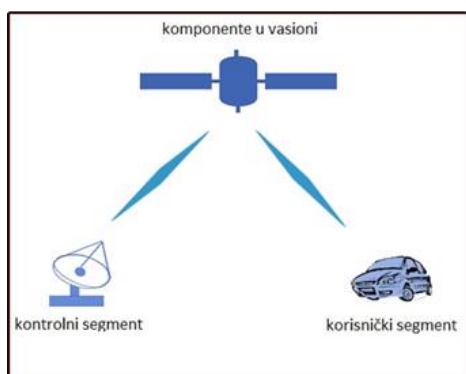
Slika 2. Blok III satelit

NAPOMENA:

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

3. TEHNIČKI OPIS SISTEMA

GPS sistem se sastoji od tri komponente: komponente u vasioni, kontrolne komponente i korisničke komponente. Ove komponente prikazane su na slici 3.



Slika 3. Komponente GPS sistema

3.1. Komponenta u vasioni

Komponentu u vasioni čine GPS sateliti u MEO (Medium Earth Orbit) orbiti Zemlje na visini od približno 20200 km. Broj i raspored satelita menjao se tokom vremena, a tehničke se karakteristike poboljšavale, kako se GPS sistem razvijao. Originalni koncept Blok II satelita predstavljaju 24 GPS satelita koji se kreću u 6 orbitalnih ravni, ravnomerno raspoređenih u odnosu na Zemlju (međusobno su razdvojene za ugao od 60°), koje su nagnute pod uglom od 55° u odnosu na ekvatorijalnu ravan (slika 3.1.1.).

U svakoj orbitalnoj ravni se kreću po 4 satelita, po orbitama koje su skoro kružne (ekscentričnost elipse 1°), međusobno pravilno raspoređeni po kružnici orbite, pod uglom od 90° jedan u odnosu na drugi. Prečnik orbita je oko četiri puta veći od prečnika Zemlje i svaki od satelita jednom obiđe svoju orbitu za 11 časova i 58 minuta, tako da u odnosu na površinu Zemlje svaki satelit svakog dana obiđe istu putanju. Ovaj broj i pravilan raspored satelita garantuje da se sa svake tačke na Zemlji u svakom trenutku na horizontu nalazi bar četiri satelita. To su četiri satelita neophodna za precizno određivanje pozicije GPS prijemnika.

Pošto sateliti izlaze iz funkcije zbog održavanja, kvarova ili isteka radnog veka, oko Zemlje kruži više satelita i često ih je aktivno više od 24. Sadašnja konstelacija se sastoji od 31 Blok IIR/IIR(M)/IIF GPS satelita koji su u funkciji, i to 12 Blok IIR, 7 Blok IIR(M) i 12 Blok IIF satelita.

3.2. Kontrolna komponenta

Kontrolnu komponentu čine stanice za praćenje satelita, kontrolne stanice i zemljišne antene. Stanice za praćenje satelita nalaze se na Havajima, Kejpu Kanaveralu na Floridi, Kvajlin ostrvu, ostrvu Asension, ostrvu Dijego Garsija i Kolorado Springsu, u Koloradu. Uloga ovih šest stanica je da prate kretanje satelita i prikupljaju podatke o tačnoj poziciji, brzini i ispravnosti rada satelita. Stanice za praćenje satelita imaju cezijumske atomske časovnike i mogu istovremeno da prikupljaju podatke od 11 satelita. Podaci se šalju glavnoj kontrolnoj stanici u Kolorado Springsu. Tu se vrše proračuni kako bi se predvidelo kretanje satelita i rad časovnika i preko zemljišnih antena

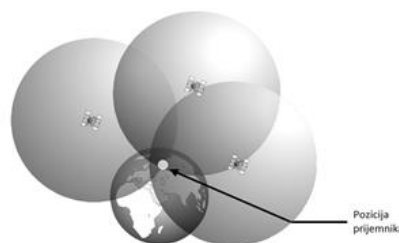
koje se nalaze na Kvajlin ostrvu, na ostrvu Asension, ostrvu Dijego Garsija i Kolorado Springsu, satelitima se šalju ažurirani podaci o njihovoj tačnoj poziciji i vremenu. Pri tome se koristi S frekvencijski opseg. Ažuriranje se vrši dva puta dnevno, čime se vrše fina podešavanja sistema. Novije generacije satelita su u stanju da međusobno komuniciraju i sinhronizuju podatke, tako da preciznost određivanja pozicije ne bi bila bitno narušena ni kad bi sateliti danima radili nezavisno od kontrolne komponente na Zemlji.

3.3. Korisnička komponenta

Korisničku komponentu čine GPS prijemnici na Zemlji koji se koriste u vojne ili civilne svrhe. Prijemnici mogu biti komponente uključene u druge uređaje, kao npr. u mobilni telefon, u časovnik i slično, ili samostalni uređaji. Osnovne komponente GPS prijemnika su antena podešena na frekvencije GPS satelita, kristalni oscilator koji služi kao časovnik i mikroprocesor koji obrađuje signale. Prijemnici se često klasifikuju prema tome koliko kanala imaju. Svaki kanal prati po jedan satelit. Stariji modeli su imali četiri do pet kanala, a današnji uglavnom 12 do 24 kanala. GPS prijemnici mogu imati u sastavu komponentu koja prima diferencijalne signale. Diferencijalni signal se dobija preko standardnog RS-232 porta ili preko interne antene. Od 2006. godine čak i jeftini prijemnici uključuju i prijemnike za oblasne sisteme augmentacije.

3.4. Određivanje pozicije prijemnika

Određivanje pozicije vrši se na osnovu merenja udaljenosti od tri ili više satelita. Svaki satelit emituje mikrotalasnu sekvencu radio signala koja je poznata GPS prijemniku. Prijemnik prima taj signal, i u zavisnosti od vremena proteklog od momenta slanja do momenta prijema u stanju je da odredi tačnu udaljenost satelita. Podrazumevana brzina signala je brzina bliska svetlosnoj. Svaki satelit emituje signal kojim se identifikuje, saopštava svoju poziciju i vreme kada je emitovao signal. Ako se zna udaljenost prijemnika od satelita i pozicija satelita, poznato je da se prijemnik nalazi negde na sferi određenog poluprečnika u čijem je centru satelit. Pošto su poznate pozicije tri satelita i udaljenost prijemnika od svakog od njih, postupkom trilateracije se može odrediti pozicija prijemnika (slika 4.). Trilateracija se bazira na činjenici da se tri sfere seku u najviše dve tačke (od kojih jedna obično nema smisla jer se nalazi u svemiru) [2].



Slika 4. Pozicija prijemnika je u tački u kojoj se tri sfere seku

4. DIFERENCIJALNI GPS SISTEMI

Za diferencijalni GPS se koriste prijemnici na fiksnim stanicama čije su koordinate poznate. Stalno se proračunava pozicija ovih prijemnika na osnovu GPS signala i poredi se sa stvarnom, poznatom pozicijom.

Usled raznih faktora koji unose grešku, stvarna i proračunata pozicija nisu jednake. Tako je poznata greška u proračunu pozicije na lokaciji fiksne stanice, a pretpostavlja se da će se na obližnjim lokacijama (na udaljenostima do 200 km) pojavljivati ista vrsta i vrednost greške. GPS prijemnici, kada se nalaze u blizini ovih fiksnih stanica dodatno primaju ovu informaciju i koriste je za korekciju greške svoje pozicije. Potrebno je što više ovakvih grešaka otkloniti da bi se dobile što je moguće tačnije pozicije [3]. Augmentaciju predstavljaju metode za poboljšanje preciznosti, pouzdanosti, integriteta i dostupnosti GPS sistema koje se oslanjaju na informacije dobijene od drugih izvora, a ne GPS satelita, a koje se takođe koriste u proračunu pozicije. Postoji više takvih sistema i uglavnom se razlikuju prema tome kako GPS senzor prima ove informacije. Neki sistemi emituju informacije o faktorima koji unose grešku, npr. grešku časovnika, grešku pozicije satelita, grešku koju unosi jonosfera. Drugi sistemi pružaju informacije o tome kolika je bila greška u prošlosti, a treći pružaju dodatne navigacione informacije koje se mogu koristiti u proračunu pozicije.

5. GPS U POLJOPRIVREDI

Munjeviti napredak u elektronici, računarskim tehnologijama kao i u satelitskim tehnologijama, podstakao je, kao nikada do sada, interesovanje za razvoj sistema navođenja vozila uz pomoć GPS sistema za navigaciju. U ovom poglavlju će biti opisana primena GPS satelitskog sistema u jednoj veoma bitnoj grani industrije: poljoprivredi. U poljoprivredi su zahtevi u pogledu pozicioniranja specifični. Zahteva se znatno viša tačnost nego za potrebe u, na primer saobraćaju, gde nekoliko metara ne predstavlja grešku od odlučujućeg značaja.

Najdirektnija posledica opremanja poljoprivredne mehanizacije, pre svega traktora, opremom za satelitsko pozicioniranje i automatsko upravljanje je preciznije uklapanje prohoda. Ovo podrazumeva preciznije vođenje mašina po pravcu i smanjenje preklopa. Pod smanjenjem preklopa se podrazumeva smanjenje površine preklapanja dva uzastopna prohoda koje načini traktor vučući za sobom određenu priključnu mašinu (plug, tanjirača, podrivač, setvospremač, drljača, prskalice, rasipač mineralnog đubriva, sejalice, itd...). Osim preklapanja može doći do situacije da prilikom obrade zemljišta bez satelitskog navođenja između dva uzastopna prohoda ostane neobrađeni deo. To je veoma izražen problem kod rukovanja poljoprivrednim mašinama širokih radnih zahvata, kada upravljač ne može najbolje iz kabine da odredi radni zahvat mašine.

Iz preciznijeg vođenja mašina po pravcu proističu sledeće direktne prednosti i uštede: smanjenje gubitaka i oštećenja biljne mase i kvalitetnija struktura. Iz smanjenja broja prohoda po parceli proističu sledeće direktne prednosti i uštede: smanjenje poljoprivrednih inputa, smanjenje potrošnje goriva, poboljšanje ekoloških uslova, povećanje produktivnosti rada, itd... Stepen navedenih direktnih prednosti i ušteta pri određenoj poljoprivrednoj operaciji zavisi od samih zahteva operacije [4].

Samo korišćenje satelitske navigacije donosi i dve indirektno prednosti i uštede koje se odnose na sve poljoprivredne operacije [5]:

- Prva je mogućnost rada noću (moguće je raditi noću i bez satelitske navigacije, manuelnim navođenjem, ali je efikasnost umnogome smanjena), što je naročito bitno pri ograničenim vremenskim rokovima za obavljanje određenih poljoprivrednih operacija. To ograničenje je uglavnom posledica loših vremenskih prilika. Preklop pri operaciji podrivanja ili tanjiranja (slike 6.2. i 6.3.) i manuelnom navođenju danju je 10 – 20 cm, a noću i 40 – 50 cm. Preciznim satelitskim navođenjem preklapanje može da se svede na 5 – 10 cm, i to u svim uslovima.
- Druga indirektna prednost odnosi se na to što većina sistema za satelitsko navođenje ima integrisane i druge funkcije, koje mogu dobro da posluže za menadžment, knjigovodstvo, razne dokumentacije i planiranje proizvodnje u narednom periodu.

5.1. GPS i njegova tačnost za potrebe poljoprivrede

Ugrađeni u traktore i kombajne, GPS prijemnici prosleđuju tačnu geografsku poziciju, kao i tačno vreme reda nanosekunde. Zajedno sa informacijama o trenutnom nivou prinosa, količini tečnosti koju primenjuje rasprskivač i gustini setve, ovi podaci postaju značajni za dalju analizu u narednim fazama obrade zemljišta.

Za sveukupan uvid u stanje parcele i prinosa, potrebni su nam podaci koji mogu biti satelitski snimci, aero snimci, novoi primene hemijskih sredstava, rezultati analize zemljišta, podaci o klimi, podaci o korovskim biljkama, itd...

Problem koji je prisutan je što do GPS prijemnika stižu degradirani signali poslani sa satelita. Neki od uzroka degradacije signala su:

- prolazak kroz sve slojeve zemljine atmosfere,
- „efemeris“ – uticaj sunca i meseca na male promene u orbiti satelita (koriguje se slanjem korekcija sa 5 zemaljskih stanica),
- greška atomskih satova koji se nalaze na satelitima,
- greška samog prijemnika – u zavisnosti od namene,
- „višestruka putanja“ – odbijanje signala od velikih objekata u blizini prijemnika.

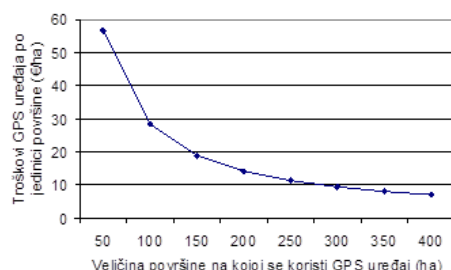
Napredniji prijemnici mogu korigovati atmosfersku i grešku višestruke putanje. Značajno poboljšanje tačnosti se dobija, kao što je navedeno ranije u radu, korišćenjem diferencijalne korekcije koju uvodi dodatni prijemnik, koji se nalazi relativno blizu prijemnika sa koga se očitava pozicija. Ovaj dodatni prijemnik se naziva bazna stanica. Bazna stanica ima precizno određenu poziciju i ona pomoću transmitera šalje diferencijalni korekcionni signal pokretnom prijemniku, koji u svoje očitavanje pozicije uvršćuje korekciju koju prima od bazne stanice. Korekcija u realnom vremenu je neophodna ukoliko se želi dobiti korekcija u trenutku prikupljanja podataka, dok je odložena obrada korekcije potrebna u slučajevima kada se želi dobiti precizniji podatak o poziciji gde je izvršeno merenje, a tako precizan podatak o poziciji nije potreban u realnom vremenu.

Nije za svaku poljoprivrednu operaciju potreban jednak nivo preciznosti signala.

5.2. Uštede ostvarene primenom GPS navigacije

Na grafikonu 1. su prikazani troškovi korišćenja GPS uređaja u zavisnosti od površine koja se obrađuje [6].

Grafikon 1. Troškovi GPS uređaja izraženi po jedinici površine, u zavisnosti od veličine površine na kojoj se primenjuje



5.3. Vrste navođenja

Kao što je pomenuto ranije u tekstu ne zahteva svaka poljoprivredna operacija jednak nivo preciznosti GPS signala. U skladu sa tom činjenicom postoje tri vrste navođenja pomoću GPS signala [7], [8]:

- **Manuelno navođenje:** koristi se kada zahtevana poljoprivredna operacija ne zahteva veliku tačnost navođenja, kao što je razbacivanje mineralnih đubriva, prskanje, setvospremiranje, tanjiranje,... Ove potrebe zadovoljava EGNOS korekcionni signal, tačnosti uklapanja prohoda $\pm (15 - 20)$ cm, koji je pritom besplatan. Čovek sve vreme prateći signalizacije lampice na navigacionom displeju upravlja traktorom.
- **Kombinovano manuelno/automatsko navođenje:** funkcioniše tako što je na upravljač traktora prikačen elektromotor, koji funkcioniše tako što prima električne signale od navigacionog kontrolera i pretvara ih u mehaničke pokrete (okretanje volana). Čovek u svakom trenutku može da preuzme upravljanje. Koristi se kod svih vrsta poljoprivrednih operacija, i podržava sve vrste korekcionnih signala. Sistem se sastoji od antene, navigacionog kontrolera, displeja za navigaciju i elektromotora prikačenog na upravljač traktora. Moguće su dve varijante elektromotora koji se priključuju na volan traktora:
 - Elektromotori koji mogu da se prenose iz traktora u traktor, veoma laka ugradnja na svaki volan;
 - Elektromotori koji su integrisani u volan i nemaju mogućnost prenošenja iz vozila u vozilo;
- **Automatsko navođenje:** sistem se sastoji od GPS antene, displeja za navigaciju u kabini, upravljačkog senzora montiranog na prednjim točkovima traktora, kontrolera autopilot sistema (navigacionog kontrolera), elektrohidrauličnog ventila. Funkcioniše tako što antena prima signal koji šalje displeju za navigaciju. Upravljački senzor montiran na prednjim točkovima pri kretanju traktora meri ugao točkova, zatim šalje signal kontroleru autopilot sistema koji ima senzore nagiba za sve tri ose (T3 tehnologija) kompenzujući time GPS poziciju pri naganjanju i zanošenju traktora u svim pozicijama. Kontroler autopilot sistema preko elektrohidrauličnog ventila deluje direktno na volan. Automatsko navođenje (autopilot) se koristi kod svih vrsta poljoprivrednih operacija.

6. ZAKLJUČAK

Savremena poljoprivreda u razvijenim zemljama, ali u poslednje vreme i u Srbiji, suočena je sa zahtevima da se ostvari što viši kvalitet proizvoda, da se proizvodi po što nižim cenama i da bude što manje štetnih uticaja na životnu sredinu. Jedan od bitnijih načina da se ovi uslovi ostvare, pored racionalnijeg iskorišćavanja goriva, mineralnih đubriva, herbicida, pesticida i insekticida, jeste primena satelitskog navođenja poljoprivrednih mašina. Kao što je prikazano u izlaganju gore, cena korišćenja satelitskog navođenja poljoprivrednih mašina, rapidno opada sa povećanjem površina zemljišta koje se obrađuje. Tu se javlja problem 'malih' poljoprivrednih proizvođača, koji ne nalaze interes u korišćenju ovih sistema. Izazov za budućnost predstavlja približavanje ovih sistema baš ovoj ciljnoj grupi.

Da bi se GPS sistemi još više integrisali u oblast poljoprivrede, potrebno je kod svakog poljoprivrednog proizvođača podići nivo informatičkog znanja, tako da je pored ulaganja u mašine potrebno ulagati i u znanje, bez kojeg moderni poljoprivredni sistemi ne mogu imati punu efikasnost pri eksploataciji.

7. LITERATURA

- [1]<http://www.gps.gov/systems/gps/space/>
 - [2]<http://www.viser.edu.rs/download/uploads/12497.pdf>
 - [3]<http://gps sistemi.wikispaces.com/Tehnike+za+poboljšanje+preciznosti>
 - [4]http://arhiva.nara.ac.rs/bitstream/handle/123456789/532/PT_04-2012%20-%20Markovic.pdf?sequence=1
 - [5]<http://www.vps.ns.ac.rs/SB/2013/3.1.pdf>
 - [6]Milan Martinov, *Primena GPS pozicioniranja za navođenje i uklapanje prohoda u poljoprivredi Vojvodine*, Fakultet tehničkih nauka, Novi Sad, 2008.
 - [7]https://d3u1quraki94yp.cloudfront.net/caseih/emea/EMEAASSETS/Products/AFS%20-%20Advanced-Farming-Systems/Brochures/13c8003_com_afs_2014_.pdf
 - [8]https://d3u1quraki94yp.cloudfront.net/caseih/NAFTA/NAFTAASSETS/Products/Advanced-Farming-Systems/Brochures/AFS_Brochure.pdf
- Svim adresama pristuljeno u septembru 2016.

Kratka biografija:



Dejan Varga rođen je u Bečeju 1983. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Telekomunikacije odbranio je 2016. godine



Dejan Nemec rođen je 1972. god. Diplomirao, specijalizirao i magistrirao je na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva. Oblast interesovanja su telekomunikacije i obrada signala.

INTEGRACIJA SA ELEKTRONSKIM SERVISIMA MINISTARSTVA FINANSIJA**INTEGRATION WITH MINISTRY OF FINANCE E-SERVICES**

Vladimir Šobić, Branko Milosavljević, Gordana Milosavljević *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu je prikazano jedno rešenje integracije sa elektronskim servisima ministarstva finansija. Konkretna implementacija biće data na primeru informacionog sistema Fakulteta tehničkih nauka.

Abstract – This paper presents one solution for integration with Ministry of finance e-services. Concrete implementation will be shown on the example of information system of the Faculty of Technical Sciences.

Ključne reči: XML, Web servisi, Spring framework, MyBatis, Castor XML, elektronski servisi, e-Uprava

1. UVOD

Ovaj rad se bavi opisom povezivanja postojećeg informacionog sistema sa elektronskim servisima ministarstva finansija. Potreba za ovakvom vrstom integracije javila se nakon donošenja zakona o rokovima izmirenja novčanih obaveza u komercijalnim transakcijama [1], uredbe Vlade Srbije o registru zaposlenih [2], zakona o izmenama i dopunama zakona o poreskom postupku i poreskoj administraciji [3] i pravilnika o podnošenju poreske prijave elektronskim putem [4]. Ovim zakonima, uredbom i pravilnikom, napravljen je korak ka omasovljenju elektronskih servisa samog ministarstva i olakšana je komunikacija sa korisnicima budžetskih sredstava i privrednim subjektima.

Ministarstvo finansija je stavilo na raspolaganje tri web aplikacije, odnosno elektronska servisa: RINO, Registar-Zaposlenih i ePorezi. Zbog konciznosti rada, implementacija integracije će biti data na primeru integracije sa RINO sistemom. Arhitektura sistema i implementacija integracije sa druga dva sistema je identična.

2. SPECIFIKACIJA SISTEMA

Ministarstvo finansija – Uprava za trezor (u daljem tekstu samo Uprava za trezor) počela je u drugom kvartalu 2013. godine realizaciju projekta RINO – Registar izmirenja novčanih obaveza. Cilj ovog projekta je da se u skladu sa zakonom o rokovima izmirenja novčanih obaveza u komercijalnim transakcijama i pravilnika o načinu i postupku vršenja nadzora nad sprovođenjem zakona o rokovima izmirenja novčanih obaveza u komercijalnim transakcijama korisnika javnih sredstava [1], u Upravi za trezor vrši evidencija i nadzor nad rokovima izmirenja novčanih obaveza u komercijalnim transakcijama između korisnika javnih sredstava i privrednih subjekata kada su korisnici javnih sredstava dužnici. Nadzor se vrši na osnovu obaveznih podataka koje dostavljaju korisnici budžetskih sredstava i to:

NAPOMENA:

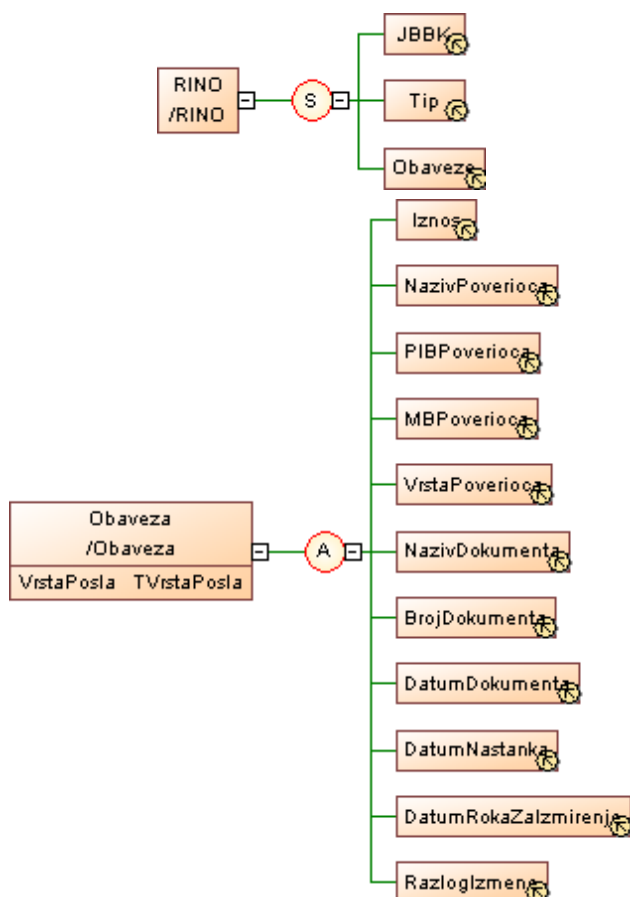
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Branko Milosavljević, red. prof.

1. Naziva korisnika i JBBK (jedinstvenog broja budžetskog korisnika kod Uprave trezora),
2. Naziv poverioca,
3. PIB(poreski identifikacioni broj) i MB(matični broj) poverioca,
4. Naziv dokumenta kojim je preuzeta novčana obaveza (ugovor, sporazum i dr.), kao i odgovarajućeg zahteva za isplatu novčane obaveze (ulazni račun, privremena ili konačna situacija ili drugi odgovarajući zahtev za isplatu),
5. Broj i datum dokumenta kojim je preuzeta novčana obaveza, kao i broj i datum odgovarajućeg zahteva za isplatu novčane obaveze,
6. Iznos novčane obaveze iz dokumenta kojim je preuzeta novčana obaveza, kao i iznos novčane obaveze iz odgovarajućeg zahteva za isplatu novčane obaveze,
7. Datum nastanka novčane obaveze, utvrđen u skladu sa zakonom,
8. Rok za izmirenje novčane obaveze - datum valute plaćanja (ugovoreni rok ili zakonski u slučaju ako rok nije ugovoren)
9. Datum izmirenja novčane obaveze,
10. Datum evidentiranja obaveze (datum ažuriranja podataka),
11. Iznos izmirene novčane obaveze.

Registar RINO je elektronska baza podataka, koja sadrži podatke o prijavljenim obavezama i izvršenim izmirenjima po tim obavezama u komercijalnim transakcijama [1]. Korisnici javnih sredstava ove podatke dostavljaju preko web aplikacije, ručnim unosom podataka kroz korisnički interfejs ili učitavanjem XML dokumenata, čija je struktura propisana internim tehničkim uputstvom Uprave za trezor. Ručni unos podataka je pogodan za budžetske korisnike sa malim brojem novčanih obaveza.

Kako se na primeru informacionog sistema Fakulteta tehničkih nauka, na godišnjem nivou evidentira preko 10.000 ulaznih računa, ručni unos tih obaveza nije dolazio u obzir. Bilo je potrebno razviti integracioni sloj servisa koji će proizvoditi XML dokumenta odgovarajuće strukture ali i biti u stanju da parsira XML dokumenta koja dolaze od RINO web aplikacije Uprave trezora, jer komunikacija sa ovom web aplikacijom nije jednosmerna.

Prilikom učitavanja XML dokumenata na web aplikaciju, vrši se formalna i logička provera ispravnosti samog dokumenta. Formalna kontrola podrazumeva proveru da li je XML dokument validan i da li je u skladu sa XML schema-om, čiji je deo prikazan na slici 1.



Slika 1. Grafički prikaz XML schema-e za element *Obaveza*

Logička kontrola se odnosi na proveru datuma nastanka novčanih obaveza koji ne mogu biti u dalekoj prošlosti niti budućnosti. Proverava se i da li postoji korisnik javnih sredstava sa unetim JBBK-om. Takođe, ne mogu se dva puta evidentirati iste obaveze tj. dokumenti sa istim originalnim brojem računa. Najbitnija provera se odnosi na identifikaciju naloga u platnom prometu Uprave trezora za izmirenje nastale obaveze. Time se sprečava da korisnik javnih sredstava unese pogrešan ili lažni podatak o izmirenju novčane obaveze. Polja koja označavaju šifarnike (PIB i MB) se validiraju u odnosu na postojanje u odgovarajućim šifarnicima. Ukoliko je bilo koja od ovih provera neuspešna, podaci iz XML dokumenta se neće uneti u RINO bazu podataka.

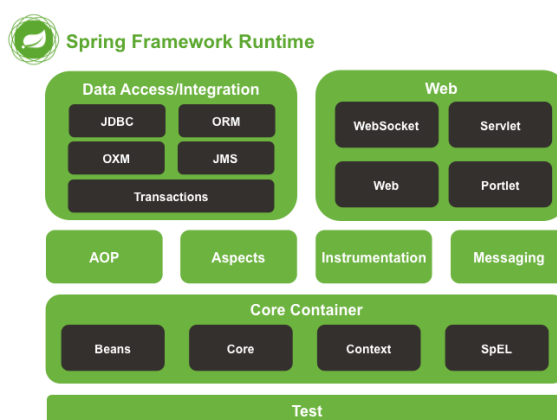
3. PREGLED KORIŠĆENIH TEHNOLOGIJA

Prilikom odabira radnih okvira i tehnologija, postavljen je zadatak da rešenje mora biti bazirano na platformama otvorenog koda.

3.1. Spring radni okvir

Kao osnov za razvoj integracionog sloja servisa, koristi se Spring radni okvir. Spring je *Dependency Injection* radni okvir koji koristi princip inverzije kontrole (*Inversion of Control*) [5]. Pojam DI (*Dependency Injection*) uveo je Martin Fowler [6] i predstavlja mehanizam ubrizgavanja (ubacivanja) zavisnosti u neku klasu spolja. IoC (*Inversion of Control*) je naziv za tehniku gde se proces kreiranja i upravljanja zavisnostima neke klase izmešta iz nje same u spoljašnji proces. Kako se navodi u knjizi [5], pojmovi IoC i DI se mogu smatrati za sinonime kada je u

pitanju Spring radni okvir. Pored ove osnovne funkcije, Spring pruža infrastrukturu za razvoj Java baziranih poslovnih (*enterprise*) aplikacija. Uz pomoć njega se mogu kreirati kako desktop, tako i konzolne i *web*, odnosno JEE (*Java Enterprise Edition*) aplikacije. Kako je Spring radni okvir postao *defacto* standard u razvoju *enterprise* Java aplikacija, neće se detaljnije zalaziti u objašnjenje principa rada ovog radnog okvira. Spring radni okvir podržava više načina za pristup podacima u relacionim bazama podataka. Podržani su JDBC API, iBatis/MyBatis, Hibernate, JPA. Zbog podrške za rad sa nasleđenim (*legacy*) relacionim bazama podataka koje koriste složene (kompozitne) primarne ključeve u tabelama, kao i podrške za pozivanje poslovne logike koja se nalazi u uskladištenim procedurama, u ovom radu, koristiće se MyBatis radni okvir kao sloj za perzistenciju. Da bi se bolje shvatila pozicija MyBatis radnog okvira unutar Spring ekosistema, dat je grafički prikaz na slici 2.



Slika 2. Prikaz komponenti Spring radnog okvira

Podrška za MyBatis radni okvir nalazi se u delu Spring-a koji se bavi pristupom podacima i integracijom. MyBatis-u to omogućava da koristi Spring podršku za rukovanje transakcijama kao i da Core Container kreira MyBatis Mapper i Session objekte i ubrizgava ih u druge Java klase.

3.2. MyBatis

MyBatis 3 je radni okvir za perzistenciju objekata u bazu podataka sa podrškom za proizvoljne SQL iskaze, uskladištene procedure i kompleksna mapiranja [7]. Autor ovog radnog okvira je Clinton Begin i kreirao ga je 2001. godine pod nazivom iBatis. U literaturi [8] se još navodi da je to i radni okvir za mapiranje SQL iskaza na Java objekte. Takođe, autor samog radnog okvira ga u knjizi [9] definiše kao *data mapper*. Prema definiciji Martina Fowlera [10], *mapper* je objekat koji uspostavlja komunikaciju između dva nezavisna objekta. *Data mapper* Fowler [10] definiše kao sloj *mapper*-a koji prenosi podatke između objekata i baze podataka na takav način da ne stvara međusobne zavisnosti objekata, baze podataka i samog *mapper*-skog sloja. Za razliku od klasičnih ORM (*Object-Relational Mapping*) radnih okvira za objektno-relaciono mapiranje, MyBatis ne mapira Java objekte na tabele u relacionoj bazi podataka ili attribute klase na kolone tabela, nego mapira Java metode na SQL iskaze. Može se reći da MyBatis ima

hibridni pristup koji kombinuje JDBC i ORM [5] [9]. Zasnovan je na ideji da su ogromna sredstva uložena u razvoj relacionih baza podataka i SQL jezika i da oni za industriju softvera predstavljaju veliku vrednost. Čest je slučaj u praksi, da sama baza podataka i SQL iskazi korišćeni nad tom bazom, nadžive jednu ili više verzija aplikacije koje su pisane nad tom bazom. Dešava se i da se izvorni kod aplikacije u drugoj verziji napiše u nekom drugom programskog jeziku a da sama baza i SQL iskazi u velikoj meri ostanu isti. Zbog toga je mudro osloniti se na tu investiciju pa MyBatis ne pokušava da sakrije SQL, niti izbegava njegovu upotrebu. Ovaj radni okvir se oslanja na SQL i čini njegovu upotrebu jednostavnijom i omogućava lakšu integraciju u moderne objektno orjentisane razvojne platforme. Pored Java platforme, podržana je i .NET platforma kroz MyBatis.NET radni okvir koji predstavlja port MyBatis radnog okvira na Microsoft-ovu razvojnu platformu.

MyBatis 3 radni okvir je nastao kao grana u razvoju iBatis radnog okvira i predstavlja unapređenje postojećeg koncepta. Jedno od unapređenja se ogleda u tome da se sav kod može napisati u Javi, od upita, preko konfiguracionih klasa do mapper klasa. Kod iBatis-a se podacima pristupalo preko DAO (*Data Access Object pattern*) klasa dok se kod MyBatis-a koriste *Mapper* klase. DAO klase koriste mapirane iskaze dok su metode *Mapper* klase mapirani iskazi.

3.3. Castor

Castor je radni okvir otvorenog koda za vezivanje podataka (data binding) na Java platformi [11]. On omogućava serijalizaciju Java objekata u XML dokumenta i obrnuto kao i serijalizaciju Java objekata u relacione tabele baze podataka. Ukoliko već postoji Java objektni model, Castor XML mapiranje se može koristiti kao veza između XML dokumenta i postojećeg Java objektnog modela.

Castor omogućava programeru da specificira serijalizaciju i deserijalizaciju kroz *mapping* datoteku. Ova datoteka daje informacije Castor-u na kakav način su međusobno povezani dati XML dokument i odgovarajući skup Java objekata. Kako je Castor projekat nastao pre JAXB (*Java Architecture for XML Binding*) specifikacije, može se reći da je mapping datoteka ekvivalent *binding schema* XML datoteke u JAXB specifikaciji [12]. Takođe, maperska datoteka predstavlja dobar način da se izoluju promene u strukturi Java objektnog modela od promena formata odgovarajućeg XML dokumenta. Konkretno, to znači da se promene formata XML dokumenta ne odražavaju na Java objektni model, odnosno objektni model ne mora zbog toga da se menja. Ovo važi i za promene objektnog modela, koje ne uzrokuju promene u formatu XML dokumenta. Takođe, Castor podržava i podrazumevanu konverziju atributa objekata u XML tagove bez potrebe za maperskom datotekom. Ovo praktično znači da je moguće vezivanje Java objekata na XML dokumenta bez postojanja odgovarajuće XML *schema*-e.

Maperska datoteka je u suštini XML dokument koji opisuje kako se Java klase, metode i atributi trebaju transformisati u odgovarajuće elemente XML dokumenta i obrnuto. Konkretno, ako je definisan element *zap* unutar XML dokumenta, on se može deserijalizovati u Java

klasu *Zaposleni* dok se atribut *adresaPrebivalista* može serijalizovati u XML element *adr1*. Ključne klase Castor radnog okvira su *Marshaller*, *Unmarshaller* i *XMLContext*. *Marshaller* i *Unmarshaller* su klase zadužene za serijalizaciju i deserijalizaciju, respektivno. *XMLContext* je klasa koja služi za konfiguraciju samog Castor-a i instanciranje prve dve klase koje obavljaju konkretan posao.

3.4. Spring integracija sa Castor-om

Spring radni okvir, kako je prikazano na slici 2, u svom delu koji je zadužen za pristup podacima i integraciju sadrži i modul za *Object/XML* mapiranje ili skraćeno OXM. Ovaj modul je zadužen za serijalizaciju Java objekata u XML i deserijalizaciju XML-a u Java objekte koristeći neki od konkretnih radnih okvira poput Castor-a, JAXB-a, XMLBeans-a, JiBX-a ili XStream-a. XML može biti u formi DOM dokumenta, ulaznog ili izlaznog *stream*-a ili *SAX handler*-a. Poput samog Castor-a, unutar Spring OXM modula, centralno mesto zauzimaju dva interfejsa, *Marshaller* i *Unmarshaller*. Interfejs *Marshaller* je zadužen za serijalizaciju objektnog grafa u XML dok je *Unmarshaller* zadužen za deserijalizaciju XML dokumenata u Java objekte.

Najveća prednost korišćenja Spring OXM modula je konzistentna upotreba interfejsa. Ceo modul referencira dva glavna interfejsa, *Marshaller* i *Unmarshaller* što za posledicu ima jednostavnu zamenu konkretnog OXM radnog okvira kroz konfiguraciju aplikacije. Kod koji se bavi serijalizacijom i deserijalizacijom ostaje nepromenjen ili su potrebne minimalne izmene, najčešće u konfiguraciji. Čak je moguće koristiti kombinaciju OXM radnih okvira pa neke objekte serijalizovati koristeći Castor, a druge koristeći npr. XMLBeans, na neinvazivan način tj. bez izmene koda. To omogućava da se iskoriste sve prednosti određenog OXM radnog okvira u zavisnosti od slučaja korišćenja. Spring, takođe, omogućava i konzistentnu obradu izuzetaka tj. sve izuzetke konkretnih OXM radnih okvira prevodi u svoje izuzetke koji nasleđuju *XMLMappingException* baznu klasu pa kod biva nezavisan od konkretnih izuzetaka OXM radnih okvira.

4. IMPLEMENTACIJA SISTEMA

Postojeći informacioni sistem Fakulteta tehničkih nauka, skraćeno FTNIS, sadrži podatke koji su neophodni za izveštavanje ka ministarstvu finansija i za korišćenje elektronskih servisa ministarstva. Podaci unutar FTNIS-a se nalaze u formi tabela u relacionoj bazi podataka i opisuju entitete realnog poslovnog sistema. Ovi entiteti nisu identični entitetima koji figuriraju u poslovnim sistemima ministarstva finansija. Prvi korak u implementaciji servisnog sloja za integraciju je kreiranje objektnog modela entiteta koje prepoznaje ministarstvo. U drugom koraku uspostavlja se mapiranje tih objekata na SQL iskaze nad postojećim relacionim modelom baze podataka koristeći MyBatis i mapiranje objekata na specificirana XML dokumenta koristeći Castor. Treći korak predstavlja kreiranje Spring servisa i Spring MVC *controller*-a koji će uvezati sve komponente u funkcionalnu Spring baziranu aplikaciju. Arhitektura aplikacije data je u formi grafičkog prikaza na slici 3.

ANALIZA POGONSKIH ASPEKATA RADA POSTROJENJA NA ČVRSTU BIOMASU OPERATIONAL ASPECTS ANALYSIS OF SOLID BIOMASS-FUELLED HEAT PLANT

Stefan Predojević, Dragan Milićević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu se govori o biomasi kao energentu, načinu njenog dobijanja, kao i o tehnologiji postrojenja u kojima se upotrebljava biomasa. Akcenat rada je stavljen na analizu rada jednog tipičnog postrojenja koje kao gorivo koristi čvrstu biomasu.

Abstract – In this paper, biomass as an energy source, method of its creation and technology of plants which use biomass as a fuel is described. The accent was placed on the analysis of the work of a typical plant which uses solid biomass as a fuel.

Ključne reči: kotao, biomasa, energetika.

1. UVOD

Preduslov sveopšteg tehnološkog napretka jeste korišćenje raspoložive energije u industriji, domaćinstvima, trgovini, poslovanju, itd. Da bi se energija upotrebila mora se pretvoriti iz raspoloživih u upotrebljive oblike primenom odgovarajućih tehnoloških postrojenja. Raspoloživi oblici energije iz prirodnih izvora se mogu podeliti na [5]:

- Obnovljive - ubraja se energija dobijena iz vode, vetra, sunca, kao i iz biomase, itd.
- Neobnovljive – ubrajaju se fosilna goriva (ugalj, nafta, i prirodni gas), nuklearna goriva.

Prednost obnovljivih izvora se ogleda u tome što su količine neograničene, ako se sa njima na pravi način postupa. To podrazumeva da se u slučaju drvene biomase jednaka količina zasađuje i odstranjuje sa zemljišta, u suprotnom upotreba biomase nije ekološki opravdana. Sa druge strane količine neobnovljivih izvora su ograničene i njihovom upotrebom doprinosi se zagađenju životne sredine.

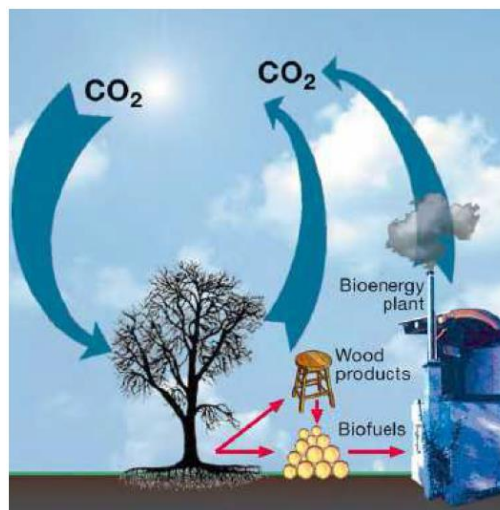
2. BIOMASA

2.1 Biomasa kao energent

Biomasa obuhvata biorazgradive materije biljnog i životinjskog porekla, dobija se preradom otpada i ostataka poljoprivredne proizvodnje, kao i iz šumarske proizvodnje. Može se naći u čvrstom, tečnom i gasovitom stanju [1].

Posmatrano sa ekološkog stanovišta, upotrebom biomase se postiže smanjenje emisije CO_2 , ne doprinosi se narušavanju životne sredine. Dolazi do slučaja kumulativne neutralnosti, emisija CO_2 pri sagorevanju je jednaka količini CO_2 koju je biljka potrošila u procesu fotosinteze tokom rasta. Princip kumulativne neutralnosti

je ilustrovan na slici 1. Dodatna prednost upotrebe biomase je korišćenje pepela (nusproizvod) kao đubriva, koje istovremeno doprinosi povećanju biljne mase. Sastav biomase je takav da u njemu nema sumpora, pa u gasovima prilikom sagorevanja nema sumporoksida, pritom usled niže temperature sagorevanja skoro da nema emisije azotnih oksida u produktima sagorevanja [1], [2].



Slika 1. Prikaz kumulativne neutralnosti biomase [1]

2.2. Tehnologija dobijanja i prerade biomase

U procesu pripreme biomase potrebno je utrošiti energiju. Ako se razmatra šumska biomasa, energija se troši na seču drva, pošumljavanje i rast šume. U nekim slučajevima se biomasa dobija kao nusproizvod, nema dodatnih utrošaka energije (npr. slama kao proizvod pšenice, kukuruzovina itd.). Na slici 2 dat je šematski prikaz podela čvrste biomase.

Iz razloga male energetske biomase vrednosti po jedinici mase, često se primenjuje prerada prirodnih pojava oblika biomase u prilagođeni sekundarni oblik. Time se olakšava transport, upotreba, kao i skladištenje.

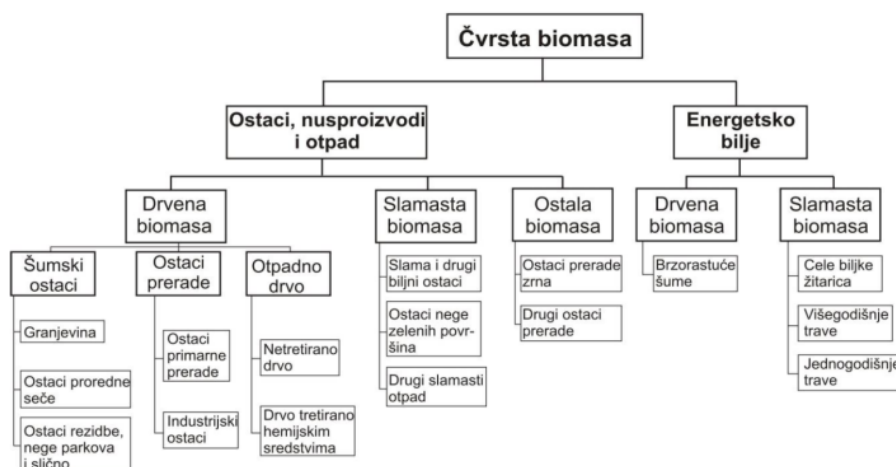
Tehnologije prerade biomase su sledeće:

- Zgrušnavanje,
- Usitnjavanje,
- Biohemijsko pretvaranje (anaerobno truljenje, fermentacija, esterifikacija),
- Termohemijsko pretvaranje (gasifikacija, piroliza).

U ovom radu se analizirala upotreba drvene biomase koja je dobijena procesom usitnjavanja. To je proces tokom koga se od ogrevnog drveta, krupnog i sitnog ostatka iz šume i ostataka iz procesa prerade drveta, drvena sečka dobija mašinskim usitnjavanjem.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Milićević, docent.



Slika 2. Šematski prikaz podele čvrste biomase [3]

Veličina drvene sečke je 3 – 5 cm u prečniku. Izgled formirane sečke prikazana je na slici 3.



Slika 3. Drvo u formi drvne sečke [2]

Drvna sečka je prikladna kao gorivo za velike sisteme, između ostalog zbog mogućnosti automatskog punjenja ložišta. Pored navedenog prednost sečke je i ta što se može koristiti kao gorivo sa visokim sadržajem vlage. Dodatni razlog upotrebe drvene sečke je i cena, koja je manja u odnosu na pelet.

3. TEHNOLOGIJE POSTROJENJA NA BIOMASU

Krajnji produkt energetskih sistema koji su projektovani na bazi termičke obrade biomase je energija u različitim oblicima. Oblik u kome će se energija isporučiti direktno zavisi od potreba potrošača i mogućnostima da iskoristi tu energiju na najbolji mogući način.

Trenutno zastupljeni energetski sistemi, na osnovu oblika krajnjeg produkta energije, mogu pripasti jednoj od navedenih [3]:

- sistemi za proizvodnju isključivo toplotne energije,
- sistemi za proizvodnju isključivo električne energije,
- sistemi za istovremenu proizvodnju električne i toplotne energije.

Na osnovu navedene podele, postrojenje koje je predmet ovog rada pripada grupi sistema za proizvodnju isključivo toplotne energije.

Sistemi ovog tipa, gde skoro sva toplotna energija biva distribuirana kroz mreže daljinskog grejanja, najbolje pokazuje prednosti u ekološkom smislu u odnosu na sisteme bazirane na fosilnim gorivima.

3.1 Proizvodnja energije termičkim tretmanom biomase

Metode za proizvodnju električne energije termičkim tretmanom biomase su sledeći:

- konvencionalni ciklus,
- gasifikacija i
- zajedničko spaljivanje (eng. co-firing).

U okviru konvencionalnog ciklusa, prilikom procesa sagorevanja, mogu se izdvojiti tri procesa, gde se samo u jednom proizvodi korisna toplota:

- Sušenje – na temperaturi do 150 °C, što u principu predstavlja isparavanje preostale vlage koja je ostala u biomasi,
- Piroliza do 600 °C - predstavlja cepanje i gasifikaciju materijala koji se spaljuje,
- Oksidacija - na temperaturi do 1300 °C, što predstavlja spaljivanje sa kiseonikom.

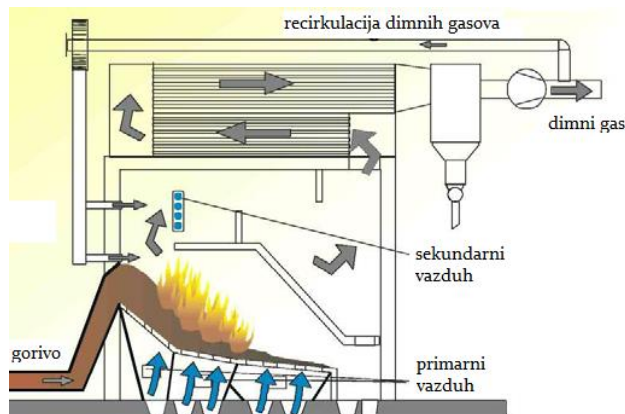
Po principu konstrukcije se razlikuju sledeća postrojenja:

- postrojenja sa rešetkom,
- postrojenja sa injeksionim spaljivanjem,
- postrojenja sa spaljivanjem u fluidiziranim sloju,
- specijalna postrojenja.

Kotao koji je upotrebljen u postrojenju koje je opisano u ovom radu radi na principu postrojenja sa pokretnom rešetkom. Izgled ovakvog postrojenja je dat na slici 4. Gorivo se sipa na rešetku, nakon toga dolazi do transportovanja biomase napred ka ložištu. Primarni vazduh se uvlači sa donje strane rešetke, prolazi kroz rešetku i vodi se u goreći sloj. Sekundarni vazduh se uduvava preko gorećeg sloja biomase. Nedostatak ovih postrojenja je taj što rade sa ograničenim opsegom granulacije goriva, u tom smislu su nefleksibilna [3].

Postrojenja sa rešetkom su pogodna za biomasu koja u sebi sadrži veliku koncentraciju vlage, promenljivu

veličinu čestica i visok sadržaj pepela. Moguće su i mešavine drvenih goriva, ali u zavisnosti od smjera dimnih gasova postoje ograničenja pri mešavini drveta i slame, žitarica zbog različitog ponašanja pri sagorevanju [4].



Slika 4. Postrojenje sa kosom rešetkom [2]

4. OPIS REALNOG SISTEMA

Na slici 5 se nalazi šematski prikaz postrojenja. Gorivo koje se nalazi u skladištu se transportuje sa kosim transporterom (M2) ka međubunkeru. Šiber na međubunkeru se otvara i zatvara prema potrebi, informacije o trenutnom položaju šibera obezbeđuju induktivni senzori GS2.1 i GS2.2. Komande za zatvaranje i otvaranje šibera se zadaju preko elektromagnetnih ventila (EMV) Y2.1 i Y2.2. Pored šibera na međubunkeru, motor M3 pokreće i rešetku. Biomasa se dalje transportuje sa pužnim transporterima (M4.1 i M4.2) do ložišta. Na rešetku koja se nalazi u ložištu se doprema biomasa, informacije o položaju rešetke se dobijaju sa induktivnih senzora GS3.1 i GS3.2, koji se aktiviraju u slučaju da se rešetka nalazi u krajnjem, početnom položaju, respektivno. Komande za kretanje rešetke nazad i napred se zadaju preko EMV Y3.1 i Y3.2, respektivno. Rad motora koji dopremaju gorivo do ložišta je uslovljen sa vrednosti temperature izlazne vode koja se meri sa Pt100 sondom (TT6). Na usisu goriva u ložište se nalaze kapilarni termostati TS1 i TS2, koji se aktiviraju u slučaju da temperatura konstrukcije u kojoj se nalaze pužni transporteri poraste preko 80°C. Impulsni vodommer (FS1) daje informaciju o tome da li ima potrebne vode za gašenje požara.

Motor M6, pokreće ventilator primarnog vazduha (VPV), održava temperaturu plamena u ložištu. Motor M8 pokreće ventilator recirkulacionog vazduha (VRV), koji takođe ima ulogu da održava temperaturu plamena u ložištu.

Motor M7 pokreće ventilator sekundarnog vazduha (VSV), brzina okretanja zavisi od koncentracije kiseonika u dimnim gasovima. Dodatno se sa klapnama kontroliše količina vazduha koja ulazi u ložište (PVV6.1, PVV6.2, RDGV8.1, RDGV8.2, O2V7.1, O2V7.2). U ložištu se nalaze grejači E1 i E2, uslov za njihov rad je rad VPV. Voda koja se zagreva u kotlu, pomoću recirkulacione pumpe (M10) kruži u sistemu.

Otpepeljivanje kotla se vrši motorom (M5), koji radi povremeno, u vremenskim intervalima. Senzori za

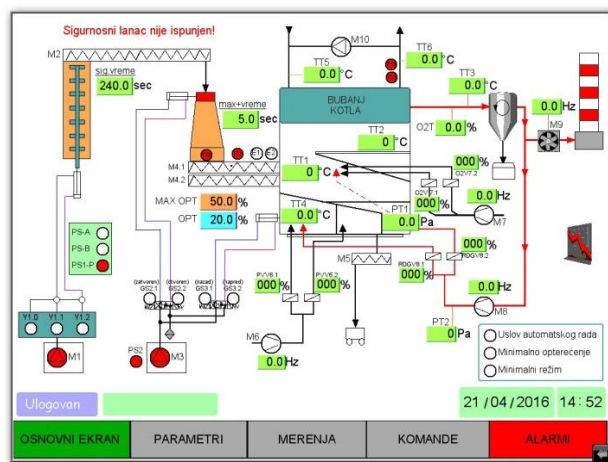
merenje temperature su tipa termopar (*eng. thermocouple*) TT1 (vrednost temperature iznad rešetke), TT2 (vrednost temperature iznad svoda). Dok su ostale sonde za merenje Pt100 tipa TT3, TT4, TT5, TT6. Sonda TT3 vrši merenje temperature dimnih gasova, TT4 vrši merenje temperature ispod rešetke, TT5 vrši merenje ulazne vode u kotao, a TT6 vrši merenje izlazne vode iz kotla. Pored otpornih termometara (vrše pretvaranje temperature u otpornost) se koristi i pasivni transponder koji ima svrhu konvertovanja vrednosti otpora u strujni signal opsega (4 – 20 ma). Senzor PT2 vrši merenje vrednosti pritiska u kanalu recirkulacionog vazduha.

Svi signali koji se dobijaju sa senzora i komande koje se izdaju se obrađuju sa PLC (*eng. programmable logic controller*) jedinicom. Za napajanje delova koji ne smeju ostati bez napajanja u slučaju ispada mrežnog napajanja koristi se uređaj besprekidnog napajanja (UPS *eng. uninterruptible power supply*). Ispravljač koji obezbeđuje napon za dolazne i odlazne signale sa senzora se napaja u slučaju nestanka mrežnog napajanja sa UPS-a.

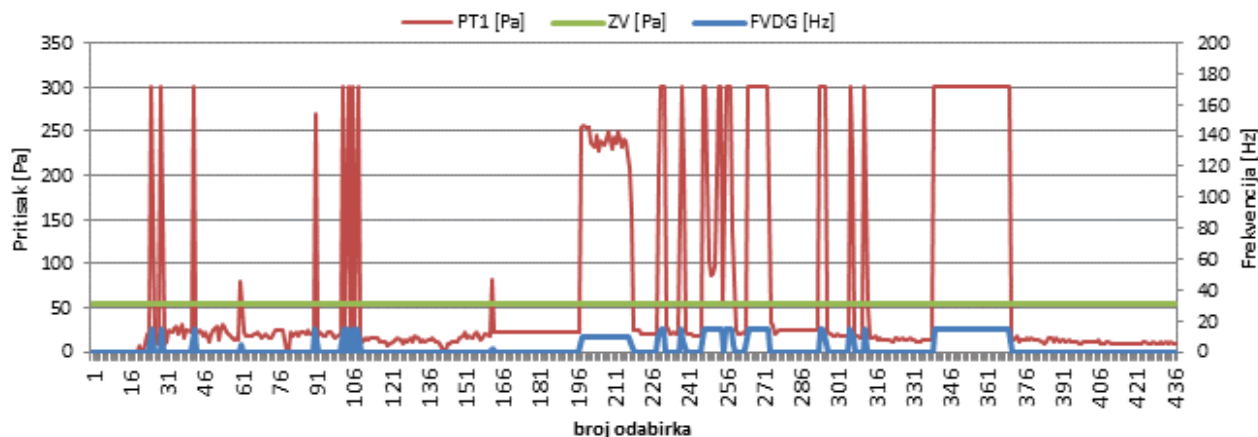
Motor M9 je motor koji pokreće ventilator dimnih gasova, senzor podpritiska PT1 vrši merenje pritiska u ložištu. Uloga vrentilatora je da održava vrednost pritiska u ložištu na vrednost od 54 Pa. U slučaju da pritisak poraste preko granice, kao što se može videti sa slike 6, u tim trenucima ventilator dimnih gasova se aktivira i vraća pritisak ispod vrednosti od 54 Pa. U svim trenucima u kojima dolazi do aktivacije ventilatora dimnih gasova se uspešno izvršila regulacija.

Vrednosti temperature u ložištu dobijene su sa senzora TT1 i TT2. Motori M4.1 i M4.2 su motori koji pokreću pužne transportere. Promenljiv rad pužnih transportera ima za cilj da obezbedi željenu vrednost temperature unutar ložišta korigovanjem količine goriva koja u ložište ulazi.

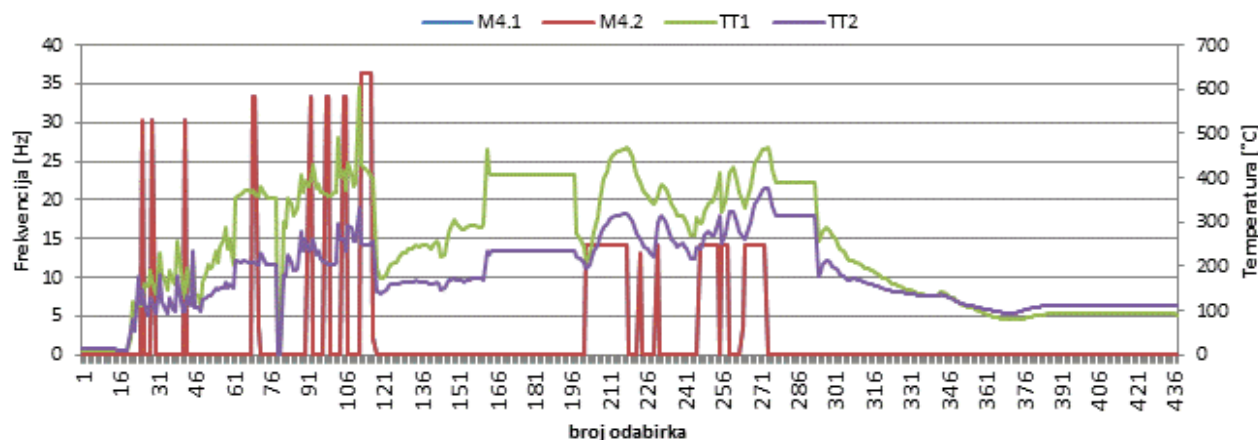
Sa slike 7 se vidi da u slučaju pada vrednosti temeperature dolazi do aktivacije pužnih dozatora, čime se određena količina goriva doprema do ložišta. Pužni dozatori se nalaze u neaktivnom stanju sve dok se ne desi propad vrednosti temperature.



Slika 5. Šematski prikaz postrojenja



Slika 6. Zavisnost frekvencije ventilatora dimnih gasova od podpritiska u ložištu



Slika 7. Rad pužnih transportera u zavisnosti od temperature unutar ložišta

5. ZAKLJUČAK

Rad pokazuje da sistem automatike kotlovskog postrojenja na čvrstu biomasu, shodno prikazanim graficima, uspeva da blagovremeno odreaguje na poremećaje. Svaki poremećaj u vidu promene bitnih parametara (pritisak, temperatura) unutar ložišta dovodi do adekvatnog odgovora sistema i njegovog svođenja u prethodno zadate okvire. Dalji razvoj ovog sistema se može odnositi na proširenja koja će omogućiti analizu ostalih parametara koji se mere. Time bi postojao uvid u stanje celokupnog sistema i stvarnu zavisnost preostalih parametara.

6. LITERATURA

- [1] G. H. "Obnovljivi izvori energije", Bečelov rad, Sveučilište Josipa Jurja Strossmayera u Osijeku, Osijek, 2015.
- [2] M. Brkić, T. Janić, G. Pejović, B. Kostić, "Raspoloživost i troškovi biomase za potrebe sistema daljinskog grejanja na području opština Vrbasa i Kule", Predstudija, Eko Produkt, Novi Sad, 2012.
- [3] M. Martinov, Đ. Đatkov, J. Krstić, G. Vujić, M. Golub, M. Brkić, S. Bojić, "Program za ocenu ekonomskih pokazatelja za energetska primenu biomase", Studija, Fakultet tehničkih nauka, 2011.

- [4] J. Vos, "Biomass Energy for Heating and Hot Water Supply in Belarus", Guide, University of Twente, 2005.
- [5] V. Katić, I. Kapetanović, N. Sarajlić, "Obnovljivi izvori električne energije", Fakultet tehničkih nauka, 2007.

Kratka biografija:



Stefan Predojević rođen je u Kragujevcu 1992. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Energetska elektronika i električne mašine odbranio je 2016. god.



Dragan Miličević rođen je u Tuzli 1977. god. Doktorirao je na Fakultetu tehničkih nauka 2014. godine. Oblast interesovanja su elektromotorni pogoni, energetska elektronika i obnovljivi izvori električne energije.

IMPLEMENTACIJA MEMORIJSKOG BUILT-IN SELF TEST KONTOLERA DESIGN IMPLEMENTATION OF A MEMORY BUILT-IN SELF TEST CONTROLLER

Andrea Erdeljan, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu su analizirani tipovi kvarova u memorijama i algoritmi za njihovo testiranje. Opisana je hardverska implementacija built-in self test kontrolera sa I2C interfejsom. Kontroler implementira nekoliko test algoritama i pogodan je za testiranje RAM memorija. Pored implementacije opisan je i postupak verifikacije koristeći UVM metodologiju. Dat je i primer moguće konekcije implementiranog jezgra i memorije.

Abstract – This paper analyses memory fault types and algorithms for testing those faults. A hardware implementation of a built-in self test controller is described. The controller has an I2C interface and implements several test algorithms. It is suitable for testing random access memories. Beside the design and implementation process, verification of the controller using UVM methodology is also presented.

Cljučne reči: built-in self test, DFT, VHDL, UVM, March, I2C

1. UVOD

Memorije su danas jedna od najčešće korišćenih komponenti. Skoro svi sistemi na čipu sadrže bar neki tip ugrađene memorije, bila ona SRAM, DRAM, ROM ili flash memorija. Sa napretkom tehnologije dolazi i do porasta kompleksnosti samih memorija, njihovog kapaciteta, frekvencije takta i gustine. Kvalitet ugrađene memorije je kritičan za kvalitet čitavog sistema zbog čega testiranje memorije postaje neophodno. Međutim klasičnim funkcionalnim testiranjem je jako teško efikasno istestirati čitavu memoriju. Zbog toga postaje postaje neophodno razviti jeftino i efikasno rešenje za njihovo testiranje. Built-in self test metoda je jedno od rešenja ovog problema. Stavljanjem tester kola na sam čip se značajno ubrzava proces testiranja i smanjuje njegova cena.

Realizovani kontroler implementira četiri test algoritma i nekoliko načina rada kako bi korisnik mogao da u potpunosti prilagodi kontroler svojim potrebama. Interfejs jezgra je standardni I2C za koji su potrebne samo dve fizičke linije. Kontroler poseduje i self-repair mogućnosti kako bi se, u slučaju pronalaska kvara u memoriji, isti mogao zamaskirati. Pored razvoja RTL-a samog kontrolera, razvijeno je i kompletno verifikaciono okruženje. Odlikuje ga potpuna randomizacija procesa testiranja i automatska provera ispravnosti što u velikoj meri olakšava proces verifikacije.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Ivan Mezei, docent.

Rad je podeljen u nekoliko celina. Na početku rada analizirana je problematika kvarova u memorijama i potreba za njihovim brzim i jednostavnim otkrivanjem. Predstavljeni su najčešće korišćeni algoritmi za testiranje. Diskutovane su prednosti i mane korišćenja built-in self testova i prikazan je postupak implementacije kontrolera. Zatim je prikazan i razvoj verifikacionog okruženja koristeći univerzalnu verifikacionu metodologiju (UVM). Po završetku navedenih procesa ukratko je prikazan način povezivanja projektovanog kontrolera sa instancom memorije.

2. KVAROVI U MEMORIJAMA

U ovom poglavlju je dat pregled kvarova koji se javljaju u memorijama. Dat je njihov opis i potrebni koraci za njihovo testiranje.

Stuck-at kvar se javlja kada je vrednost memorijske ćelije uvek 0 (SA0) ili uvek 1 (SA1).

Transition kvarovi se pojavljuju se na ćelijama koje ne mogu učiniti prelaz sa nule na jedinicu (*up transition fault*) ili sa jedinice na nulu (*down transition fault*). *Transition* kvarovi manifestuju *stuck-at* ponašanje nakon pokušaja lošeg prelaza.

Coupling kvarovi se javljaju kada upis u jednu ćeliju menja sadržaj druge ćelije. Postoji više vrsta ovih kvarova. Među njima su: *inversion*, *idempotent*, *bridging* i *state coupling* kvarovi koji se razlikuju u uzroku i načinu menjanja druge ćelije. Kod *inversion* kvarova prelaz $0 \rightarrow 1$ ili $1 \rightarrow 0$ jedne ćelije invertuje vrednost u drugoj ćeliji. *Idempotent* kvarovi se manifestuju kada prelaz $0 \rightarrow 1$ ili $1 \rightarrow 0$ jedne ćelije forsira vrednost (ili 0 ili 1) u drugoj ćeliji, dok kod *bridging* kvarova postoji električna veza između ćelija. Kod ovih kvarova grešku uzrokuje logička vrednost, a ne prelaz u ćeliji. U zavisnosti od logičke vrednosti *bridge*-a kvarovi mogu biti ili AND kvarovi ili OR kvarovi. Poslednja vrsta su *state* kvarovi gde stanje jedne ćelije forsira vrednost (ili 0 ili 1) u drugoj ćeliji.

Najrazličitiji i najteži za testiranje su *neighborhood pattern sensitive* kvarovi. Javljaju kada vrednost ćelije ili sposobnost ćelije da menja vrednost zavisi od vrednosti neke susedne ćelije. U zavisnosti od ponašanja dele se na aktivne, pasivne i statične.

3. TEST ALGORITMI

Test algoritam je konačna sekvenca test elemenata. Test element sadrži broj memorijskih operacija (upis ili čitanje), vrednost podatka i adrese. Glavni ciljevi pri razvoju ovih algoritama su što viša pokrivenost kvarova i što kraće vreme izvršavanja. Nažalost ovi zahtevi su u suprotnosti, pa se u praksi često vrši kompromis. Biraju

se testovi koji ne garantuju detekciju svih kvarova, ali imaju prihvatljivo kratko vreme izvršavanja. Testovi se mogu podeliti u dve grupe: klasični i March algoritmi.

3.1. Klasični test algoritmi

Klasični algoritmi uglavnom ili imaju jako dobru pokrivenost kvarova uz jako dugačko vreme izvršavanja ili je situacija obrnuta. Imaju nisku pokrivenost, a jako su brzi. Zbog ovoga se ovi algoritmi sve ređe koriste. Primer iz prve grupe je *checkerboard* algoritam: detektuje samo *stuck-at* kvarove i eventualno *transition* kvarove (u zavisnosti od implementacije), ali je zato kompleksnosti $O(n)$. Za razliku od njega „GALPAT“ algoritam detektuje sve *stuck-at*, *transition* i *coupling* kvarove, ali mu je kompleksnost $O(n^2)$, što je neprihvatljivo za built-in self test.

BIST kontroler implementira *checkerboard* algoritam. Ukoliko se memorija zamisli kao šahovska tabla, gde svaka lokacija predstavlja jedno polje *checkerboard* algoritma se može opisati sledećim pseudo-kodom:

1. upisati podatak u sve lokacije obojene crno, a invertovani podatak u sve bele lokacije
2. pročitati celu memoriju
3. upisati invertovani podatak u sve lokacije obojene crno, a podatak u sve bele lokacije
4. pročitati celu memoriju

3.2. March test algoritmi

March grupa algoritama dovoljno dobro zadovoljava oba kriterijuma: imaju dobru pokrivenost kvarova uz relativno kratko vreme izvršavanja. Zbog toga su postali najčešće korišćeni algoritmi za testiranje memorije. Oni mogu detektovati *stuck-at*, *transition* i *coupling* kvarove, pri čemu im je kompleksnost $O(n)$. U ovom radu su implementirana tri March algoritma. Sledi njihov opis.

March X algoritam je jedan od najčešće korišćenih March algoritama. Pseudo-kod je dat niže.

1. for addr in 0 to n - 1:
 - 1.1. upiši podatak u A[addr]
2. for addr in 0 to n-1:
 - 2.1. pročitaj A[addr] i uporedi sa podatkom (kvar ukoliko se ne slažu)
 - 2.2. upiši invertovani podatak u A[addr]
3. for addr in n-1 to 0:
 - 3.1. pročitaj A[addr] i uporedi sa invertovanim podatkom (kvar ukoliko se ne slažu)
 - 3.2. upiši podatak u A[addr]
4. for addr in 0 to n-1:
 - 4.1. pročitaj A[addr] i uporedi sa podatkom (kvar ukoliko se ne slažu)

March C- algoritam je poboljšanje March C algoritma. On je optimalan algoritam jer nema redundantnih operacija. Pseudo-kod je:

1. for addr in 0 to n - 1:
 - 1.1. upiši podatak u A[addr]
2. for addr in 0 to n-1:
 - 2.1. pročitaj A[addr] i uporedi sa podatkom (kvar ukoliko se ne slažu)
 - 2.2. upiši invertovani podatak u A[addr]
3. for addr in 0 to n-1:

- 3.1. pročitaj A[addr] i uporedi sa invertovanim podatkom (kvar ukoliko se ne slažu)
- 3.2. upiši podatak u A[addr]
4. for addr in n-1 to 0:
 - 4.1. pročitaj A[addr] i uporedi sa podatkom (kvar ukoliko se ne slažu)
 - 4.2. upiši invertovani podatak u A[addr]
5. for addr in n-1 to 0:
 - 5.1. pročitaj A[addr] i uporedi sa invertovanim podatkom (kvar ukoliko se ne slažu)
 - 5.2. upiši podatak u A[addr]
6. for addr in 0 to n-1:
 - 6.1. pročitaj A[addr] i uporedi sa podatkom (kvar ukoliko se ne slažu)

Poslednji implementirani algoritam je March Y algoritam. Njegov pseudo-kod je:

1. for addr in 0 to n - 1:
 - 1.1. upiši podatak u A[addr]
2. for addr in 0 to n-1:
 - 2.1. pročitaj A[addr] i uporedi sa podatkom (kvar ukoliko se ne slažu)
 - 2.2. upiši invertovani podatak u A[addr]
 - 2.3. pročitaj A[addr] i uporedi sa invertovanim podatkom (kvar ukoliko se ne slažu)
3. for addr in n-1 to 0:
 - 3.1. pročitaj A[addr] i uporedi sa invertovanim podatkom (kvar ukoliko se ne slažu)
 - 3.2. upiši podatak u A[addr]
 - 3.2. pročitaj A[addr] i uporedi sa podatkom (kvar ukoliko se ne slažu)
4. for addr in 0 to n-1:
 - 4.1. pročitaj A[addr] i uporedi sa podatkom (kvar ukoliko se ne slažu)

4. BUILT-IN SELF TEST

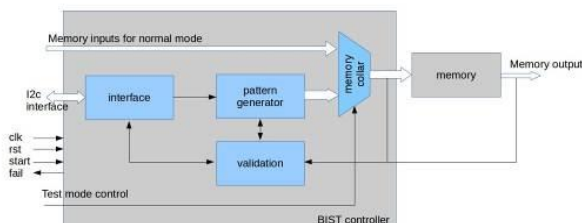
Built-in self test (ili skraćeno BIST) je tehnika dizajniranja dodatnog hardvera kako bi kola mogla da testiraju svoju funkcionalnost, čime se smanjuje zavisnost od spoljašnje test opreme (ATE - *Automated Test Equipment*). Stavljanjem tester kola na sam čip se značajno ubrzava proces testiranja i smanjuje njegova cena. Glavni pristup pri projektovanju memorijskog BIST-a je dovajanje kontrolera oko memorije koje omogućava pristup samoj memoriji i povećava kontrolabilnost i obzervabilnost. Time se takođe povećava ponovna upotreba IP-a. Ukoliko na čipu postoji više memorija jedan kontroler se može koristiti za testiranje svih čime se smanjuje dodatni prostor. Testiranje memorija bi se uvek trebalo vršiti istom ili većom brzinom od one koja će se koristiti u redovnoj upotrebi. BIST ovo omogućuje.

Međutim, ova metoda ima i svoje mane. Korišćenje BIST-a znači dodavanje kola koje je nepotrebno za normalan rad memorije. Za male memorije može biti veoma neefikasno ukoliko je kontroler veći od same memorije. Još jedna mana je manja fleksibilnost nego kod direktnog testiranja. Iako postoji veliki broj test algoritama oni moraju biti implementirani u samom kontroleru i ne mogu se naknadno dodavati ili menjati

nakon što se kontroler ugradi (sem redizajniranjem i ponovnim ugrađivanjem kontrolera). Dok pri direktnom testiranju ipak postoji veća sloboda pri izboru testova i načinu samog testiranja.

5. IMPLEMENTACIJA

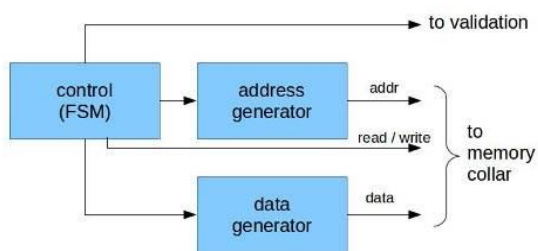
U ovom poglavlju je predstavljena hardverska implementacija memorijskog built-in self test kontrolera. Na slici 1 prikazan je blok dijagram implementacije.



Slika 1. Blok dijagram BIST kontrolera

Sa slike vidimo da je kontroler podjeljen u nekoliko komponenti: interfejs, memorijski omotač, generator adresa i podataka i deo za validaciju.

Kao što je već napomenuto, kontroler implementira standardni I2C interfejs. Preko ovog interfejsa pristupa se registarkoj banci. Registri se mogu podeliti u dve grupe: kontrolni i statusni. Kontrolni registri određuju način rada kontrolera. Omogućavaju podešavanja algoritama kao i odabir i podešavanja različitih funkcionalnosti kontrolera (odabir algoritma, kašenje pri upisu i čitanju, način završetka testa, opseg adresa, vrednost podataka, ...). Statusni registri daju informacije o rezultatima testa, opisuju pronađene kvarove ukoliko ih ima i daju informacije o uspešnosti popravke memorije. Postoji nekoliko statusnih registara. Oni čuvaju vrednosti adrese u kojima postoji kvar, vrednost pročitane podatka, broj pronađenih kvarova... U statusnim registrima se takođe vodi računa o *self-repair* mehanizmu. Ukoliko se pronađe kvar u memoriji lokacija se pamti u registru i koristi se jedna od rezervnih lokacija da bi se zamenila kvarna.



Slika 2. Pattern generator

U generatorskoj komponenti se vrši implementacija samih algoritama. Na slici 2 je prikazan blok dijagram. Vidimo da se sastoji od kontrolnog dela, adresnog generatora i generatora podataka. Prilikom implementacije generatora korišćena je tehnika projektovanja digitalnih sistema kao FSM. Mašina stanja kontroliše rad blokova i vodi računa o izvršavanju algoritama. Takođe šalje informacije bloku za validaciju, preko signala *valid_o*, kako bi se

podaci proveravali u pravom trenutku, kao i informaciju u završetku testa. Adresni generator u zavisnosti od trenutnog stanja i algoritma generiše potrebne adrese, dok se u generatoru podataka, uz informaciju o vrednosti osnovnog podatka iz registra, vrši manipulacija i generisanje potrebnih podataka.

Zadatak bloka za validaciju je nadgledanje izlaznih podataka memorije, poređenje sa očekivanim vrednostima i reagovanje na pronađeni kvar. Reakcija na kvar obuhvata postavljanje statusnih registara, drajvovanje FAIL izlazne linije i pokušaj popravke memorije ukoliko je to moguće.

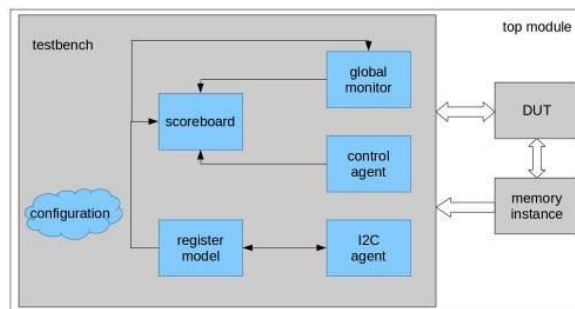
BIST kontroler mora podržavati dva načina rada: normalni i test. U normalnom režimu treba omogućiti normalno ponašanje memorije, odnosno da postojanje BIST kontrolera nikako ne utiče na rad memorije. Dok u test režimu kontroler preuzima kontrolu nad memorijom i drajvuje podatke i adresu na njene ulaze. Ovo je zadatak memorijskog omotača. Memorijski omotač je implementiran kao multiplexer koji će na osnovu ulazne linije TEST_MODE izabrati da li se na interfejs memorije prosleđuju signali koje drajvuje kontroler ili se prosleđuju spoljašnji signali.

6. VERIFIKACIJA

Funkcionalna verifikacija je rađena prateći standardizovanu univerzalnu verifikacionu metodologiju kao bi se osigurala detaljna verifikacija i laka ponovna upotreba okruženja.

6.1. Verifikacione komponente

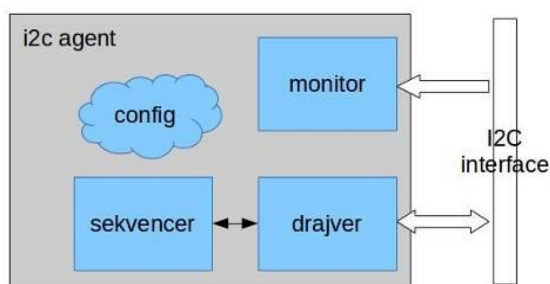
Na slici 3 vidimo komponente realizovanog okruženja.



Slika 3. Verifikaciono okruženje

Testbenč se sastoji od i2c agenta, kontrolnog agenta, globalnog monitora, *scoreboard* komponente i registarskog sloja sa svim pratećim komponentama. Sva podešavanja u testbenču se vrše iz konfiguracione klase. U top modulu je izvršeno povezivanje testbenča sa BIST kontrolerom kao dizajnom koji se testira (DUT) i instancom memorije koju kontroler testira.

I2C i kontrolni agent u potpunosti prate metodologiju i sadrže sve potrebne podkomponente - drajver, sekvencer i monitor. Podešavanja agenta se vrše preko konfiguracione klase. Na slici 4 prikazan je I2C agent. Kontrolni agent je zadužen za generisanje i nadgledanje kontrolnih signala START, FAIL i TEST_MODE.



Slika 4. I2C agent

Registarski sloj omogućava praćenje stanja u registrima dizajna i lak pristup registrima i memoriji. Apstrahuje operacije čitanja i upisa u registre i time olakšava pisanje testova i verifikaciju dizajna. Komponente registarskog sloja su registarski blok sa svim registrima, adapter i prediktor. Registar blok obuhvata sve registre u okruženju i kreira jedinstvenu mapu registara sa definisanim adresama i nazivima. Sam registar je opisan registarskim modelom koji sadrži detaljne informacije o samom registru (polja, veličinu, reset vrednosti, ...). Adapter se koristi za konverziju apstraktnih registarskih sekvenci u konkretne, kompatibilne sekvence koje se zatim izvršavaju na master sekvenceru tj. drajveru, ali i za konverziju u obrnutom smeru kako bi se transakcije koje prikupi monitor mogle iskoristiti za ažuriranje registarskog modela da bi isti ostao verna kopija registara u dizajnu (veza sa prediktorom). Kako bi registri u modelu bili adekvatno ažurirani, potrebno je koristiti prediktor. Povezan je sa monitorom koji mu, nakon prikupljene transakcije, šalje informacije na osnovu kojih se ažuriraju registri u modelu. Ovim se omogućava adekvatno praćenje stanja registara u dizajnu. Prediktor je povezan sa adapterom kako bi se omogućila konverzija, kao i sa registarskim modelom kome šalje potrebne informacije

Globalni monitor je zadužen za nadgledanje signala ka memoriji, kontrolu protokola, kao i sakupljanje paketa koje zatim šalje *scoreboard*-u na kontrolu.

Komponenta *scoreboard* vrši proveru ispravnosti algoritama. Na osnovu stanja u registrima i informacija iz globalnog monitora implementira istu logiku kao dizajn, ali na višem nivou apstrakcije. Nakon primanja podataka iz globalnog monitora vrši proveru i javlja grešku ukoliko dođe do neslaganja. Još jedna uloga *scoreboard*-a je da na osnovu sakupljenih informacija ažurira statusne registre.

6.2. Proces verifikacije

Nakon razvoja čitavog okruženja pristupilo se procesu verifikacije. Zbog funkcionalnosti dizajna, potrebno je razviti i model memorijske komponente nad kojom dizajn vrši testiranje.

Koristeći ModelSim simulator vršilo se umetanje grešaka u memoriji kako bi se simulirali kvarovi i testiralo reagovanje kontrolera na pronalazak istih. Za to se koristila naredba *signal_force*, pri čemu su lokacije sa greškom, broj grešaka i vrednost bile nasumično generisane u cilju što detaljnije verifikacije BIST kontrolera.

Tokom testa su upisuju nasumično izabrane vrednosti u registre i startuje izvršavanje. Pošto je sva provera automatska, potrebno je samo pustiti odgovarajući test i nakon završetka tumačiti rezultate.

7. ZAKLJUČAK

U ovom radu predstavljen je proces implementacije i verifikacije memorijskog built-in self test kontrolera. Objasnjena je potreba za ovakvim kontrolerom i dat primer njegovog korišćenja.

Razvijeni kontroler odlikuje visok stepen konfigurabilnosti – od podrške za razne osobine memorije koju testira, do različitih vrsta algoritama i načina generisanja podataka.

U cilju poboljšanja implementiranog IP jezgra moguće je dalje optimizovati implementaciju radi smanjenja resursa, ali i dodavanje novih algoritama ukoliko se ukaže potreba za njima.

8. LITERATURA

- [1] Allen C. Cheng, "Comprehensive Study on designing Memory BIST: Algorithms, implementations and Trade-offs", EECS 579, University of Michigan Fall 2002
- [2] Wang, Wu, Wen, "VLSI Test Principles and Architectures, Ch 8, Morgan Kaufmann, 2006
- [3] P. Bannen, "Testing Word Oriented Embedded RAMs Using Built-in Self Test", University of Twente

Kratka biografija:



Andrea Erdeljan rođena je 3.8.1992. u Kikindi, Srbija. Završila je gimnaziju "Dušan Vasiljev", prirodni smer, u Kikindi. Osnovne akademske studije je završila na Fakultetu tehničkih nauka 2015. god i nakon toga upisala master studije na istom fakultetu.

**PRIMENA KOMPRIMUJUĆEG ODABIRANJA ZA DETEKCIJU KORISNIKA
U BEŽIČNIM KOMUNIKACIJAMA****USER ACTIVITY DETECTION IN WIRELESS COMMUNICATIONS
USING COMPRESSED SENSING**Veljko Boljanović, *Fakultet tehničkih nauka, Novi Sad***Oblast - ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj - U radu je opisana osnovna ideja komprimujućeg odabiranja, a zatim je problem detekcije aktivnih korisnika smešten u kontekst komprimujućeg odabiranja. Posmatran je jednostavan scenario u bežičnom sistemu u kojem je svaki od korisnika aktivan sa malom verovatnoćom. Za detekciju korisnika, korišćen je poznati algoritam OMP (engl. Orthogonal Matching Pursuit), a njegove performanse iskazane su u odnosu na jednostavni SUD (engl. Single User Detection) algoritam.

Abstract - In this paper, the basic idea of compressed sensing is described. In addition, the problem of user activity detection is put in the context of compressed sensing. We consider a simple wireless system in which users are active with low probability. The well-known OMP algorithm is used for user activity detection and its performance is presented and compared with that of the simple SUD algorithm.

Ključne reči: komprimujuće odabiranje, detekcija, OMP, SUD, bežične komunikacije

1. UVOD

Slučajni pristup medijumu predstavlja dobru alternativu za tradicionalne tehnike (TDMA, FDMA, CDMA) za upravljanje potrošivim resursima. Ukoliko pretpostavimo da svaki od korisnika može biti aktivan sa malom verovatnoćom, jasno je da će u bilo kojem trenutku prosečan broj aktivnih korisnika biti mali. Ova osobina može se iskoristiti na efikasan način ukoliko se ceo problem postavi u kontekst komprimujućeg odabiranja. Time se omogućava prijemu, tj. baznoj stanici, da napravi broj "merjenja" koji je manji od ukupnog broja korisnika. Pokazuje se da je smanjeni broj merjenja dovoljan kako bi se odredilo koji korisnici su aktivni. Kako bi ceo postupak dobio pun smisao, potrebno je da postoje efikasni detekcioni algoritmi koji bi omogućili da se iz rezultatnog vektora merjenja otkriju aktivni korisnici. Postoje brojni detekcioni algoritmi, a u ovom radu korišćen je OMP koji je čest predmet istraživanja i unapređivanja. U sistemima bez šuma, pokazuje se da OMP radi bez greške nakon određenog broja merjenja m . Sa druge strane, kada u sistemu postoji šum, OMP ne garantuje da će idealno detektovati aktivne korisnike. Takođe, šum doprinosi i povećanju granice za potreban broj merjenja m . Performanse OMP algoritma najbolje se vide kada se on uporedi sa jednostavnim detekcionim

algoritmom, kao što je SUD. Glavni parametar poređenja jeste broj merjenja, i pokazuje se da je za istu verovatnoću greške, OMP zahteva manji broj merjenja m .

2. KOMPRIMUJUĆE ODABIRANJE

Komprimujuće odabiranje predstavlja tehniku vršenja manjeg broja merjenja (odabiranja) signala, dovoljnih za rekonstrukciju tog signala. Ukoliko je proces merjenja signala linearan, tada se komprimujuće odabiranje i rekonstrukcija signala svode na rešavanje niza linearnih jednačina [1]. Sa matematičkog stanovišta, veza između rezultatnog vektora $y \in \mathbb{C}^m$ i originalnog vektora $x \in \mathbb{C}^N$ može se predstaviti na sledeći način:

$$Ax = y \quad (1)$$

gde matrica $A \in \mathbb{C}^{m \times N}$ modeluje proces merjenja. Ako je a_{ij} element matrice A , x_j element vektora x , y_i element vektora y , pri čemu $i \in [0, m]$ i $j \in [0, N]$, tada se sistem linearnih jednačina može napisati na sledeći način:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1N}x_N &= y_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2N}x_N &= y_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \dots + a_{3N}x_N &= y_3 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \dots + a_{mN}x_N &= y_m \end{aligned}$$

Sa klasičnog matematičkog stanovišta, potrebno je da broj jednačina bude veći ili jednak broju nepoznatih kako bi sistem bio rešiv. To znači da bi m trebalo biti bar jednako sa N kako sistem ne bi bio neodređen. Ukoliko postoji bar jedno rešenje u neodređenom sistemu, tada je ukupan broj rešenja beskonačno velik. Ipak, pokazuju se da je moguće da važi $m < N$, a da pritom sistem jednačina bude rešiv. Da bi ovako nešto bilo moguće, potrebno je da bude ispunjen jedan uslov: vektor x mora biti "redak" (engl. sparse). Redak vektor je onaj kod kojeg je većina elemenata jednaka nuli. Dolazi se do zaključka da je određeni signal moguće kompresovati pomoću komprimujućeg odabiranja, tj. predstaviti ga pomoću vektora manje dužine, ukoliko je taj signal redak.

Kada bi broj merjenja m nekog retkog signala bio jednak sa dužinom tog signala N , tada bi se moglo reći da su merjenja vršena na neefikasan način. U procesu formiranja odabiraka, odnosno formiranja sistema linearnih jednačina, većina nepoznatih u sistemu bila bi odbačena zbog toga što je većina elemenata vektora x jednaka nuli. Stoga je opravdano nad retkim signalima napraviti broj merjenja koji je manji od N .

NAPOMENA:

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

Upotreba komprimujućeg odabiranja nameće dva pitanja [1]:

- Kakva matrica $A \in \mathbb{C}^{m \times N}$ treba da se koristi u samom procesu?
- Da li postoji efikasan detekcioni algoritam koji nam može omogućiti rekonstrukciju vektora x iz izraza (1)?

Matrica $A \in \mathbb{C}^{m \times N}$ mora biti definisana tako da se može obezbediti rekonstrukcija svakog vektora x . Vrlo čest i pogodan način za definisanje matrice A jeste korišćenje slučajnih matrica (engl. random matrices). Ove matrice dobijaju se generisanjem elemenata na osnovu određene raspodele. Na primer, Bernulijeve matrice su matrice čiji elementi predstavljaju nezavisne slučajne promenljive koje mogu imati vrednost +1 ili -1, sa jednakom verovatnoćom. Drugi primer slučajnih matrica jeste Gausova matrica. To je matrica čiji su elementi nezavisne slučajne promenljive sa sa Gausovom normalnom raspodelom [2]. U ovom radu koristi se upravo Gausova matrica, i to takva da je očekivana vrednost norme svake od kolona matrice jednaka jedinici.

Postoje razni detekcioni algoritmi, a u ovom radu korišćen je pohlepni iterativni OMP algoritam. Njegove performanse upoređene su sa jednostavnim detekcionim SUD algoritmom.

3. DETEKCIONI ALGORITMI

Komprimujuće odabiranje ne bi imalo smisla ukoliko bi bilo nemoguće rekonstruisati originalni vektor. Različiti detekcioni algoritmi daju pun smisao celom postupku, omogućavajući rekonstrukciju vektora.

Ponekad prilikom primene komprimujućeg odabiranja nije od velike važnosti tačna rekonstrukcija vrednosti originalnog vektora (signala), nego je bitno samo odrediti na kojim pozicijama se nalaze nenulti elementi u vektoru. Na primer, traženje nenulatih elemenata potrebno je prilikom detekcije aktivnih korisnika u bežičnim komunikacijama.

3.1. OMP algoritam

OMP algoritam opisan je od strane Y. C. Pati, R. Rezaifar i P. S. Krishnaprasad u radu [4], ali i od strane J. A. Tropp i A. C. Gilbert u [5]. OMP predstavlja pohlepni iterativni algoritam koji u svakom koraku pravi trenutno optimalnu odluku, sa nadom da će na kraju doći do optimalnog rešenja. Ovaj algoritam donosi odluke na osnovu sličnosti između reziduala i njegovih projekcija na kolone matrice A .

Neka važi (1) i neka je $x \in \mathbb{R}^N$ originalni vektor, $A \in \mathbb{R}^{m \times N}$ matrica koja modeluje proces merenja, a $y \in \mathbb{R}^m$ rezultatni vektor. Uzimanjem skupa realnih brojeva $\mathbb{R}$ ne gubi se na opštosti. Vektori a_i predstavljaju kolone matrice A . Bez gubitka opštosti, može se usvojiti da važi $\|a_i\|_2 = 1$ za svako $i \in [1, N]$. Takođe, usvojicemo oznake za aktuelni skup različitih indeksa kolona S_{j-1} i indeks selektovane kolone Ω_j , gde je j redni broj iteracije.

Svaka iteracija OMP algoritma, sastoji se iz dva glavna dela: selektovanje određene kolone matrice A i promena reziduala. U prvom delu se trenutni rezidual r_{j-1} projektuje na sve kolone matrice A , uz određivanje

indeksa kolone sa kojom rezidual ima najveću meru sličnosti (korelaciju). Izabrani indeks kolone Ω_j dodaje se aktuelnom skupu S_{j-1} . Nakon toga, vrši se procena elemenata vektora x na osnovu svih trenutno odabranih kolona. Drugi deo OMP algoritma odnosi se na promenu reziduala r_{j-1} i to na osnovu procenjenih vrednosti elemenata vektora x . S obzirom na način na koji se rezidual menja, novi rezidual je ortogonalan na sve prethodno odabrane kolone.

OMP algoritam može se sumirati na sledeći način:

- 1) Inicijalizuj.
 $r_0 = y$
 $S_0 = \{\emptyset\}$
 $j = 1$
- 2) Identifikuj indeks najznačajnije kolone i proširi skup S .
 $\Omega_j = \operatorname{argmax}_i |\langle r_{j-1}, a_i \rangle|$
 $1 \leq i \leq N$
 $S_j = S_{j-1} \cup \{\Omega_j\}$
- 3) Proceni koeficijente koji aproksimiraju y , koristeći sve selektovane kolone.
 $\hat{x}_j = \operatorname{argmin}_x \|y - A_{S_j} x\|_2$
- 4) Osveži r_j i j .
 $r_j = y - A_{S_j} \hat{x}_j$
 Inkrementiraj j .
- 5) Proveri uslov zaustavljanja.
 Ako uslov nije ispunjen, skoči na korak 2).
 Ako je uslov ispunjen, vrati vektor x u kojem su nenulte vrednosti na pozicijama Ω dobijene iz poslednje procene $\hat{x}_j$, pri čemu je $\Omega \in S_j$. Ostale vrednosti u vektoru x jednake su nuli.

OMP algoritam iterira sve dok se ne ispunji uslov zaustavljanja. Ovaj uslov može se odnositi na maksimalan broj iteracija, preostalu energiju u rezidualu, ili drugo. Vrlo često se koristi uslov da je norma reziduala manja od određene granice $\varepsilon > 0$, tj. da važi $\|r_j\|_2 < \varepsilon$.

3.2. SUD algoritam

SUD algoritam spada u grupu algoritama koji za potrebe detekcije nenulatih elemenata vektora x postavljaju određeni prag, na osnovu kojeg se odlučuje koje kolone matrice A će biti selektovane. Rekonstrukcija vektora x , tj. aproksimacija vrednosti elemenata, može se uraditi minimizacijom kvadratne greške.

Postoje različiti pristupi koji se koriste prilikom postavljanja praga. Najjednostavniji pristup jeste postavljanje praga korelacije na određenu fiksnu vrednost μ . Sve kolone čija je korelacija sa vektorom y veća od μ bivaju selektovane, dok se ostale zanemaruju. Algoritam sa ovakvim pristupom naziva se detekcija pojedinih korisnika (SUD). Naziv "korisnik" ekvivalentan je nazivu "element vektora x ".

Korelacija ρ_i između kolone a_i i vektora y može se izračunati na sledeći način:

$$\rho_i = \frac{|a_i' y|^2}{\|a_i\|_2^2 \|y\|_2^2} \quad (2)$$

pri čemu je $i \in [1, N]$. Sve korelacije se porede sa pragom i biraju one koje su veće od μ .

SUD algoritam može se sumirati na sledeći način:

- 1) Inicijalizuj.
 $S_0 = \{\emptyset\}$
- 2) Za svako $i = 1, \dots, N$, izračunaj korelaciju ρ_i i uporedi je sa pragom μ . Izaberi indekse onih kolona koje imaju korelacije veće od μ .

$$\rho_i = \frac{|a_i' y|^2}{\|a_i\|_2^2 \|y\|_2^2}$$
Ako $\rho_i > \mu$: $S_i = S_{i-1} \cup \{i\}$, inače $S_i = S_{i-1}$.

SUD algoritam je jednostavniji od OMP algoritma, ali to ne znači da je on dobra alternativa za OMP. Definisanje ovog algoritma ima za cilj da pokaže koliko je iterativni pohlepni algoritam OMP zapravo efikasan.

4, FORMULACIJA PROBLEMA

Scenarij koji je korišćen inspirisan je radom [3], čiji su autori Alyson K. Fletcher, Sundeep Rangan i Vivek K. Goyal. U takvom scenariju, više korisnika komunicira sa jednom prijemnom stanicom, pri čemu šalju jednostavne kontrolne poruke koje sadrže informaciju o tome da li su aktivni u određenom trenutku. Ovakav tip komunikacije često se sreće u bežičnim komunikacijama. Na primer, u ćelijskim sistemima korisnici komuniciraju sa baznom stanicom pomoću sličnih poruka, sa ciljem da ostvare inicijalni pristup ili da pošalju zahtev za slanje.

Scenarij koji se posmatra u radu postavljen je u kontekst komprimujućeg odabiranja. To znači da bi prijemnik na osnovu određenog broja merenja trebalo da detektuje koji korisnici su aktivni, pri čemu je broj merenja manji od ukupnog broja korisnika. Postupak detektovanja vrši se pomoću različitih detekcionih algoritama, a poruke koje su korisnici poslali mogu se aproksimirati korišćenjem l_2 minimizacije. Ipak, ono što je najbitnije u ovom jednostavnom scenariju jeste detekcija aktivnih korisnika.

Pretpostavimo da N korisnika koristi isti bežični kanal prema jednom prijemniku. Svakom korisniku j dodeljena je kodna reč $a_j \in \mathbb{C}^m$, koja zapravo predstavlja vektor dužine m .

Na prijemnoj strani, signal svakog od korisnika ima oblik $x_j a_j$, gde je x_j kompleksni skalar. Ako korisnik nije aktivan važi $x_j = 0$. Sa druge strane, ako je korisnik aktivan x_j predstavlja proizvod simbola koji korisnik prenosi i slabljenja koje unosi kanal. Ukupan signal na prijemnoj strani ima oblik kao izraz (1):

$$y = Ax$$

gde $A \in \mathbb{C}^{m \times N}$ predstavlja matricu čije su kolone vektori a_j , $j = 1, \dots, N$. Budući da se vektori a_j nazivaju kodnim rečima, matrica A naziva se rečnikom. Vektor $x \in \mathbb{C}^N$ naziva se modulirani vektor s obzirom na to da njegovi elementi predstavljaju proizvod moduliranih simbola i slabljenja u kanalu.

Pošto je poznat vektor x , moguće je definisati i skup aktivnih korisnika:

$$S_{true} = \{j : x_j \neq 0\}. \quad (3)$$

Moguće je posmatrati vektor x kao deterministički ili slučajni vektor. Kod determinističkog vektora, poznato je koji korisnici su aktivni, tj. koji elementi vektora x su različiti od nule. Sa druge strane, kod slučajnog vektora, svaki od korisnika može biti aktivan sa određenom verovatnoćom. Zbog toga je zgodno uvesti veličinu λ koja se naziva verovatnoća aktivnosti i za koju važi $\lambda \in (0, 1)$.

Detekcioni algoritmi koji se koriste u radu ne zahtevaju prethodno znanje o moduliranim simbolima. Ovakav pristup je i poželjan jer uglavnom nije poznato slabljenje koje unosi kanal. Opravdano je pretpostaviti da ne postoji informacija o slabljenju, pre svega zbog vrste komunikacije između korisnika i prijemnika - ne postoji pilot sekvenca koja najavljuje komunikaciju i koja može ispitati slabljenje u kanalu.

Ukoliko se za aktivnog korisnika j definiše snaga moduliranog simbola kao $p_j = |x_j|^2$, tada se deterministički skup $\{p_j\}_{j=1}^N$ može nazvati profil snage (engl. power profile). U ovom radu koristi se specifični profil snage u kojem je snaga moduliranog simbola p_j jednaka za sve korisnike. Izjednačavanje snage je rađeno iako u sistemu ne postoji šum.

Elementi matrice A predstavljaju nezavisne slučajne promenljive sa istom raspodelom: $\mathcal{CN}(0, 1/m)$. Može se lako dokazati da je u tom slučaju očekivana varijansa bilo koje od kolona matrice A jednaka jedinici.

Veoma važna pretpostavka u postavci problema jeste da su transmisije svih korisnika idealno sinhronizovane. Drugim rečima, na prijemnoj strani poznato je barem vreme propagacije svakog od korisnika. Takođe, pretpostavlja se da ne postoji višestruka propagacija (engl. multipath).

5. NUMERIČKA SIMULACIJA

Prilikom numeričke simulacije treba uzeti u obzir da su teorijski rezultati dobijeni ispitivanjem problema u asimptotskoj analizi. U takvom pristupu, podrazumeva se da razne veličine, poput dužine vektora, teže beskonačnosti. Međutim, u praksi najčešće nije moguće ispitati da li su idealni teorijski rezultati tačni jer je nemoguće obezbediti beskonačne dužine ili vrednosti pojedinih veličina. Zbog toga se u praksi često koristi relaksiraniji pristup u kojem se dozvoljava da se sa određenom verovatnoćom greške odstupa od idealnih teorijskih rezultata. Na primer, u [1], idealan prag korelacije μ SUD algoritma dobijen je pod pretpostavkom da su vektori beskonačne dužine. Taj idealni prag omogućuje idealnu detekciju skupa aktivnih korisnika. Ipak, u praksi nije moguće koristiti vektore beskonačne dužine, stoga je pogodnije prag korelacije odrediti na osnovu verovatnoće da je neaktivan korisnik detektovan kao aktivan. Ova verovatnoća se na engleskom naziva *false alarm probability*. Za SUD algoritam, ova verovatnoća može biti definisana kao:

$$p_{FA} = Pr(j \in \hat{S} \mid j \notin S_{true}) = Pr(\rho_j > \mu \mid j \notin S_{true}) \quad (4)$$

U [3] je pokazano da se prag korelacije μ može izračunati na sledeći način:

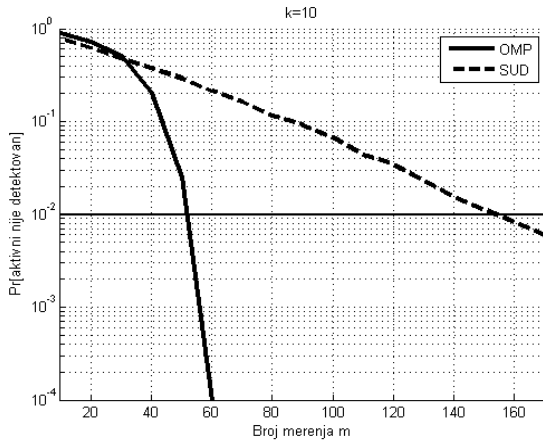
$$\mu = -\log(p_{FA})/m \quad (5)$$

za određenu p_{FA} . U ovom radu korišćena je vrednost $p_{FA} = 10^{-3}$. Pri ovoj fiksnoj vrednosti za p_{FA} , merena je verovatnoća da aktivni korisnik ne bude detektovan kao aktivan. Ova verovatnoća u engleskom se naziva *missed detection rate*, i data je pomoću izraza:

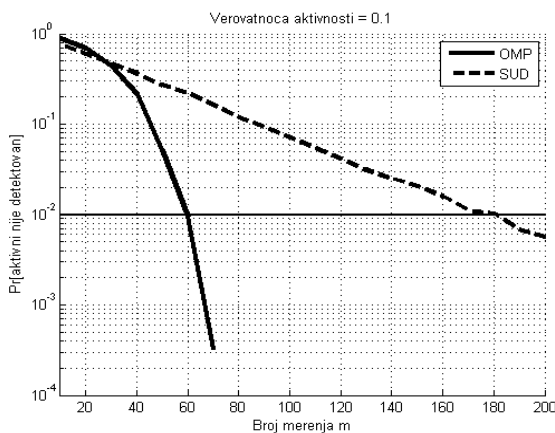
$$p_{MD} = Pr(j \notin \hat{S} \mid j \in S_{true}). \quad (6)$$

Verovatnoća da aktivni korisnik nije detektovan uprosečena je za sve $j \in S_{true}$.

Analiza je vršena nad $N = 100$ korisnika, i to na dva načina: sa determinističkim vektorom x kod kojeg ima $k = 10$ aktivnih korisnika, i sa slučajnim vektorom x kod kojeg je verovatnoća aktivnosti korisnika $\lambda = 0.1$. Oba puta korišćena je Monte Karlo metoda sa 1000 pokušaja za određivanje verovatnoće da aktivni korisnik nije detektovan. Rezultati su prikazani grafički na slikama 1 i 2, i na njima se mogu uporediti rezultati za OMP i SUD algoritme.



Slika 1 - Potreban broj merenja m za model sa $N = 100$ korisnika, od kojih je slučajnih $k = 10$ aktivno, pri čemu $SNR \rightarrow \infty$.



Slika 2 - Potreban broj merenja m za model sa $N = 100$ korisnika, sa verovatnoćom aktivnosti $\lambda = 0.1$, pri čemu $SNR \rightarrow \infty$.

6. ZAKLJUČAK

Ukoliko je broj aktivnih korisnika značajno manji od ukupnog broja korisnika, tada je opravdano postaviti ceo problem u kontekst komprimujućeg odabiranja. Na taj način, mogu se detektovati svi aktivni korisnici, a pritom se može koristiti broj merenja koji je manji od ukupnog broja korisnika. Pri samom odabiranju, potrebno je voditi računa o konstrukciji matrice A , a često se kao rešenja uzimaju matrice čiji su elementi slučajne promenljive.

Pokazano je da komprimujuće odabiranje dobija pun smisao ukoliko postoji dovoljno efikasan detekcioni algoritam koji se može primeniti. U radu je korišćen iterativni pohlepni algoritam OMP i pokazana je njegova efikasnost u poređenju sa SUD algoritmom. Glavni kriterijum za poređenje bio je potrebn broj merenja da se postigne određena verovatnoća greške i ispostavilo se da OMP zahteva daleko manji broj merenja od SUD algoritma. Ovakav zaključak može se izvesti bez obzira na to da li je vektor x deterministički ili slučajan. Ipak, može se primetiti da OMP algoritam zahteva manji broj merenja m u situaciji kada je vektor x deterministički u poređenju sa situacijom kada je vektor x slučajan.

7. LITERATURA

- [1] Simon Foucart i Holger Rauhut, "A Mathematical Introduction to Compressive Sensing", Springer Science+Business Media, New York 2013
- [2] Holger Rauhut, "Compressive Sensing and Structured Random Matrices", June 2013
- [3] Alyson K. Fletcher, Sundee Rangan i Vivek K Goyal, "On-Off Random Access Channels: A Compressed Sensing Framework", arXiv:0903.1022
- [4] Y. C. Pati, R. Rezaiifar i P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decompositions", Proc. 27th Asilomar Conf. on Sig., Sys. and Comp., 1993.
- [5] J. A. Tropp i A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit", IEEE Trans. Inform. Th., Vol. 53, No. 12, pp. 4655–4666, Dec. 2007

Kratka biografija:



Veljko Boljanović rođen je u Zrenjaninu 1992. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Telekomunikacioni sistemi odbranio je 2016. godine.

IP JEZGRO ZA OBRADU PIKSELA I NJEGOVA IMPLEMENTACIJA NA ZEDBOARD PLATFORMI**IMPLEMENTATION OF PIXEL SHADER IP CORE ON ZEDBOARD PLATFORM**Vladimir Vrbaški, *Fakultet tehničkih nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj – U ovom radu predstavljen je VHDL dizajn jedinice za obradu piksela unutar grafičkog procesora, a takođe je opisana aplikacija, koja je razvijena u C programskom jeziku. Osnovna funkcionalnost ovako projektovanog dizajna jeste primena programskog fajla, koji sadrži proizvoljan broj instrukcija, koje određuju izgled slike na ekranu monitora. Kompletan dizajn je implementiran na PSoC (programmable system on chip) čipu, tako da se C aplikacija izvršava na ARM procesoru, dok je FPGA (field-programmable gate array) deo tog čipa konfigurisan prema VHDL dizajnu. Implementacijom na stvarnom hardveru potvrđena je ispravnost rada celokupnog sistema.

Abstract – This paper proposes the VHDL design of pixel shader unit from GPU. The application developed in C programming language has also been described. The main functionality of this design is the application of programming file, which consists of any number of instructions, which determine picture displayed on the monitor screen. Complete design is implemented in Programmable system on chip (PSoC). The C application is running on the ARM processor, while the FPGA part of that chip is configured by VHDL design. Operation correctness is confirmed by implementation on the real hardware.

Ključne reči: Grafički processor, obrada piksela, VGA, VHDL, FPGA, SoC, C

1. UVOD

Tema ovog rada jeste projektovanje *pixel shader core*-a, koji predstavlja deo svakog standardnog *graphics processor unit* (GPU) *core*-a, i implementacija dizajna na *field-programmable gate array* (FPGA) čipu [1].

Osnovna funkcionalnost projektovanog *pixel shader core*-a jeste primena programskog fajla sa proizvoljnim brojem asemblerskih instrukcija, pomoću kojih se određuje slika na ekranu monitora, koja se iscrtava putem standardizovanog *video graphics array* (VGA) interfejsa [2].

Jezik u kojem je izvršeno modelovanje dizajna je VHDL (engl. *VHSIC hardware description language*) [3]. On predstavlja standardizovani jezik za opis hardvera, a njegova glavna prednost je u tome što omogućava verifikaciju, odnosno simulaciju dizajna, pre sinteze na stvarnom hardveru.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vuk Vranjković.

Prilikom modelovanja, korišćen je *register-transfer level* (RTL) nivo apstrakcije [4], gde se modeluju sinhrona digitalna kola, tako što se nad digitalnim signalima (podacima) primenjuju logičke operacije, i zatim se ti podaci upisuju u registre (koji najčešće predstavljaju D flip-flobove).

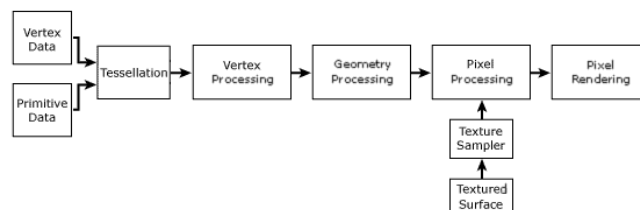
Dizajn je implementiran na čipu Zynq7000 [5] na ploči ZedBoard [6] kompanije Xilinx. Ovaj čip predstavlja *All Programmable system on chip* (PSoC). Glavna karakteristika ove familije čipova je da se u njima nalazi ARM procesor (*processing system* – PS), a takođe i konfigurabilni deo (*programmable logic* – PL), koji predstavlja FPGA arhitekturu.

Interfejs, koji služi za komunikaciju između PS i PL dela na čipu Zynq7000 je *advanced extensible interface* (AXI). AXI protokol je nastao u okviru *advanced microcontroller bus architecture* (AMBA) specifikacije [7], koju je razvila kompanija ARM.

U drugom poglavlju dat je kratak opis GPU *core*-a, koji je deo svake standardne grafičke kartice. U trećem poglavlju opisan je dizajn; svi podblokovi od kojih se sastoji, kao i njihov međusobni interfejs. Četvrto poglavlje sadrži prikaz performansi, pomoću kojih se može odrediti efikasnost dizajna. U petom poglavlju dat je opis blok dizajna, koji predstavlja instancioniranje i povezivanje IP *core*-ova, dok je u šestom poglavlju opisan softverski deo. Sedmo poglavlje prikazuje implementaciju dizajna, koja potvrđuje njegovu ispravnost. Na kraju, u osmom poglavlju dat je zaključak; šta je urađeno, i kako se dalje može unapređivati i koristiti ovaj *pixel shader core*.

2. ANALIZA PROBLEMA

Pixel shader core, koji je projektovan u okviru ovog rada, predstavlja krajnji deo protočne obrade (engl. *pipeline*) unutar svakog standardnog grafičkog procesora (GPU *core*-a). Neki od delova protočne obrade su: *vertex shader* i *geometry shader*, što se može videti na slici 1 [8]. Ta dva koraka protočne obrade vrše konverziju 3D modela u 2D sliku, dok *pixel shader core* izračunava boje pojedinih piksela te slike, koja se zatim iscrtava na ekranu monitora.

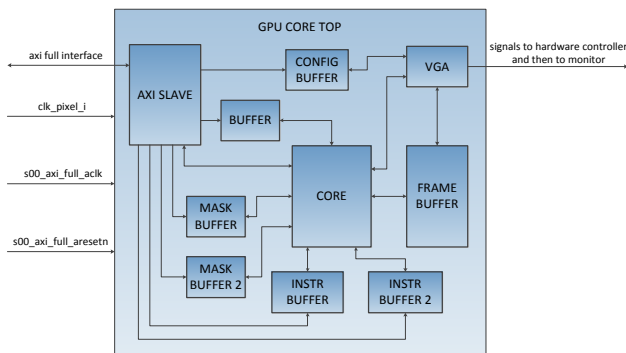
Slika 1. Struktura GPU *core*-a (preuzeto iz [8])

Za realizaciju *pixel shader core*-a najčešće se koristi *single instruction, multiple data* (SIMD) arhitektura, gde se jedna instrukcija primenjuje na velikoj količini podataka (piksela). U ovom dizajnu je korišćena upravo takva arhitektura. Tipičan *pixel shader core*-a se sastoji od:

1. Registra za prihvatanje instrukcije
2. Veći broj ALU (*arithmetic logic unit*) jedinica (tipično od 8 do 64)
3. Memorije za smeštanje obrađenih piksela

3. OPIS PODBLOKOVA DIZAJNA

Blok šema kompletnog dizajna na najvišem nivou data je na slici 2. Ulazi i izlazi dizajna mogu se videti na toj slici, i oni su povezani preko blok dijagrama sa drugim pomoćnim IP *core*-ovima, i sa ARM procesorom na Zynq7000 čipu (videti poglavlje 5).



Slika 2. Blok šema kompletnog dizajna sa unutrašnjom strukturom

Ulazi kompletnog dizajna su:

1. s00_axi_full_aclk, koji predstavlja *clock* signal po kome radi GPU *core*
2. clk_pixel_i, koji predstavlja *clock* signal po kome radi VGA kontroler
3. s00_axi_full_aresetn, koji se dovodi na GPU *core* i AXI *Full slave* uređaj; ovaj signal je aktivan na '0' i tada drži sve registre u GPU *core*-u i AXI *Full slave*-u u početnom stanju
4. s00_axi_full interfejs signali, koji se dovode na AXI *Full slave* uređaj, sa AXI *Full master* uređaja, koji predstavlja ARM procesor na Zynq7000 čipu

Izlazi kompletnog dizajna su:

1. signali, koji se vode sa VGA kontrolera na hardverski kontroler za VGA signal, a zatim sa njega na VGA konektor i na monitor; to su:
 - a. hsync_o, koji predstavlja horizontalnu sinhronizaciju slike na monitoru; kada je na '0', tada monitor počinje iscertavanje piksela u sledećoj horizontalnoj liniji
 - b. vsync_o, koji predstavlja vertikalnu sinhronizaciju slike na monitoru; kada je na '0', tada monitor počinje iscertavanje novog *frame*-a od gornjeg levog ugla
 - c. rgb_o[11:0], koji se vodi na hardverski kontroler, i predstavlja piksel koji se trenutno iscertava na monitoru; menja se na svaku rastuću ivicu clk_pixel_i signala, što predstavlja brzinu iscertavanja piksela u svakoj horizontalnoj liniji

2. s00_axi_full interfejs signali, koji se vode sa AXI *Full slave*-a na AXI *Full master* uređaj

Kompletnan dizajn sadrži osam različitih podblokova (entiteta), i oni su:

1. VGA kontroler (VGA), koji piksele slike pročitane iz *frame* bafera, prosleđuje na izlaz rgb_o[23:0], pritom pazeći na horizontalnu i vertikalnu sinhronizaciju slike na monitoru (hsync_o i vsync_o)
2. Bafer za sliku (BUFFER) je *dual port* blok RAM memorija (*random-access memory*) veličine 2048 lokacija x 192 bita, u koju se upisuje slika, koja će se prikazati na ekranu monitora
3. Baferi za instrukcije (INSTR BUFFER i INSTR BUFFER 2) su *dual port* blok RAM memorije veličine 32 lokacije x 32 bita, u koje se upisuju instrukcije koje je potrebno primeniti nad slikom
4. Baferi za maske (MASK BUFFER i MASK BUFFER 2) su *dual port* blok RAM memorije veličine 2048 lokacije x 256 bita, u koje se upisuju maske za sve 32 instrukcije koje se primenjuju nad slikom
5. Bafer za rezoluciju (CONFIG BUFFER) je *dual port* blok RAM memorija, veličine 8 lokacija x 32 bita, u koju se upisuju podaci o rezoluciji, koji određuju veličinu i položaj slike na ekranu
6. *Frame* bafer (FRAME BUFFER) je *dual port* blok RAM memorija, veličine 4096 lokacija x 192 bita, u kojoj se nalaze dve kopije slike; nad jednom se primenjuju instrukcije, dok se druga prikazuje na ekranu
7. GPU *core* (CORE), koji predstavlja centralni podblok, u kome se nalazi mašina konačnih stanja, koja je odgovorna za upravljanje radom kompletnog dizajna, a takođe, u tom bloku se nalazi i 8 ALU jedinica, u kojima se izvršavaju zadate instrukcije
8. AXI *Full slave* kontroler (AXI SLAVE), koji predstavlja *slave* uređaj u AXI *Full* komunikaciji, i koji je odgovoran za upisivanje piksela slike, podataka o rezoluciji, maski i instrukcija u odgovarajuće bafer; potrebno je on bude povezan preko AXI interfejsa na AXI *Full master* kontroler, koji će inicirati prenos slike, podataka o rezoluciji, maski i instrukcija, tako što će slati odgovarajuće adrese i podatke

4. PERFORMANSE DIZAJNA

Performanse projektovanog dizajna izražene su vremenom potrebnim za izvršavanje jedne ALU instrukcije nad celom slikom.

Formula, po kojoj je moguće veoma približno tačno odrediti vreme, za koje će se izvršiti jedna instrukcija nad celom slikom, glasi:

$$t = T \frac{P}{N}, \quad (1)$$

gde je T perioda *clock* signala, po kome radi *core* u dizajnu, P je veličina slike u baferu za sliku u pikselima, dok je N broj ALU jedinica.

U simulaciji je izmereno da potrebno vreme za izvršavanje jedne instrukcije nad celom slikom iznosi $t = 20,51$ us. To vreme je izmereno za sledeće vrednosti: N =

8, $P = 16384$, i $T = 10$ ns. Kada se te vrednosti uvrste u prethodnu formulu, dobije se $t = 20,48$ us, što se veoma približno slaže sa vremenom izmerenim u simulaciji.

Te tri vrednosti, prema kojima je izmereno vreme u simulaciji, predstavljaju upravo i vrednosti, koje važe i u implementiranom dizajnu.

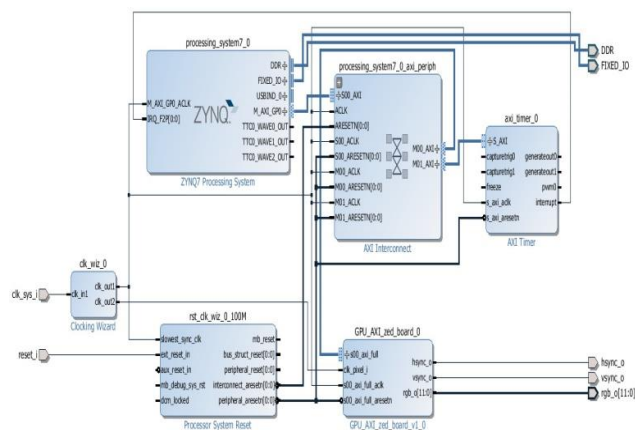
5. BLOK DIZAJN SISTEMA

Kompletan dizajn, opisan u trećem poglavlju, zapakovan je kao IP core GPU_AXI_zed_board_v1_0 (slika 3), pomoću alata za kreiranje i pakovanje IP core-ova u alatu „Vivado“ [9].



Slika 3. Blok šema zapakovanog core-a GPU_AXI_zed_board_0

Blok šema zapakovanog IP core-a je prikazana na slici 3. Prilikom pakovanja, zadati su određeni generički parametri, koji se mogu menjati pre implementacije, i tako se taj IP core može prilagoditi nekom drugom FPGA čipu, ili nekoj drugoj ploči, ako je to potrebno.



Slika 4. Slika blok dizajna iz alata „Vivado“ kompanije Xilinx

Na slici 4 dat je kompletan blok dizajn, u kome su povezani odgovarajući IP core-ovi, koji su potrebni da bi dizajn ispravno funkcionisao. Ulazi i izlazi blok dizajna se tačno mapiraju na odgovarajuće ulaze i izlaze FPGA čipa na ZedBoard ploči. To mapiranje se vrši pomoću XDC fajla, koji je takođe napisan.

IP core-ovi u blok dizajnu su:

1. GPU_AXI_zed_board_0, koji predstavlja dizajn, koji je projektovan u ovom projektu, a opisan je detaljno u trećem poglavlju
2. processing_system7_0, koji predstavlja jedan ARM procesor na ploči ZedBoard (na toj ploči postoje dva procesora), koji se ponaša kao AXI Full master uređaj

3. processing_system7_0_axi_periph predstavlja AXI interconnect, preko koga se povezuje AXI interfejs od AXI Full master-a (processing_system7_0) ka AXI Full slave-u (GPU_AXI_zed_board_0)
4. rst_clk_wiz_0_100M predstavlja IP core za generisanje reset signala za IP core-ove: GPU_AXI_zed_board_0, processing_system7_0_axi_periph, i axi_timer_0; ulaz je signal reset_i, koji je mapiran na jedan taster na ploči ZedBoard
5. clk_wiz_0 predstavlja IP core, koji je odgovoran za generisanje clock signala za sve ostale IP core-ove u blok dizajnu; ulazni signal je clk_sys_i frekvencije 100MHz, a izlazni signali su: clk_out1 frekvencije 100MHz, i clk_out2 frekvencije 66,67MHz
6. axi_timer_0 je IP core, koji predstavlja hardverski tajmer, i služi za generisanje vremenskog kašnjenja pomoću prekida (engl. interrupt); vrednost kašnjenja se upisuje od strane processing_system7_0, preko AXI interfejsa, a vrednost se određuje softverski, i promenljiva je

6. SOFTVER SISTEMA

Aplikacija je pisana u C programskom jeziku, i razvijana je u alatu SDK (engl. Software Development Kit), koji je deo alata „Vivado“.

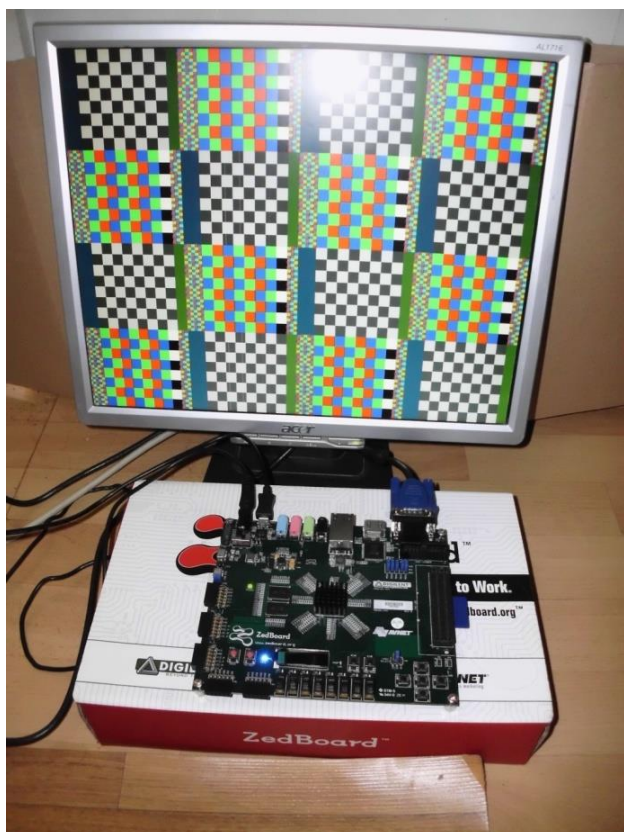
Fajlovi, koje korisnik mora specificirati, kako bi koristio aplikaciju, smeštaju se na SD karticu, koja se priključuje na odgovarajući konektor na ZedBoard ploči. Postoje tri tipa fajla, koje korisnik mora specificirati, i smestiti ih na SD karticu, i to su:

1. programski fajlovi, što su TXT fajlovi, koji sadrže zapisane zahteve za slanje slika, instrukcija, podataka o rezoluciji, ili maski, u vidu asemblerskih instrukcija, koje parsira aplikacija; ti zahtevi se zadaju i izvršavaju sekvencijalno (jedan zahtev za drugim); takođe, moguće je izvršiti samo jedan takav program prilikom jednog pokretanja aplikacije
2. fajlovi za slike, što su BMP fajlovi, koji sadrže sliku rezolucije tačno one, koja može da stane u bafer za sliku (u opisu, koji je dat u trećem poglavlju, rezolucija je 128 x 128 piksela); kada aplikacija dobije zahtev za slanje te slike, ta slika se šalje preko AXI interfejsa, i upisuje u bafer za sliku
3. fajlovi za maske, što su BMP fajlovi, u kojima su sakrivene maske za sve instrukcije, koje se nalaze u baferu za instrukcije (u opisu, koji je dat u trećem poglavlju, maksimalan broj instrukcija je 32); kada aplikacija dobije zahtev za slanje tih maski, te maske se šalju preko AXI interfejsa, i upisuju u odgovarajući bafer za maske

7. IMPLEMENTACIJA

Implementacijom na stvarnom hardveru potvrđena je ispravnost rada celokupnog sistema. Pokretanjem aplikacije za nekoliko napisanih programskih fajlova, potvrđeno je da i harverski, i softverski deo rade ispravno.

Slika na ekranu monitora menja se tačno prema zahtevima, koji su zapisani asemblerskim instrukcijama u tim programima.



Slika 5. Slika ploče ZedBoard na kojoj je implementiran dizajn i monitora na kome se iscrta slika

Na slici 5 može se videti implementacija projektovanog sistema na hardveru. Zynq7000 čip na ploči ZedBoard je konfigurisan prema opisanom blok dizajnu (slika 4). Zatim je pokrenuta aplikacija na ARM procesoru na toj ploči, koja iščitava određeni programski fajl. ZedBoard ploča je priključena na monitor, na kome se iscrtava obrađena slika.

8. ZAKLJUČAK

U ovom radu, projektovana je jedinica za obradu piksela unutar standardnog GPU core-a, a zatim i implementirana na ploči ZedBoard kompanije Xilinx. Dizajn je modelovan u VHDL jeziku, i zatim zapakovan u IP core, pomoću alata za kreiranje i pakovanje IP core-ova, u alatu „Vivado“ kompanije Xilinx. Nakon toga, razvijena je aplikacija, u C programskom jeziku, koja se izvršava na ARM procesoru na ploči ZedBoard. Ta aplikacija iščitava programski fajl, i na osnovu njega šalje podatke (piksele slike, podatke o rezoluciji, maske i instrukcije), u određenim vremenskim trenucima, ka jedinici za obradu piksela, putem standardnog AXI Full protokola, prilagođenog potrebama ovog dizajna.

Osnovna funkcionalnost ovako projektovanog sistema predstavlja primena programskog fajla sa proizvoljnim brojem asemblerskih instrukcija, pomoću kojih se određuje slika, koja se prikazuje na ekranu monitora, a iscrta se putem standardizovanog VGA interfejsa.

Ovako projektovan dizajn pixel shader GPU core-a bi mogao biti dobra osnova za projektovanje kompletnog GPU core. Tada bi se mogle dodati i komponente kao što su vertex shader i geometry shader, a zatim i prilagoditi ovom dizajnu. Takođe, neki drugi pravac razvoja bi mogao biti razvoj animacija, odnosno igara u 2D grafici. Za ovu svrhu bi, na primer, mogle da se povežu još neke periferije na ZedBoard ploču.

9. LITERATURA

- [1] Wikipedia - The Free Encyclopedia, https://en.wikipedia.org/wiki/Field-programmable_gate_array, (datum pristupa: 2016-9-12)
- [2] Wikipedia - The Free Encyclopedia, https://en.wikipedia.org/wiki/VGA_connector, (datum pristupa: 2016-9-12)
- [3] The Institute of Electrical and Electronics Engineers, Inc. (2000), IEEE Standard VHDL Language Reference Manual, ISBN 0-7381-1948-2 SH94817, ISBN 0-7381-1949-0 SS94817, 3 Park Avenue, New York, NY 10016-5997, USA
- [4] Wikipedia - The Free Encyclopedia, https://en.wikipedia.org/wiki/Register-transfer_level, (datum pristupa: 2016-9-12)
- [5] Xilinx (2015), Zynq-7000 All Programmable SoC Technical Reference Manual, UG585 (v1.10)
- [6] Xilinx (2014), ZedBoard_HW_UG (v2_2)
- [7] ARM (2003, 2004, 2010, 2011), AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™ ACE and ACE-Lite™, ARM IHI 0022D (ID102711)
- [8] Microsoft, [https://msdn.microsoft.com/en-us/library/windows/desktop/bb219679\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/bb219679(v=vs.85).aspx), (datum pristupa: 2016-9-12)
- [9] Xilinx (2015), Vivado Design Suite User Guide Creating and Packaging Custom IP, UG1118 (v2015.3)

Kratka biografija:



Vladimir Vrbaški rođen je u Zrenjaninu 1992. god. Bachelor rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Mikroprocesorski sistemi i algoritmi odbranio je 2015. godine.

**IZGRADNJA I ZAŠTITA DISTRIBUTIVNE 110/20 KV I 20/0,4 KV
TRANSFORMATORSKIH STANICA****CONSTRUCTION AND PROTECTION DISTRIBUTION 110/20 I 20/0,4 KV
TRANSFORMATOR STATIONS**

Aleksandar Tankosić, *Fakultet tehničkih nauka, Novi Sad*

Oblast - ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Rad ukratko opisuje uslove i izgradnju transformatorskih stanica 110/20 i 20/0,4 kV gde je opisana visokonaponska oprema, njene tehničke karakteristike i načine primene. Opisana je i zaštita elemenata koje čine sastavni deo električnih mreža kao i konkretni primeri kako je primenjena i izvedena zaštita i na transformatorskim stanicama, i način parametrisanja. Rad se bavi i načinima izgradnje tipskih SN/NN transformatorskih stanica na našem području uz konkretan primer.

Abstract - The paper describes briefly the conditions and construction of 110/20 and 20 / 0.4 kV substation where is described high-voltage equipment, its technical characteristics and methods of application. It is described and the protection of elements that form an integral part of the electrical networks as well as concrete examples of how it is applied and implemented to protect substations, and the way of parameterization. The paper deals with the ways of construction typical MV / LV transformer stations on our territory with a concrete example.

Cljučne reči: transformatorska stanica, zaštita, izgradnja, tipске transformatorske stanice,

1. UVOD

Elektroenergetski sistem sa svojim elementima čini jedan od najvažnijih i strateških delova današnjeg modernog sveta bez kojeg je nezamisliv razvoj društva i privrede u poslednjem veku, a pogotovo u sadašnjem. Elektroenergetski sistem postaje sve kompleksnijim do polovine ovoga veka predviđa se da potrošnja električne energije poraste tri puta u odnosu na današnju potrošnju. Zato se mora razviti sadašnja mreža i investirati u novu da bi se zadovoljili budući zahteve za energijom. Rad se bavi tipskim rešenjima transformatorskih stanica 110/20 kV i 20/0,4 kV [1,2].

U drugoj glavi objašnjeno je od čega se sastoji EES i u njemu načini izgradnje transformatorske stanice 110/20 kV. Načini kako se odabire lokacija, čega se treba pridržavati i kako treba da izgledaju rešenja koja su tipiska u sistemu. Opisana je najvažnija oprema na 110 kV i 20 kV strani i transformator.

Treća glava daje opis Simesove zaštite Siprotec, opšti zahtevi mikroprocesorske zaštite, struktura sistema i DIGSI 4 programski paket za parametrisanje zaštite.

NAPOMENA:

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

Četvrta glava posvećena je opisu rekonstrukcije zaštite u transformatorskoj stanici 110/20 kV na konkretnom primeru sa opisom postojećeg i budućeg stanja.

U petoj prikazana su tipiska rešenja (stubnih, montažno-betonskih i kompaktno-betonskih transformatorskih stanica) koja se koriste u našem elektroenergetskom sistemu.

Šesta glava je posvećena konkretnom primeru izgradnje tipске stubne transformatorske stanice.

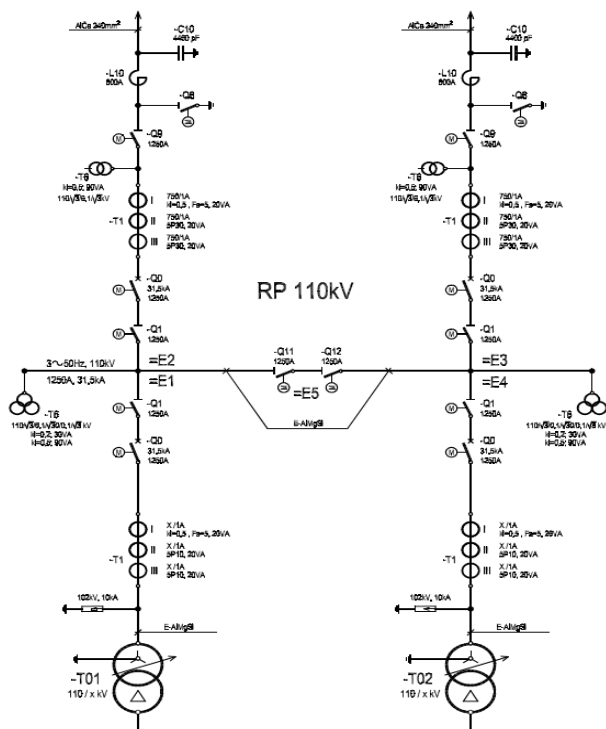
**2. DISTRIBUTIVNA TRANSFORMATORSKA
STANICA 110/20 kV**

Elektroenergetski sistem (EES) sastoji se od 4 osnovne celine:

1. Elektrane (izvori električne energije) različitih vrsta u kojima se proizvodi električna energija
2. Prenosna mreža kojom se električna energija prenosi od elektrana do distributivne mreže i velikih potrošača, te razmjenjuje snaga između povezanih elektroenergetskih sistema.
3. Distributivna mreža kojom se električna energija preuzeta iz prenosne mreže ili manjih elektrana priključenih na distributivnu mrežu distribuira do srednjih i malih potrošača priključenih na distributivnu mrežu.
4. Potrošači električne energije koji preuzimaju električnu energiju iz prenosne ili distributivne mreže i troše je za vlastite potrebe, tj. za napajanje različitih vrsta električnih potrošača.

Lokacija transformatorske stanice 110/20 kV se određuje prema urbanističko-tehničkim uslovima koje izdaje nadležni organ i oni utvrđuju tehničke uslove za izgradnju TS i priključenje na infrastrukturne instalacije [3-12].

U distributivnoj TS 110/20 kV na području Elektrovojvodine koristi se vazduhom izolovano postrojenje za otvorene prostore dok se metalom oklopljeno i gasom SF₆ izolovano postrojenje ne koristi, mada je ono predviđeno da se prvenstveno koristi u gradskim jezgrama sa velikom gustinom opterećenjem, ali planiranjem u sistemu EPS-a izbegava se gradnja transformatorskih stanica 110/20 kV u takvim uslovima i obično je lokacija transformatorske stanice, a sa njima i razvodnih postrojenja nalazi u prigradskom području gde je moguća sigurna izvedba razvodnog postrojenja vazduhom izolovanog.

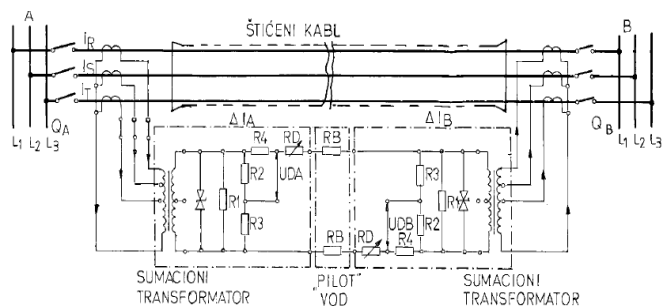


Slika 1. Jednopolna šema postrojenja 110 kV za spoljnu motažu – klasično postrojenje

Oprema na 110 kV strani:

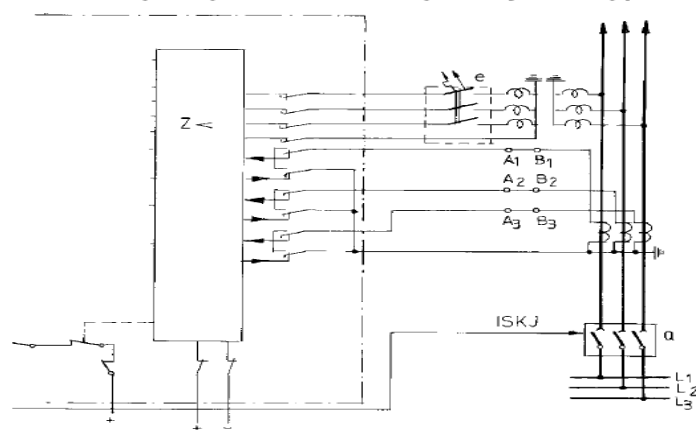
- **Odvodnici prenapona** su uređaji koje je napravljen za zaštitu električnih aparata od trenutnih prenapona i za ograničenje trajanja i češće za smanjenje amplitude prateće struje.
- **Naponski transformatori** su merni uređaji čiji je sekundarni napon u normalnom radu praktično proporcionalan sa primarnim naponom i sa njim je praktično u fazi. Nazivni primarni napon 110 kV Nazivni sekundarni napon 100 V ili 200 V (retko).
- **Strujni transformator** je merni transformator čija je sekundarna struja, u normalnom radu, približno proporcionalna primarnoj struji i približno sa njom u fazi.. Standardne vrednosti nazivnih sekundarnih struja su: 5A i 1A.
- **Prekidači** su mehanički rasklopni aparati koji mogu da se uključuju, provode i prekidaju nominalnu struju kola kao i da uključuju, provode u određenom vremenu i prekidaju struje u nenormalnim uslovima strujnog kola, kao što su one pri kratkom spoju.
- **Rastavljači** vidno i sigurno odvajaju deo postrojenja koje nije pod naponom od dela koji je pod naponom kako bi omogućili pristup pojedinim elementima, dok su drugi delovi postrojenja u pogonu.
- **Sabirnice** omogućavaju međusobno povezivanje različitih elemenata elektroenergetskog sistema (vodovi, transformatorske stanice, razvodna postrojenja) odnosno elemenata unutar elektroenergetskih objekata (transformatori, generator, motori itd).

U ovom poglavlju je naglašena pažnja na izbor uređaja za zaštitu 110 kV kablovskih i nadzemnih vodova koji se koriste u elektrodistributivnoj mreži Srbije.



Slika 2. Principijska šema veze podužne diferencijalne zaštite

Osnovni parametri podužne diferencijalne zaštite, podužni diferencijalni relej je naznačene struje 1A osetljiv na kvarove unutar štice deonice (zone) sa što kraćim sopstvenim vremenom ($t \leq 30\text{ms}$). Relej deluje na isključenje prekidača bez vremenske zadržke. Relej treba da bude neosetljiv na više harmonike i struje uključenja kabela.



Slika 3 - Principijska šema vezivanja distantne zaštite

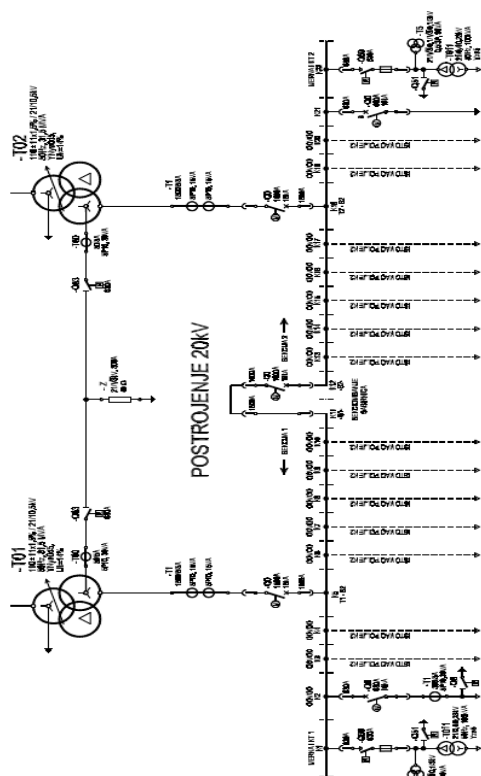
Kao osnovna zaštita 110 kV nadzemnih vodova koristi se selektivna distantna zaštita za petljaste višestrano napajanje mreže srednjeg i visokog napon, sledećih osnovnih zahteva:

- Pobuda je na prekostrujnom ili podimpedantnom principu.
- Zaštita se priključuje na tri strujna transformatora sekundarne struje 1 A i tri naponska transformatora sekundarnog napon 100/ $\sqrt{3}$ V.

Uz osnovnu zaštitu nadzemnog 110 kV voda primenjuje se i i uređaj za automatsko ponovno uključjenje (APU).

Zaštita sabirnica 110 kV ostvaruje se na jedan od sledećih načina:

- Okretanjem smeru slobodnih stepena distantne zaštite u oba dalekovodna polja, uz smanjenje odgovarajućih vremenskih zadržki na minimum.
- Korišćenjem II-og stepena distantne zaštite u susednom postrojenju 110 kV.



Slika 4 - Jednopolna šema postrojenja 20 kV u TS 110/20 kV instalisane snage 2x31,5 MVA

U distributivnoj TS 110/20 kV koristi se razvodno postrojenje sa vazduhom izolovanom, metalom zaštićenom rasklopnom aparaturom sa četiri odeljka po čeliji: sabirnički, prekidački sa vakuumskim prekidačima na izvlačenje, izlazni (kablovski), odeljak za niskonaponsku opremu.

Razvodno postrojenja 20 kV se montira u zasebnu prostoriju (zgradu).

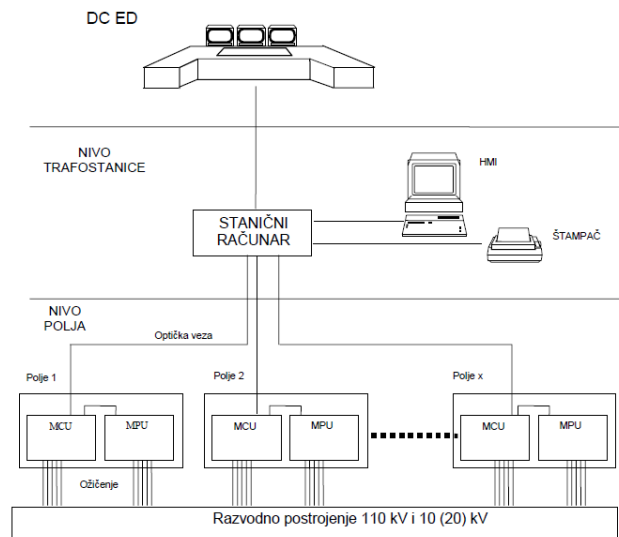
Energetski transformator je trofazna jedinica sa dva ili tri odvojena namotaja, namotaji 110 kV je primarni namotaj sa izvodima za regulaciju. Ovi transformatori se izvođe za spoljnu montažu bez obzira na njihovo mesto ugradnje.

3. OPIS MIKROPROCESORSKE ZAŠTITE SIPROTEC, PROGRAMSKI JEZIK DIGSI 4

Sistem mikroprocesorske zaštite i upravljanja vrši kompletan nadzor i upravljanje cele transformatorske stanice. Ovaj sistem mora da ispuni sve zahteve upravljanja kako iz transformatorske stanice tako i iz dispečerskog centra.

Upravljačka i zaštitna oprema za razvodno postrojenje 20 kV će biti smeštena na 20 kV čelijama (dok je za postrojenje 110 kV upravljačka i zaštitna oprema smeštena u posebne ormare u upravljačkoj prostoriji).

Struktura sistema treba da bude zasnovana u dva hijerarhijska decentralizovana nivoa. Transformatorska stanica će biti nadzirana i upravljana preko HMI interfejsa sa nivoa transformatorske stanice, dok će pojedinačna polja, čelije, sistemi sopstvene potrošnje biti nadzirani i upravljani preko opreme na nivou polja.



Slika 5 - Struktura sistema

Simensova mikroprocesorska zaštita SIPROTEC 7sj82 je deo nove SIPROTEC 5 serije modularnih uređaja.

Ovaj uređaj je specijalno dizajniran za ekonomično upravljanje sredjenaponskim mrežama. Na prednjoj strani releja nalazi se veliki LCD displej na kome se mogu očitavati svi podaci u releju kao i vrednosti odabranih mernih veličina (struje, naponi, uglovi...).



Slika 6 - SIPROTEC 7sj82 mikroprocesorski uređaj

Programski paket DIGSI je danas poznato ime na tržištu. Prvobitno lansiran kao MS-DOS softver, DIGSI, u svojoj trećoj verziji, je dovoljno razvijen da postane pogodan alat za podešavanje numeričkih zaštitnih releja.

Programski paket DIGSI 4, razumljiv i lak za upotrebu, predstavlja logičnu inovaciju i omogućava podešavanje, puštanje u rad i funkcionisanje svih SIPROTEC releja zaštite, bez obzira na njihovu verziju. Zahvaljujući svome modularnom dizajnu, DIGSI 4 se može ekonomično koristiti u zavisnosti od odgovarajućih zahteva. Osnovna verzija DIGSI 4 pokriva većinu standardnih zahteva i može se proširiti sa dodatnim komponentama na zahtev kupca.

4. TRANSFORMATORSKA STANICA 110/20 KV SUBOTICA 2, REKONSTRUKCIJA POSTOJEĆE ZAŠTITE

Postojeće stanje: Postrojenje 110 kV TS Subotica 2 je izvedeno kao spoljno postrojenje sa klasičnom opremom u nestandardnoj H šemi. Sastoji se od dva dalekovodno-transformatorska polja, kompletno opremljena i od jednog spojnog polja 110 kV sa ukupno tri prekidača 110 kV od kojih je jedan spojni, a dva su transformatorska.

Postrojenje 20 kV se sastoji od dve sekcije jednostrukih glavnih sabirnica i jedinstvenih pomoćnih sabirnica. Čelije su klasičnog tipa, vazduhom izolovane. Postrojenje se sastoji od 24 čelije od kojih su dve trafo, jedna spojka i dodatak spojke, dve merne, jedna čelija kućnog transformatora i 17 izvodnih čelija [2].

Buduće stanje: U postrojenju 110 kV, ovom adaptacijom je predviđena kompletna zamena zaštitne opreme 110 kV mikroprocesorskim zaštitno-upravljačkim uređajima opremljenim fiber optičkom komunikacijom.

U postrojenju 20 kV, ovom adaptacijom je predviđena kompletna zamena zaštitne opreme 20 kV novim mikroprocesorskim zaštitno-upravljačkim uređajima. Svi novi mikroprocesorski uređaji biće povezani sistemom lokalnog i daljinskog upravljanja fiber optičkom komunikacijom IEC 61850.

5. DISTRIBUTIVNA TRANSFORMATORSKA STANICA 20/0,4 KV

Opis izgradnja i montažna tipskih rešenja distributivnih transformatorskih stanica 20/0,4 kV:

- Stubna transformatorska stanica (EV-1, EV-2, EV-3, EV-4).
- Montažno-betonska transformatorska stanica (EV11A, EV21, EV41A).
- Kompaktno-betonska transformatorska stanica

6. PRAKTIČAN PRIMER RELIZACIJE STS EV-4 NA PRIMERU STS "PETROVOSELSKI PUT" U BEČEJU

Da bi se data transformatorska stanica izgradila prema situacionom rešenju mora da se radi prema uslovima koje izdaje opština postupa.

Celokupna elektroenergetska mreža i transformatorske stanice moraju se graditi u skladu sa važećim zakonskim propisima i tehničkim uslovima:

Novi priključni dalekovod 20 kV počinje od postojećeg ČRS DV-20 kV br.1, a završava se na novoj STS "Petrovoselski put" EV-4 10(20)/0,4 kV.

STS "Petrovoselski put" na pogodno mesto u blizini izgrađene ISTS "Seke Ištvana" izgradiće se nova stubna transformatorska stanica 10(20)/0,4 kV, 400 kVA tipa EV-4, uz ugradnju energetskog transformatora naznačene snage 160 kVA.

Od nove STS "Petrovoselski put" do novog ČRS NN, izgradiće se nova nadzemna mreža na novim BS NN stubovima sa provodnicima SKS 2x(X00/0A 2x70+50/8+2x16 mm²).

7. ZAKLJUČAK

U master radu opisane su distributivne transformatorske stanice 110/20 kV i 20/0,42 kV i oprema koja se koristi kao tipska rešenja u okviru EPS-a, konkretno u okviru Elektrovojvodine u svim njenim ograncima. Da bi se došlo do tipskih projekata koji su primenjivi u praksi trebalo je mnogo godina i iskustva sa različitim rešenjima.

Ulaganjem u relejnu zaštitu, automatiku i izgradnju distributivnih transformatorskih stanica moguće je postići značajno poboljšanje kvaliteta napajanja potrošača.

8. LITERATURA

- [1] Tehnička preporuka 12, osnovni tehnički zahtevi za izgradnju distributivnih transformatorskih stanica 110/10 kV, 110/20 kV i 110/35/10 kV, EPS
- [2] Elektroenergetika kroz standarde, zakone, pravilnike i tehničke preporuke; Gojko Dotlić
- [3] Tehnička preporuka br. 4a2, Zaštita elektrodistributivnih vodova 110 kV, EPS
- [4] Elektroenergetika kroz standarde, zakone, pravilnike i tehničke preporuke; Gojko Dotlić
- [5] Tehnička preporuka br. 11 Odabiranje parametara ET-a 110/x kV, EPS
- [6] Prilog tehničke preporuke 4, Specifikacija za sistem mikroprocesorske zaštite i upravljanja u trafostanicama I izdanje, EPS
- [7] file:///C:/Users/Korisnik/Downloads/DIGSI_STARTUP_A5_EN.pdf
- [8] Adaptacija objekta TS 110/20 kV „Subotica 2“ NDC d.o.o. Beograd, investitor Elektrovojvodina d.o.o. Novi Sad
- [9] Tehnička preporuka br. 1v Stubna transformatorska stanica 20/0,4 kV, EPS
- [10] <http://www.betonjerka-sombor.co.rs/info.php?id=2>
- [11] https://issuu.com/elnosbl/docs/biosco_brosura_sr_f4
- [12] Idejni projekat STS „Petrovoselski put“ Bečej, Projektni biro Elektrovojvodine d.o.o. Novi Sad

Kratka biografija:



Aleksandar Tankosić rođen je 1991.godine u Drvaru. Diplomski- rad odbranio je 2014. na Fakultetu tehničkih nauka. Od 2015. radi kao elektoinženjer u firmi GAT d.o.o. Novi Sad.

REALIZACIJA PROGRAMSKE PODRŠKE ZA KOMUNIKACIJU IZMEĐU DSP DEMO PLATFORME I PERSONALNOG RAČUNARA**DEVELOPMENT OF SOFTWARE FOR COMMUNICATION BETWEEN DSP DEMO PLATFORM AND PC**

Miloš Dobrosavljević, *Fakultet tehničkih nauka, Novi Sad*

Oblast - ELEKTROTEHNIKA I RAČUNARSTVO

Kratak zadržaj – U ovom radu su opisane osnove digitalne obrade signala kao i komunikacioni protokoli koji se koriste u audio sprežnim sistemima. Dat je opis aplikacije koja omogućava komunikaciju između računara i CS49834 DSP procesora. Aplikacija je razvijena u programskom jeziku C.

Abstract – This paper describe fundamentals of digital signal processing and communications protocols which is use in audio connected system. The paper also provides a description of application which is use to provide communication between CS49834 DSP processor and the PC. Application is developed in C programming language

1. UVOD

Proizvođači audio video potrošačke elektronike (AVR) suočavaju se nizom poteškoća prilikom razvoja novih tehnologija. Neke od ovih poteškoća ogledaju se u razvoju kompleksnog softvera koji se izvršava na digitalnom signal procesoru (DSP) i mikrokontroleru (MCU), licenciranju svake od primenjenih tehnologija, kao i sertifikaciji u skladu sa Dolby i DTS zahtevima. Osnovna uloga AVR-a jeste prijem audio i video signala sa raznih izvora a potom obrada tog signala, koju vrši DSP i na kraju prosleđivanje na izlaz kao što je zvučnik za audio signal ili ekran televizora za video signal. Pored mogućnosti obrade analognog i digitalnog signala, AVR uređaji poseduju i mogućnost dekodovanja komprimovanog signala ukoliko je to potrebno. Postoje više tipova audio formata koji se nalaze na tržištu, neki od njih su Dolby TrueHD, Dolby Digital, Dolby Digital Plus i drugi. U ovom radu bilo je potrebno realizovati aplikaciju za personalni računar (PC) u programskom jeziku C, koja preko univerzalne serijske komunikacije (USB) upravlja centralnom jedinicom za obradu audio podataka (DSP).

U cilju realizacije aplikacije se koristi CRD49834 demo platforma firme Cirrus Logic koja je referenti primer za dizajn AVR potrošačke elektronike. Ova platforma omogućava testiranje skoro svih tehnologija od Dolby (AC3, DD+, TrueHD, Atmos) i od DTS (Legacy, HD-MA, UHD) popularnih u AVR uređajima. Platforma podržava različite audio ulaze kao što su HDMI, SPDIF, DAI i analogni ulaz. CRD49834 poseduje trojezgarni DSP procesor CS49834 firme Cirrus Logic i mikrokontroler R5F100LL firme Renesas koji upravlja pomenutim DSP-om. Aplikacija koja

je realizovana u ovom radu ima zadatak da u potpunosti zameni mikrokontroler i upravlja digitalnim signal procesorom dok se uloga mikrokontrolera ogleda u prosleđivanju poruka od aplikacije ka DSP-u i obrnuto. U radu je opisana arhitektura demo platforme CRD 49834 kao i opis trojezgarnog DSP procesora CS49834, realizovana je aplikacija u programskom jeziku C i dati su algoritmi koji opisuju rad aplikacije.

2. DIGITALNA OBRADA SIGNALA

Sa razvojem i usavršavanjem digitalnih tehnologija, digitalni sistemi su postali veoma pogodni za digitalnu obradu signala u realnom vremenu i zamenili su analogne sisteme. Iz sve veće potrebe za digitalnom obradom signala, pojavljuju se mali mikroračunari, specijalno dizajnirani za obradu digitalnih signala koji su smešteni u jednom čipu. Oni se nazivaju Digitalni signal procesori ili DSP (Digital Signal Processors). Na slici 1 je prikazan lanac obrade signala u tri koraka.



Slika 1. Lanac obrade signala

Ulazni signal koji je analogni se pre obrade u DSP-u digitalizuje preko analogno digitalnog (A/D) konvertora i prosleđuje se na obradu. U fazi obrade se nad njim vrši matematičko manipulisanje koje se odigrava u DSP procesoru. Poslednja faza je prosleđivanje signala na digitalno analogni konvertor (D/A) koji signal pretvara ponovo u analogni domen i prosleđuje ga na izlaz. DSP je dizajniran da veoma brzo izvodi matematičke funkcije kao što su sabiranje, oduzimanje, množenje ili deljenje.

3. AUDIO SPREŽNI SISTEMI

U cilju pravilnog funkcionisanja lanca obrade audio podataka, neophodno je njihovo razumevanje kako bi se pravilno koristili. Najčešći tipovi audio sprega koji se koriste u audio sistemima mogu se svrstati u dve osnovne grupe: 1. Audio sprežni sistemi razvijeni za specifičnu namenu prenosa audio informacija 2. Audio sprežni sistemi opšte namene koji pored primene u oblastima obrade audio informacija mogu se sresti i u drugim sistemima opšte namene

3.1 S/PDIF sprega

Postoje različiti standardi za prenos audio signala. Jedan od najvažnijih i najrasprostranjeniji standard jeste SPDIF

NAPOMENA:

Ovaj rad poristekao je iz master rada čiji mentor je bio dr Ivan Mezei, docent.

(Sony/Philips Digital Interface) je zapravo standard koji je razvijen od strane SONY-a i PHILIPS-a 80-tih godina za potrebe povezivanja CD-a (compact disc) i DAT-a (Digital Audio Tape). Ovaj standard namenjen je za ostvarenje digitalne veze između potrošačkih audio uređaja na kratkim rastojanjima a osnovni cilj je izbegavanje višestrukih A/D i D/A konverzija u lancu obrade audio podataka. U početku je SPDIF standard bio namenjen samo za prenos signala koji nije komprimovan odnosno PCM (Pulse Code Modulation) podataka ali sa razvojem DVD-a (Digital Video Disc) i sve većom upotrebom audio kompresije, ovaj standard je nadograđen za prenos i komprimovanih audio sadržaja.

3.1.1 Prenos višekanalnih audio podataka preko S/PDIF-a i audio signala bez gubitaka

Sa pojavom DVD diskova, uređaji potrošačke elektronike počinju masovno da koriste višekanalne audio sadržaje. Za standardni format audio zapisa na DVD diskovima usvojen je Dolby Digital standard (sa bitskom brzinom od 384 ili 448 kbps), a nešto kasnije i DTS Digital Surround (sa bitskom brzinom od 1,5 Mbps). Postalo je jasno da postojeći SPDIF standard mora biti nadograđen što je i učinjeno usvajanjem IEC61937 standarda koji definiše prenos audio komprimovanog signala preko SPDIF spregu.

Pojavom novih generacija Blu-Ray diskova, SPDIF standard se sve manje koristi jer je zamenjen naprednim tipom digitalne spregu HDMI (High-Definition Multimedia Interface). Moderni uređaji potrošačke elektronike imaju mogućnost korišćenja audio sadržaja bez gubitaka (lossless). Zvučni zapis u novijoj generaciji diskova mogu da nose i do 8 kanala (7.1) sa učestanostima odabiranja 192kHz i nivoom kvantizacije 24bita po odabirku. Brzina audio sadržaja je ograničena na 24 Mbps. Za ove potrebe nije moguće koristiti SPDIF spregu, već je ona zamenjena modernijim i naprednijim spregama kao što je HDMI ili I2S.

3.2 HDMI sprega

HDMI (High-Definition Multimedia Interface) je medijum koji je namenjen za prenos nekomprimovanog video i komprimovanog ili nekomprimovanog audio sadržaja. Ova sprega podržava protok podataka brzine do 10.2 Gbps, što omogućava istovremeni prenos slike visoke rezolucije 1080p (u poslednjoj verziji 1.4 podržane su i ultraHD rezolucije kao što su 2K i 4K kao i 3D slika) i višekanalnog zvuka. Prenos slike i zvuka preko HDMI spregu se vrši u čisto digitalnom formatu i time je prenos značajno otporniji na smetnje. HDMI je šifrovan standard za prenos slike i zvuka i nepohodno je koristiti namenske HDMI prijemnike koji su sertifikovani od strane HDMI fondacije. Primer jednog takvog prijemnika koji se nalazi na demo platformi koja je korišćena u ovom radu jeste Sil9135 od proizvođača Silicon Image, jednog od osnivača HDMI fondacije.

3.3 I²C i SPI protokol

SPI (Serial Peripheral Interface) je serijska, komunikacija u oba smera koja se ostvaruje između jednog vodećeg (master) uređaja i jednog ili više pratećih (slave) uređaja. Ovaj protokol je razvijen od strane Motorole, ali iako se koristi u velikom broju familija integrisanih kola. Primena ovog protokola uključuje 4

linije za komunikaciju: SCLK linija za takt, MOSI (Master out Slave in) izlaz iz vodećeg uređaja i ulaz u prateći uređaj, MISO (Master in Slave out) izlaz iz pratećeg, ulaz u vodećeg uređaj CS (Chip select) ili SS (Slave select) Odabir pratećeg uređaja. Neke prednosti SPI komunikacije jeste prenos podataka u oba smera, veći protok od I2C komunikacije (I2C se koristi u sličnim aplikacijama kao i SPI, ali ređe), nema metapodataka (ceo propusni opseg je za podatke), nije ograničena na širinu od 8 bita i mala potrošnja.

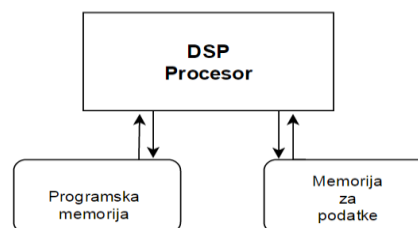
Još jedan vid komunikacije koji se koristi za serijsku spregu na malim rastojanjima a najčešće za povezivanje integrisanih kola na štampanoj ploči, jeste I2C (Inter-Integrated Circuit) česta oznaka je i I²C. Ovaj protokol je razvio Philips za potrebe komunikacije perifernih uređaja sa centralnim procesorom. Za razliku od SPI komunikacije, ovaj protokol podržava više vodećih (Master) uređaja. Ovaj protokol koristi dve fizičke linije SCL (Serial Clock) i SDA (Serial Data). Preko linije SCL prenosi se takt signal dok SDA služi za prenos podataka.

4. ARHITEKTURA CRD 49834

Referenti primer za dizajn AVR potrošačke elektronike, u ovom radu je korišćena CRD49834 demo platforma firme Cirrus Logic koja omogućava testiranje skoro svih modernijih tehnologija od Dolby i DTS. Demo platforma podržava razne audio ulaze, analogni audio ulaz preko RCA konektora, digitalni audio ulaz preko HDMI-a (konektor A tipa), optički digitalni ulaz kao i digitalni ulaz preko RCA konektora, postoji i 20 pinski digitalni DSP audio ulaz (DAI). Podržani audio izlazi na ovoj platformi su optički digitalni izlaz, DSP digitalni audio izlaz (DAO), 8x3.5 mm analogni izlaz i jedan HDMI izlaz (konektor A tipa, 1.4 standard). Ovaj sistem podržava obradu zvuka do 8 kanala. Centralna jedinica za obradu podataka jeste DSP procesor CS49834 firme Cirrus Logic koji može da obrađuje tipove zvučnih formata kao što su Dolby Digital, Dolby Digital Plus, Dolby TrueHD, DTS i DTS HD. Ovi formati se koriste za postizanje specijalnih zvučnih efekata koji se koriste u npr. u filmskoj industriji. Centralna upravljačka jedinica na demo platformi je mikrokontroler (MCU) R5F100LL firme Renesas koji preko SPI ili I2C komunikacije upravlja pomenutim DSP procesorom kao i ostalim periferijama na ploči.

4.1 DSP CS49834

DSP CS49834 je trojezgarni (jezgro A, B i C) digitalni signal procesor Crystal familije procesora, firme Cirrus Logic. Svako jezgro datog procesora radi na učestanosti od 300MHz. Procesor je realizovan u Harvard arhitekturi (Slika 2), što znači da postoji dva tipa memorije, odvojena memorija za podatke i posebna memorija za smeštanje programskog koda.

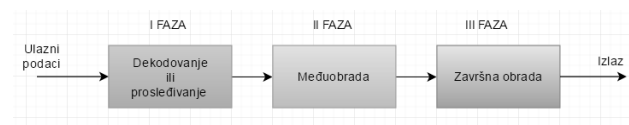


Slika 2. DSP procesor Harvard arhitektura

Memorija za podatke je podeljena u dva segmenta (X i Y segment). Količina raspoložive memorije za podatke (X i Y data) i za programski kod (P data) je data u tabeli 1. Na taj način DSP procesor ima mogućnost čitanja instrukcija i podataka iz memorije istovremeno što ubrzava rad celog procesora. Ovakva organizacija omogućava izvršavanje do 6 instrukcija u jednom taktu. Tako u jednom taktu moguće je izvršiti dva upisa ili čitanja u memoriju, dva uvećavanja ili umanjivanja adresnih registara i dve MAC instrukcije (Multiply and accumulate). Svako od tri jezgara sadrži osam 32-bitnih registara opšte namene koji su smešteni u memoriji i osam 72-bitnih registara namenjenih za operacije sa dvostrukom preciznošću ili smeštanje međurezultata koje još nazivamo i akumulatorima. Svako jezgro sadrži i dvanaest 16bitnih indeks registara koji su namenjeni za adresiranje prilikom upisa ili čitanja u memoriju. Svakim od indeks registara pripada i po jedan moduo registar koji omogućava da indeks registar može da radi u moduo režimu ili u bit-rezervisanom režimu.

4.2 Lanac obrade podataka

Lanac obrade audio podataka u DSP čipu se može posmatrati u tri faze (slika 3): Prva faza je da se podaci prvo dekoduju ukoliko je u pitanju komprimovani signal, ukoliko je u pitanju PCM samo se prosleđuju dalje na međuoobradu. U drugoj fazi se vrši međuoobrada podataka kao što je promena broja izlaznih kanala, smanjenje broja izlaznih kanala, povećanje izlaznih kanala. U završnoj trećoj fazi vrši se završna obrada nad podacima koji su rezultat nižih nivoa obrade.



Slika 3. Faze obrade podataka

Sistemska podrška CS49834 procesora u odnosu na vrstu aplikacije se može posmatrati u okviru 4 nivoa (Slot-a) koja su implementirana u svakom od postojećih tri jezgra: 1) Operativni sistem, programska podrška niskog nivoa 2) Dekoderski nivo 3) MPM (Mid-Processor Module) nivo međuoobrade 4) PPM (Post-Processor Module) nivo završne obrade.

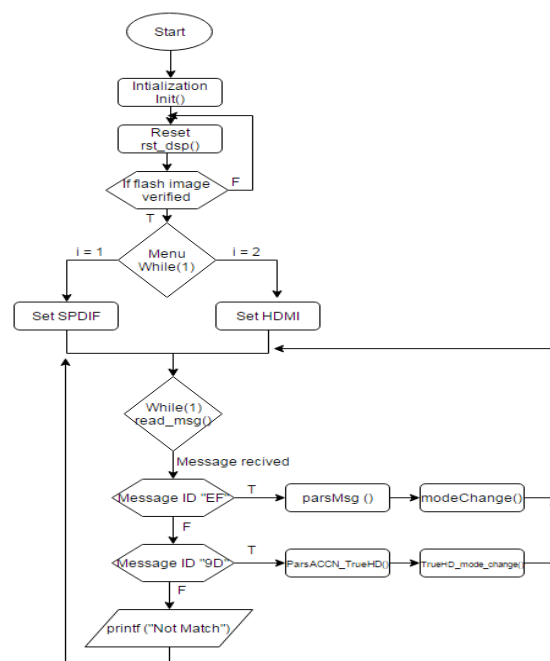
5. REALIZACIJA APLIKACIJE ZA PERSONALNI RAČUNAR

U ovom radu bilo je potrebno realizovati aplikaciju koja omogućava univerzalnu serijsku komunikaciju (USB) između računara i CRD49834 demo platforme. Aplikacija na strani računara preuzima ulogu mikrokontrolera dok uloga mikrokontrolera više nije kontrola DSP-a već samo prosleđivanje poruka ka DSP-u od strane aplikacije i obrnuto. Aplikacija prima poruke od DSP procesora i šalje individualne komande za promenu režima rada.



Slika 4. Zadatak rada

PC aplikacija treba da omogući komunikaciju preko USB-a sa DEBUG hardverskom ključ zaštitom što je prikazano na slici 4. DSP procesor i DEBUG hardverski ključ su povezani preko SPI protokola. Preko urađene aplikacije moguće je slati željena podešavanja za DSP i pratiti njegovo ponašanje. Za hardverski ključ se koristi se CDB-MCU-DEBUG firme Cirrus Logic. Za ostvarenje USB komunikacije koristi se biblioteka di_comm_wrapper.h koju je potrebno uključiti u programu. U ovoj biblioteci se nalaze ugrađene funkcije za ostvarenje USB komunikacije kako štp su diw_open funkcija koja otvara USB komunikaciju, diw_read_ctl_msg za čitanje poruka od DSP-a, diw_read_ctl_msg, read_board_cfg funkcija za čitanje vrednost registra, write_board_cfg funkcija za upis u registar itd. Algoritam rada aplikacije je dat na slici 5. Aplikacija prilikom pokretanja vrši inicijalizaciju preko funkcije init() koja inicijalizuje komponente na ploči kao što je definisanje protokola za komunikaciju i podešavanje A/D i D/A konvertora.



Slika 5. Algoritam aplikacije

Nakon inicijalizacije, DSP procesor se ponovo pokreće u Master režimu rada preko funkcije rst_dsp() i ukoliko je ponovno pokretanje uspešno, fleš je uspešno verifikovan. Program potom čeka na odabir izvora audio signala koji može da bude SPDIF(1) ili HDMI(2). Kada se odabere izvor audio signala, podešavanje odgovarajućeg izvora vrši se preko funkcije za slanje poruka DSP procesoru, write_message_to_dsp() gde se upisuje heksadecimalna vrednost 0xEF000006 00000001 ukoliko je u pitanju SPDIF ili 0xEF000006 00000001 ukoliko je u pitanju HDMI. Centralna funkcija aplikacije jeste čitanje poruka koje dolaze od strane digitalnog signal procesora. Pored čitanja poruka funkcija i poziva odgovarajući parser. Na taj način se posmatra njegovo ponašanje i upravlja njegovim radom. Prilikom pokretanja potrebno je da se funkcija neprestalno izvršava u petlji. Ovo je neophodno jer se poruke koje DSP šalje aplikaciji prosleđuju preko signala za prekid IRQ koji je aktivan na logičkoj jedinici. Ukoliko bi se samo jednom izvršila, funkcija bi se izvršila pre nego što se pojavi IRQ signal što znači da ne bi došlo do čitnaja poruka.

5.1 Raščlanjavanje poruka

Nakon čitanja poruka i detektovanja koja vrsta poruke je u pitanju poziva se odgovarajući parser. Dekodovanje poruka se zasniva na pretraživanju tabela u kojima se nalazi informacija koja se isčitava na osnovu vrednosti poruke. Parser uvek prima dve poruke kao jednu informaciju, prvi deo poruke predstavlja indeks poruke dok drugi deo predstavlja vrednost. Cela poruka koju obrađuje parser je dužine 4 bajta. U ovom radu je obrađen i slučaj dovođenja komprimovanog signala na ulazu. Signali komprimovanog audio signala se dovode preko DAI ulaza a za njihovo prosleđivanje ka DSP procesoru je neophodno podesiti HDMI primopredajnik kao izvor. U ovom radu je korišćen TrueHD tip audio formata. Kada se isčita poruka, izvrši se provera prve reči u poruci i ukoliko je ta vrednost 0x9D, to je signal da je u pitanju ACCN poruka koja je vezana za TrueHD audio format. U tom slučaju se poziva funkcija ParsACCN_TrueHD. ACCN poruka TrueHD audio formata sadrži osam reči u heksadecimalnom obliku koje nose informacije kao što su: učestanost o bitskog toka, informacije o formtu, informacije o tipu bitskog toka, potreba za pojačavanjem signala na izlazu, o dodeli kanala.

5.2 Promena režima rada DSP procesora

Poslednji zadatak aplikacije je promena režima rada DSP procesora. Funkcija nakon čitanja i raščlanjivanja Autodetect i ACCN poruka učitava odgovarajuću konkuranciju u DSP procesor. U aplikaciji su realizovane dve posebne funkcije za promenu režima rada: jedna kada je na ulazu PCM signal i posebna funkcija kada je na ulazu signal tipa True HD. Prilikom pozivanja funkcija, inicijalizuju se konstante koje učitavaju neke vrednosti iz heder fajla flash.h. Heder fajl flash.h je fajl koji se kreira prilikom programiranja projekta na fleš memoriju na ploči preko softverskog alata DSP Condenser firme Cirrus Logic. Ovaj heder fajl sadrži informacije o mogućim režimima rada DSP procesora, o režimu rada dekodera u DSP procesoru za komprimovan audio signal, kao i informacije u kom slotu jezgra se DSP procesora se učitavaju određene komponente lanca obrade audio signala. Funkcije ModeChange() i TrueHD_mode_change() sadrže definisane režime rada DSP procesora koje uključuju u zavisnosti od primljene poruke koju prosleđuje parser. Prilikom izvršavanja promene režima, funkcija šalje poruku DSP procesoru preko funkcije write_message_to_dsp koja ima heksadecimalnu vrednost 0xEF00000000000100 koja označava START. Na osnovu ove poruke, DSP procesor očekuje da će doći do jedne ili više promena modula u lancu obrade. Nakon poruke koja označava start, funkcije ModeChange() i TrueHD_mode_change() šalju vrednosti na osnovu kojih DSP procesor učitava odgovarajući modul u za to definisan slot. Na kraju se pošalje poslednja poruka koja ima heksadecimalnu vrednost 0xEF00000000000001 koja označava STOP. Nakon ove poruke DSP procesor zna da je funkcija završila sa promenom režima rada i procesor nastavlja normalan rad sa predhodno učitanim konkurencijom.

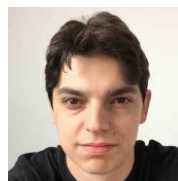
6. ZAKLJUČAK

U ovom radu realizovana je aplikacija koja simulira Host na računaru i daje softversko rešenje za komunikaciju i upravljanje DSP procesorom na CRD49834 demo platformi firme Cirrus Logic. Izvorni kod je napisan u programskom jeziku C, dat je detaljan opis funkcija kao i algoritama koji prate izvršavanje programa. Predstavljene su osnove o digitalnoj obradi signala kao i arhitektura trojezgarnog DSP procesora CS49834 i demo platforme koja je korišćena kao referentni primer audio i video prijemnika potrošačke elektronike. Izložen je detaljan opis audio sprežnih sistema i protokola za komunikaciju koji su korišćeni u ovom radu.

7. LITERATURA

- [1] Veljko Papić, *Digitalna obrada signala- predavanja*, Beograd, ETF, 2013.
- [2] *Audio sprežni podsistemi*, materijal sa predmeta Arhitekture i algoritmi DSP-a II, Novi Sad, FTN, 2012
- [3] Predrag Teodorović, Veljko Malbaša, *Mikroračunarski sistemi za rad u realnom vremenu*, Novi Sad, FTN, 2010
- [4] Ivan Mezei, Veljko Malbaša, *Mikroprocesorska elektronika-predavanja*, Novi Sad, FTN, 2015.
- [5] Sinhrona serijska komunikacija, materijal sa predmeta Računarska elektronika, Novi Sad, FTN, 2016
- [6] *A Beginner's Guide to Digital Signal Processing (DSP)*, <http://www.analog.com/en/design-center/landing-pages/001/beginners-guide-to-dsp.html> (datum pristupa: 01.08.2016)
- [7] *The SPI bus*, http://corelis.com/blog/what-is-an-spi-bus-an-introduction/#.V7DQf_mLTiU, (datum pristupa: 04.08.2016)
- [8] *CRD49834 Board User's Manual*, Cirrus Logic, 2014.
- [9] *CS498xx System Designers Guide*, Cirrus Logic, 2013.
- [10] *I2C Bus*, <http://www.i2c-bus.org/>, (datum pristupa: 04.08.2016)
- [11] *Dolby True HD*, Cirrus Logic, 2014

Kratka biografija:



Miloš Dobrosavljević rođen je u Petrovcu na Mlavi 1989. god. Visoku školu elektrotehnike i računarstva u Beogradu završio je 2011. god. Diplomirao na Fakultetu tehničkih nauka 2015 god.

ANALIZA ISPITNE STANICE ZA TESTIRANJE CEVI PRITISKOM**ANALYSIS OF PRESSURE LEAK TEST MACHINE**Viktor Udovički, Dragan Milićević *Fakultet tehničkih nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj – U radu je opisana kompletna analiza ispitne stanice za ispitivanje cevi pod pritiskom sabijenog vazduha, kao i njeno upravljanje. Dati su i predlozi unapređenja ispitne stanice.

Abstract – This paper describes the complete analysis of test station for testing the pipes with compressed air pressure and its controlling. Proposal of improvements are also given.

Cljučne reči: Elektrotehnika, automatika, upravljanje, pneumatika, programiranje.

1. UVOD

Ujedinjavanjem dva sistema, električne energije i pneumatike, dolazi se do nove tehničke celine, elektropneumatike, koja se bavi električnim upravljanjem pneumatskih elemenata (cilindara, elektromagnetnih razvodnika, ventila itd.). Elektropneumatika je prisutna u svim proizvodnim industrijama, pre svega u auto industriji, gde se za kontrolu proizvoda kao i sam proces proizvodnje koristi sabijeni vazduh upravljan nekom upravljačkom jedinicom. Cilj ovog rada je analiza principa rada ispitne stanice koja ispituje gotov proizvod –cevi, sabijenim vazduhom merenjem njenih parametara pada pritiska (curenje-loša zaptivenost cevi) i protoka vazduha kroz cev. Dobijeni rezultati moraju zadovoljiti sve granice propisane od strane proizvođača automobila gde će se ispitana cev ugraditi.

2. ANALIZA MEHANIČKOG SKLOPA ISPITNE STANICE**2.1. Karakteristike korišćenih aluminijumskih profila za konstrukciju ispitne stanice**

Manja specifična masa, izostanak pojave varničenja usled obrade alatima i veća otpornost na ogrebotine i koroziju su samo neke od prednosti aluminijumskih konstrukcija u odnosu na čelične konstrukcije.

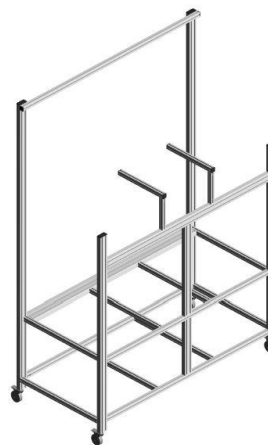
Aluminijumski profili su unapred formirani (ekstrudirani) profili od aluminijuma, unapred osmišljeni za lako sastavljanje i rastavljanje ispitnih-radnih stolova. Korišćeni aluminijumski tip profila je legure Aluminijum-Magnezijum-Silicijum (AlMgSi), sledećih karakteristika: zatezne čvrstoće min. 245 N/mm², specifične gustine 2,7 kg/dm³, tvrdoće 75 HB (tvrdoće po Brinell-u), eloksirani su (stalna otpornost na koroziju i ogrebotine).

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Milićević, docent.

2.3. Programski paket May-CAD

Za projektovanje aluminijumske konstrukcije ispitne stanice korišćen je programski paket May-CAD. Program je besplatan, jednostavan za korišćenje, poseduje sve neophodne elemente za povezivanje profila, kao i analizu finansijskog troška neophodnog za realizaciju nacrtane-željene konstrukcije. Program takođe pravi spisak potrebnih profila i spojnih elemenata. Na slici 1. prikazana je formirana mehanička konstrukcija ispitne stanice.



Slika 1. Mehanička konstrukcija ispitne stanice

2.4. Održavanje mehaničkog sklopa ispitne stanice

Pravilno i redovno održavanje ispitne stanice ima za cilj povećanje njenog veka trajanja. Obuhvata redovno podmazivanje ležaja i vodica pokretnih delova kao i redovnu kontrolu radnih i prihvatnih alata. Maziva korišćena za podmazivanje treba da imaju sledeće karakteristike: nivo opterećenja max. 15% od nominalnog dinamičkog opterećenja, opseg temperature između -10°C i 80 °C. Neka od preporučena maziva su: Klüberlub GL-261, Mobilux EP1 itd.

3. ANALIZA PNEUMATSKOG SKLOPA ISPITNE STANICE

Pneumatski sistem, koji je u ovom slučaju kontrolisan PLC-om, ima ulogu da visinu pritiska definisanu programom pripremi, napuni cevi sa vazduhom, zarobi vazduh unutar cevi (merenje pada pritiska) ili propusti vazduh kroz cev (merenje protoka) i obavesti PLC o statusima mernih elemenata (transmiteri pritiska i protokomer, respektivno).

3.1. Elementi pneumatskih sistema i pneumatska šema ispitne stanice

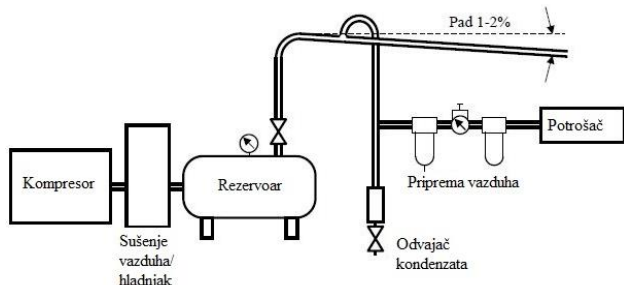
Osnovu pneumatskog sistema čini nekoliko elemenata. Kao osnovni element u sistemu se prepoznaje kompresor koji ima ulogu da mehaničku energiju pretvori u energiju

sabijenog vazduha. Jednu od vitalnih uloga u sistemu ima uređaj za sušenje i hlađenje vazduha čija je uloga da smanji vlagu u vazduhu i hladi zagrejani vazduh iz kompresora te na ovaj način pruža optimalne uslove radnom fluidu u sistemu pozitivno utičući na životni vek. Pored njega, sličnu ulogu u sistemu ima i odvajач kondenzata koji odvaja kondenzat iz vazduha kojim se odstranjuju problemi korizije i zamrzavanja kondenzata u zimskom periodu.

Kako bi izbegli varijacije u pritisku vazduha u sistemu, svaki ozbiljniji sistem poseduje rezervoar koji služi za skladištenje sabijenog vazduha radi kvalitetnijeg usklađivanja rada kompresor-potrošač.

Pripremna grupa sa manometrom ima ulogu da vrši kontinualno pripremanje vazduha potrošaču filtracijom stranih tela u vazduhu, regulacijom pritiska radi konstantnog nivoa pritiska i da vrši zauljivanje vazduha.

Slika 2. prikazuje izgled pneumatskog sistema.



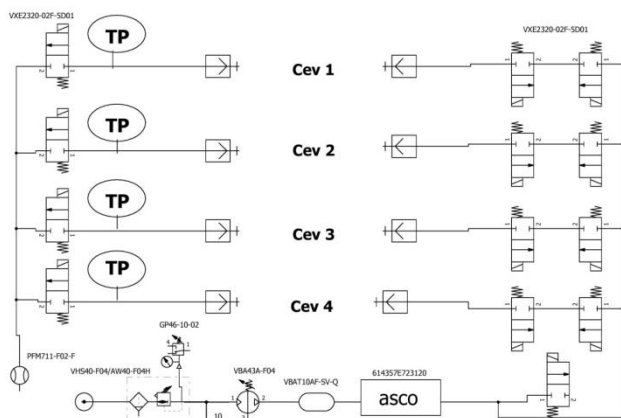
Slika 2. Osnovni elementi pneumatskog sistema

Ispitna stanica zahteva sabijeni vazduh višeg pritiska nego što kompresor obezbeđuje, i zato se koristi i multiplikator pritiska-pojačavač koji pojačava pritisak i na svom izlazu daje pritisak dva puta veći od ulaznog.

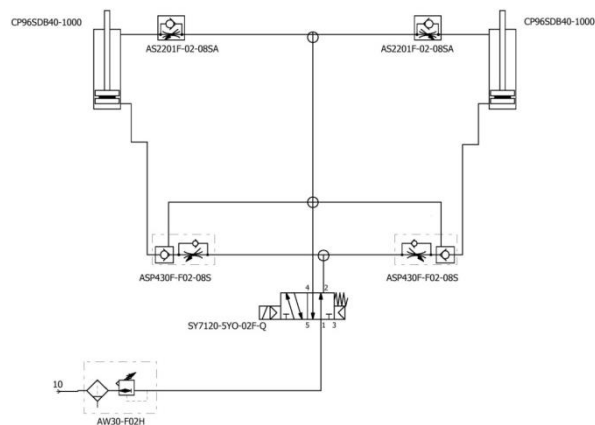
Pneumatski razvodnik u sistemu ima ulogu prekidača u pneumatskom sistemu. Bitne elemente u sistemu čine i transponderi.

U sistemu se koriste transponderi pritiska koji daju informaciju o visini pritiska u sistemu, te merič protoka vazduha-protokomer koji meri protok u ml/min. Kao mehanički izvršni elementi u sistemu se koriste pneumatski cilindri.

Slika 3. i slika 4. prikazuju pneumatsku šemu upravljačkog dela stanice i pneumatsku šemu vrata ispitne stanice, respektivno.



Slika 3. Pneumatska šema upravljačkog dela stanice



Slika 4. Pneumatska šema vrata ispitne stanice

Na slici 5. prikazani su elementi pneumatskog kola postavljeni na ispitnu stanicu.



Slika 5. Elementi pneumatskog kola postavljeni na ispitnu stanicu

3.1. Održavanje pneumatskog sistema

Radi pravilnog funkcionisanja pneumatskog sistema i produživanja radnog veka, jednom godišnje je potrebno ispustiti kondenzat, izvršiti vizuelnu proveru instalacije (proveru pneumatskih spojeva) i zamenu filtera vazduha u pripremnoj grupi.

4. ANALIZA ELEKTRIČNOG SKLOPA ISPITNE STANICE

4.1. Analiza električne šeme elektroormanu ispitne stanice

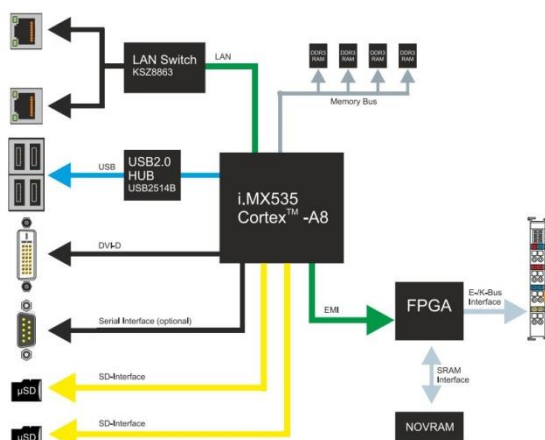
Električna šema ispitne stanice izrađena je u programskom paketu SEE Electrical CADdy++. Sistem upravljanja radi sa naponskom nivou 24 VDC. Srce sistema upravljanja predstavlja kombinacija PLC uređaja Beckhof CX9020 i panela osetljivog na dodir u boji Weintek MT8150iH. Komunikacija sa spoljašnjim svetom se ostvaruje korišćenjem ulaznih i izlaznih analognih i digitalnih modula, gde su izlazni moduli razdvojeni sa bezbednosnim releom da prilikom nepravilnog rada svi izlazni signali budu isključeni. Na bezbednosni rele su priključena tri pečurkasta tastera za nužni stop i mehanička bezbednosna ivica postavljena na dno vrata ispitne stanice. Semafor za vizuelnu indikaciju je trobojni, sa žutom, zelenom i crvenom sijalicom. Start i stop mašine je realizovan na dva mesta, na samoj ispitnoj stanici i u elektroormanu HMI panela.

Pored svake ispitne cevi se nalazi po jedna sijalica, trobojna (crvena boja-loš komad, zelena boja-dobar komad i narandžasta boja-ispitivanje u toku). Cilindri za podizanje i spuštanje vrata imaju reed rele (senzore pozicije) koji služe za informaciju u kojoj poziciji se vrata ispitne stanice nalaze.

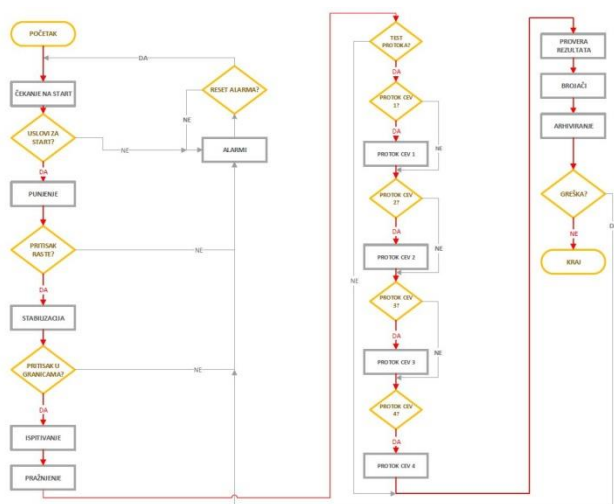
4.2. Analiza programa PLC-a ispitne stanice

Na slici 5. prikazana je blok šema PLC-a Beckhoff serije CX9020. PLC je embedded PC sa svojom radnom i skladišnom memorijom, procesorom i operativnim sistemom. Postavlja se na DIN šinu. Ima ulazno/izlazne portove: LAN port (ostvorena ethernet komunikacija sa HMI panelom), USB 2.0 port (skladištenje arhive) i DVI port (opcija priključenja na računarski monitor), Program je skladišten na micro SD memorijskoj kartici, a pisan je u programskom paketu TwinCat 2.11.

Slika 6. prikazuje algoritam programa PLC-a.



Slika 5. Blok šema PLC-a Beckhoff serije CX9020



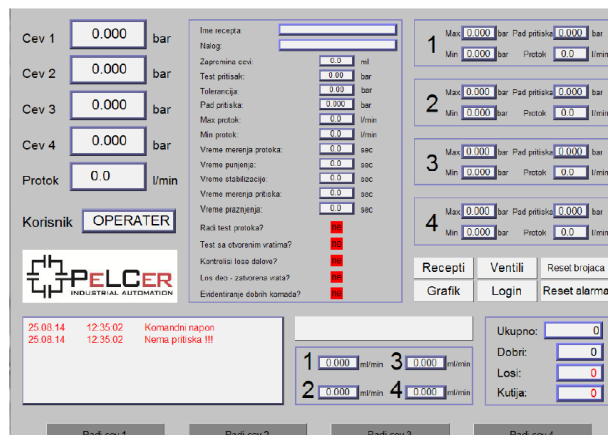
Slika 6. Algoritam programa PLC-a ispitne stanice

Crvenom linijom označen je glavni tok programa. Blok »POČETAK« označava početak programa PLC-a. Blok »ČEKANJE NA START« predstavlja tok programa gde je program pravilno učitao i spreman na rad- čeka se pokretanje procesa ispitivanja. Nakon pokretanja procesa ispitivanja, program proverava uslove zadate u samom PLC programu (neki od uslova su: da li su vrata zatvorena, da li ima komandnog napajanja, da li je pritisak prisutan, da li je odabran recept itd.). Ovo se obavlja u uslovnom bloku »USLOVI ZA START?« Ukoliko barem jedan od uslova

nije ispunjen, alarm postaje aktivan (blok »ALARMI«), i ne može se pokrenuti proces ispitivanja dok se ne otkloni problem opisan alarmom. Kada su svi uslovi za start ispunjeni, sledi proces punjenja cevi sabijenim vazduhom (blok »PUNJENJE«). Uslov »PRITISAK RASTE?« ispituje da li raste pritisak u ispitnoj cevi/cevima (kontrolise ih transmitter pritiska). Neki od razloga da pritisak ne raste mogu biti loše mehanički prihvaćena cev, loš spoj pneumatskih komponenti, loš spoj transmeta pritiska itd. Ukoliko barem jedan od uslova dat PLC programom nije ispunjen, alarm je aktivan. Ukoliko su svi uslovi ispunjeni, vrši se stabilizacija pritiska (blok »STABILIZACIJA«). Uslov »PRITISAK U GRANICAMA?« proverava da li se pritisak stabilizovao i da li je u granicama zadatim PLC-om. Ukoliko barem jedan od uslova nije ispunjen, alarm je aktivan. Ukoliko su svi uslovi ispunjeni, sledi proces ispitivanja (blok »ISPITIVANJE«). Vrš se ispitivanje tj. merenje pritiska na početku ispitivanja, i sukcesivnim merenjima pritiska nakon određenog vremena zadatog receptom. Uzima se najniži pritisak, i razlika početnog i najnižeg predstavlja pad pritiska. Sledeći korak je pražnjenje ispitnih cevi-rasterećenje sistema (blok »PRAŽNJENJE«). Sledeći uslov je pitanje da li se radi test merenja protoka vazduha kroz cev (blok »TEST PROTOKA?«). Nakon toga program sukcesivno proverava da li se meri protok na svim cevima ili samo na ponekim (operator odabira na početku programa koje lokacije za cevi ispituje, od 1 do 4). Ovi uslovi su »PROTOK CEVI 1« pa svaka sledeća cev do »ROTOK CEVI 4«. Sledi »PROVERA REZULTATA«. »BROJAČ« broji koliko je ukupno ispitanih cevi, dobrih i loših. Sledi »ARHIVIRANJE«. Na osnovu dobijenih rezultata proverava se da li je bilo neke greške (uslov »GREŠKA?«). Ukoliko nije bilo grešaka, time je proces ispitivanja cevi završen. Svaki alarm operator mora otkloniti. Program proverava da li je alarm otklonjen (uslov »RESET ALARMA?«). Ukoliko nije, program čeka na operatera i na otklanjanje alarma, ukoliko je alarm otklonjen, program može biti pokrenut iz početka.

4.2. Analiza programa touch panela (HMI-a) ispitne stanice

Program panela je pisan u programskom paketu Weintek EasyBuilder PRO 4.00.1. Osnova korisničkog interfejsa prikazana je na slici 7.



Slika 7. Početni ekran korisničkog interfejsa

Nakon što se opareter ulogovao, pritiskom na taster recepti bira jedan od prethodno unešenih receptata od strane višeg

korisničkog nivoa. Receptom su definisani parametri ispitnog pritiska, zapremine cevi, vremena merenja, da li se radi ispitivanje protoka itd. Pritiskom na taster »RADI« cev 1 do 4 odabira koju cev želi ispitivati. Ukoliko alarm nije aktivan, pokreće se proces ispitivanja pritiskom na jedan od dva »START« tastera. U svakom trenutku ispitivanja operater vidi trenutne vrednosti, kao i zadate vrednosti, stanje brojača dobrih, loših i kutija u cevi (loših cevi ubačenih u kutiju sa lošim cevima). Postoje sledeći ekrani: početni-osnovni ekran (prijavljivanje korisnika (korisnički nivoi sa dozvoljenim pristupima i izmenama), alarmi i obaveštenja prilikom rada, brojači za dobre, loše i brojač loših komada stavljenih u NOK kutiju (kutija za loše komade), trenutne vrednosti u toku ispitivanja, izmerene vrednosti nakon završenog ispitivanja, zadate vrednosti u receptu), recepti (kreiranje novih i odabir postojećih recepta), ventili (prikaz on/off stanja ventila), grafik toka ispitivanja i arhiviranje (fajl sa rezultatima koji se može obrađivati u Excel-u).

6. GRAFIK TOKA ISPITIVANJA CEVI

Proces ispitivanja cevi se sastoji iz pet koraka:

1. provera uslova u kome PLC proverava uslove koji trebaju biti ispunjeni da bi se počelo ispitivanje (opisano u poglavlju 4.2),
2. punjenje cevi sabijenim vazduhom u kome se vrši otvaranje ulaznih ventila, zatvaranje izlaznog ventila cevi, punjenje cevi vazduhom i zatvaranje ulaznog ventila nakon što je cev napunjena vazduhom,
3. stabilizacija pritiska i merenje pritiska koja ima ulogu da izvrši stabilizaciju zarobljenog vazduha u cevi određena receptom i merenje pada pritiska tj. razlike početnog pritiska nakon stabilizacije i najnižeg pritiska merenog nakon vremena definisanog receptom, i
4. merenje protoka vazduha kroz proceduru otvaranja ulaznih i izlaznog ventila i merenje protoka vazduha kroz ispitnu cev.

Koraci su ilustrovani na grafiku 8. Crvena kriva predstavlja tok ponašanja pritiska vazduha cevi 1 u toku ispitivanja pada pritiska, dok zelena kriva predstavlja tok merenja protoka vazduha cevi 1.



Slika 8. Grafik toka ispitivanja cevi

5. PREDLOZI UNAPREĐENJA ISPITNE STANICE

Kao i kod svih mašina, unapređenja su uvek moguća i neophodna. Unapređenje održavanja bezbednosti mehanike se odnose na dodavanje odeljka održavanja u korisnički interfejs programa HMI-a za vreme kada je potrebno izvršiti etaloniranje transmitera pritiska, podmazivanje klizača, zamenu filtera, kontrolu bez-bednosnih uređaja i vizuelnu kontrolu mašine. Značajno unapređenje bi se

postiglo kroz automatizovano podešavanje-skaliranje transmitera pritiska, regulatora pritiska i protokomera prilikom promene tipa prethodno navedenih uređaja. Unapređenjem nebi bilo potrebno menjati program PLC-a sa tačke gledišta nominalnih vrednosti uređaja, već unošenjem u korisnički interfejs program automatski vrši izmene na osnovu unetih vrednosti zamenskih uređaja. Instalacijom dodatna tri protokomera bi se ubrzavao proces ispitivanja, jer se protok svih cevi meri istovremeno za razliku od trenutnog pojedinačnog ispitivanja. Vidljivo unapređenje bilo bi i dodavanje senzora za prisutnost ispitnih cevi u prihvatnom alatu čime bi se povećao nivo bezbednosti na radu.

6. ZAKLJUČAK

Porastom proizvodnje delova za automobile, posebno cevi i spojeva (za vazduh, gorivo, ulja itd.) raste i potreba za ispitnim stanicama za ispitivanje ispravnosti ovih cevi. Ispitna stanica za testiranje cevi omogućava potvrdu zahtevanih karakteristika cevi od strane proizvođača automobila. Rezultati testiranja se mogu davati kao garancija krajnjem kupcu, da je cev ispitana i da rezultati dobijeni testiranjem odgovaraju standardu. Ispitna stanica analizirana u ovom radu u svakom pogledu zadovoljava zahteve tržišta. Obuhvata veliki opseg pritiska ispitivanja i tipa ispitivanja. Realizovan način rada ispitne stanice je univerzalan, i uz male mehaničke i pneumatske prepravke, može zadovoljiti bilo koji pritisak, količinu cevi i brzinu procesa ispitivanja.

7. LITERATURA

- [1] Innomatec GmbH, Instruction manual LTC-602 V1.8.2 EN, Innomatec, 2013.
- [2] PeLCer doo, Pressure leak tester korisničko uputstvo, PeLCer doo, 2016.
- [3] Bosch group Rexroth, Aluminium structural framing system, Bosch group, 2008.
- [4] Etem, Industrial aluminium profiles, Etem, 2013.
- [5] Item, MB bulding kit system, Item, 2010.
- [6] Federico M. Mazzolani, Design of Aluminium Structures, University of Naples "Federico II", Italy, 2008.
- [7] A.Dedić, Pneumatika i hidraulika u drvnoj industriji, 2003.
- [8] Radoslav Korbar, Pneumatika i hidraulika, Veleučilište u Karlovcu, 2007.

Kratka biografija:



Viktor Udovički rođen je u Senti, 1990. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva - Energetska elektronika i električne mašine odbranio je 2016. god.



Dragan Miličević rođen je u Tuzli 1977. god. Doktorirao je na Fakultetu tehničkih nauka 2014. godine. Oblast interesovanja su elektromotorni pogoni, energetska elektronika i obnovljivi izvori električne energije.

KVANTIFIKACIJA SIGNALA MR PERFUZIJE MR PERFUSION SIGNAL QUANTIFICATION

Davor Rukavina, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U medicini pojam perfuzije odnosi se na protok krvi kroz male krvne sudove u organima i tkivima. Jedna od tehnika kojom se ovaj fenomen može vizualizovati i kvantifikovati jeste tehnika magnetne rezonance (MR). U ovom radu objasnjen je postupak DSC (Dynamic Susceptibility Contrast)-MR i način izdvajanja karakterističnih parametara od kliničkog značaja: CBV (Cerebral Blood Volume), CBF (Cerebral Blood Flow), MTT (Mean Transit Time) i TBP (Time to Bolus Peak) na osnovu dekonvolucije ulazne arterijske funkcije (AIF-Arterial Input Function) i signala perfuzije u moždanom tkivu. Postupak estimacije karakterističnih parametara realizovan je sa i bez predobrade Savitzky-Golay filtrom na 24 dostupna signala DSC-MR mozga uz odgovarajuću ulaznu arterijsku funkciju.

Abstract – In medicine perfusion assumes delivery of blood to the tissues and organs through capillary beds. Magnetic resonance (MR) is a techniques most commonly used for visualisation and quantification of this phenomenon. In this paper DSC (Dynamic Susceptibility Contrast)-MR perfusion and methods for estimation of clinically significant parameters: CBV(Cerebral Blood Volume), CBF(Cerebral Blood Flow), MTT(Mean Transit Time) and TBP (Time to Bolus Peak) using deconvolution of arterial input function (AIF) and brain perfusion signals are explained. Estimation of characteristic parameters was done with and without Savitzky-Golay filter as preprocessing step on 24 brain perfusion signals and corresponding AIF.

Ključne reči: MR perfuzija, DSC-MRI, dekonvolucija, Savitzky-Golay filter,

1. UVOD

Sama reč perfuzija označava prolaz tečnosti kroz limfni sistem ili prolaz krvi kroz krvne sudove do organa ili tkiva. Perfuzija je fiziološki uslovljen parametar, zahvaljujući kojem možemo da pratimo dotok krvi do nekog organa, što je nužno za otkrivanje tumora i vaskularnih oboljenja. Jedna od prednosti MR perfuzije, za razliku od metoda na bazi X-zraka, je ta što magnetna rezonanca ne podrazumeva izlaganje štetnom zračenju. Takođe, MR perfuzija pruža odličan kontrast mekih tkiva i dobru prostornu rezoluciju.

MR perfuziono snimanje može se realizovati na dva načina:

- tehnika praćenja bolusa kontrastnog sredstva
- tehnika bez upotrebe bolusa kontrastnog sredstva

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Olivera Šveljo.

U prvu grupu spadaju *Dynamic Susceptibility Contrast* (DSC)-MRI i *Dynamic Contrast-Enhanced* (DCE)-MRI, dok u drugu grupu spada *Arterial Spin-Labeling* (ASL). DSC-MRI se u ovom trenutku koristi isključivo kod ispitivanja perfuzije u mozgu. Kod ove metode se kontrastno sredstvo daje pacijentu intravenski, i meri se gubitak signala prilikom prolaska bolusa kroz tkivo koristeći T_2^* ili T_2^* slike [1].

U ovoj tehnici koristi se spoljašnje paramagnetsko kontrastno sredstvo na bazi gadolinijuma koje se ubrizgava intravenski. U zdravom mozgu, u kome je barijera između moždanog tkiva i vaskularnog sistema očuvana, kontrast ostaje lokalizovan unutar krvnih sudova. Moždana vaskulatura zauzima mali procenat od ukupne zapremine mozga, tako da se stvara razlika u susceptibilnosti koja dovodi do gubitka MR signala u okolnom tkivu (T_2^* efekat). Kod ove metode se meri perfuzija u organu tako što se prati kako se ubačeni kontrast raspoređuje dok prolazi kroz vaskularni sistem. Dinamički se prati prvi prolaz kontrastnog agensa kroz moždano tkivo i moždane arterije i ako kontrastno sredstvo brzo prođe kroz tkivo, to označava veliki protok krvi, odnosno veliku perfuziju. Ako prolazi sporo, to označava mali protok krvi. Na slici 1.1 data je grafička ilustracija DSC-MRI eksperimenta.

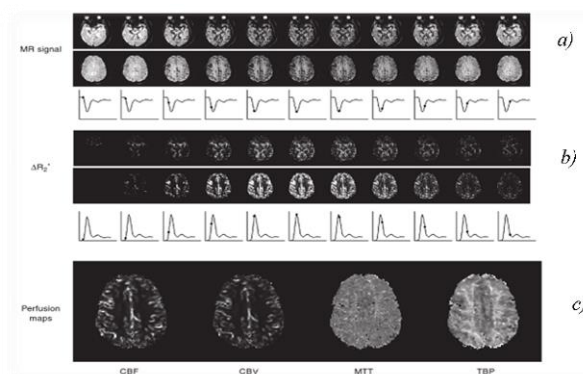
Na slici 1.1 a) date su T_2^* slike tokom prolaska kontrastnog sredstva kroz velike arterije i moždano tkivo kao i vremenska zavisnost jačine signala tokom prolaska bolusa. Na slici 1.1b date su slike na osnovu ΔR_2^* signala koje su proporcionalne koncentraciji bolusa u arterijama i tkivu (slika 1.1b dole). Na slici 1.1c date su mape karakterističnih parametara:

CBF - *Cerebral Blood Flow* (protok krvi u vremenu)

CBV - *Cerebral Blood Volume* (zapremina krvi)

MTT - *Mean Transit Time* (srednje vreme prolaska)

TBP - *Time-to-Bolus-Peak* (vreme kada je dostignut maksimum koncentracije)

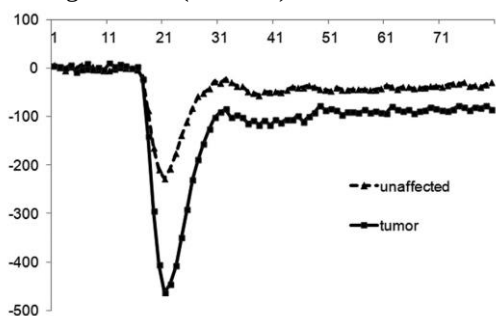


Slika 1.1: Prikaz klasičnog DSC-MRI eksperimenta i rezultujućih mapa karakterističnih parametara

Ove četiri slike (1.1c) dobijaju se ekstrakcijom perfuzionog signala sa MR slika i izračunavanjem karakterističnih parametara ovog signala. U ovom radu demonstriran je postupak ekstrakcije karakterističnih parametara od kliničkog značaja kod DSC-MR tehnike perfuzionog snimanja.

2. DINAMIKA SIGNALA DSC-MR PERFUZIJE

Kod DSC-MRI modela za perfuzionu kvantifikaciju koriste se fizički principi koji važe za nedifuzne obeleživače i oslanja se na pretpostavku da kontrastni materijal ostaje intravaskularan u prisustvu zdrave krvno-moždane barijere (BBB – Blood Brain Barrier). Kod DSC-MRI intenzitet signala odlikava hemodinamiku kontrastnog sredstva (slika 2.1)



Slika 2.1: Tipičan signal MR perfuzije kod DSC-MRI tehnike u normalnom i patološkom tkivu

Dok prolazi kroz mikrovaskulaturu, bolus kontrastnog sredstva izaziva smanjenje u jačini MR signala. Dobijene slike možemo podeliti na tri dela: referentne, slike tokom prvog prolaska bolusa i one snimljene za vreme recirkulacije. Referentne slike se snimaju pre dolaska bolusa i signal je vremenski nezavisan. Tokom prvog prolaska bolusa kontrastnog sredstva, bolus stiže u vokal i DSC-MRI signal se brzo smanjuje do svoje najmanje vrednosti (što odgovara vremenu kada je bila najveća koncentracija kontrastnog sredstva). Nakon što signal dostigne minimalnu vrednost, jačina signala se delimično vraća na referentnu vrednost. Tokom perioda recirkulacije (koji se često delimično preklapa sa vremenom prvog prolaska bolusa), DSC-MRI signal ponovo slabi (ali veoma malo i sa malom stopom), zbog ponovnog ulaska kontrastnog sredstva. Posle ovog perioda signal se teoretski vraća na referentnu vrednost [1].

3. ODREĐIVANJE KARAKTERISTIČNIH PARAMETARA PERFUZIJE

Kao što je već rečeno kod DSC-MRI se stvara razlika u susceptibilnosti koja dovodi do gubitka MR signala u okolnom tkivu ($T2^*$ efekat). Koncentracija kontrastnog sredstva u tkivu, C , je proporcionalna promeni stope relaksacije ΔR_2 :

$$\Delta R_2 = k \cdot C \quad (1)$$

dok je odnos između signala $S(t)$ i stope relaksacije ΔR_2 opisan sledećom formulom:

$$S(t) = S_0 e^{-TE \Delta R_2(t)} \quad (2)$$

gde je TE vreme eha MR impulsne sekvence, a S_0 je referentni signal, odnosno jačina signala bez prisustva kontrastnog sredstva. Iz prethodne dve formule $C(t)$ se može izraziti na sledeći način:

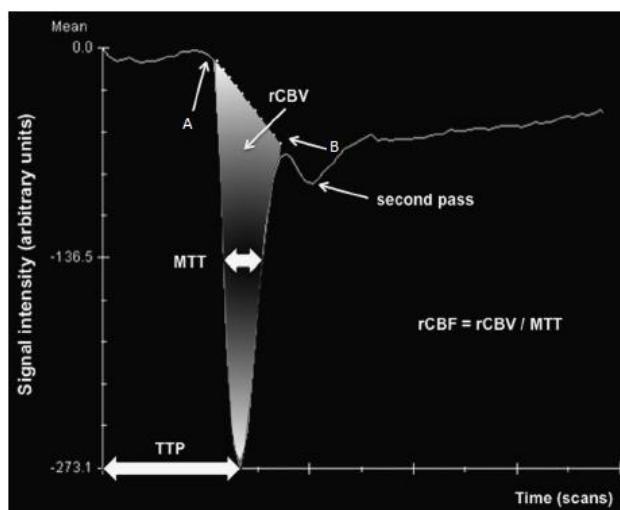
$$C(t) = -\frac{1}{k \cdot TE} \ln \left(\frac{S(t)}{S_0} \right) \quad (3)$$

$C(t)$ predstavlja koncentracijsko-vremensku krivu kontrastnog sredstva tokom vremena t , dok k predstavlja nepoznatu konstantu proporcionalnosti koja se u daljem izračunavanju skraćuje. Ako pretpostavimo uniformnu arterijsku koncentraciju kod svih arterijskih ulaza, relativna CBV merenja su određena integracijom površine ispod koncentracijsko-vremenske krive. Da bi se izračunao CBV, potrebno je odrediti odnos površina ispod koncentracijsko-vremenskih krivi tkiva i arterija na sledeći način

$$CBV = k_H \cdot \frac{\int C(t) dt}{\int AIF(t) dt} \quad (4)$$

gde $C(t)$ predstavlja koncentracijsko-vremensku krivu tkiva dok AIF predstavlja koncentracijsko-vremensku krivu arterije, tj. arterijsku ulaznu funkciju. Ovde je k_H korektivni faktor i zavisi od vrednosti hematokrita u krvi i gustine mozga [2].

Na slici 3.1 vidimo izgled tipičnog signala MR perfuzije, na kome su označeni karakteristični parametri. Bitno je da znamo koje su gornje i donje granice integrala prilikom izračunavanja CBV-a, a za to postoje tri metode, odnosno tri pravila postavljanja granica integrala. Zajedničko za sve metode je da se prvi prolazak bolusa postavi tako da počne i da se završi u svim pikselima u isto vreme. Kod prve metode donju granicu integrala postavljamo u tačku A, odnosno poslednji trenutak pre prvog prolaska bolusa. Gornju granicu postavljamo u tačku B, odnosno na mesto vizuelno najmanje koncentracije nakon prvog prolaska bolusa, ali pre drugog prolaska, odnosno recirkulacije. Kod druge metode donja granica integrala je ista kao kod prve metode, ali se integrali i dalje, tokom trajanja recirkulacije. Integracija krive nakon prvog prolaska bolusa povećava SNR, ali nepravilnosti zbog recirkulacije mogu da unose veći nivo greške kod određivanja apsolutnih vrednosti. Osim toga vrednost CBV moguće je odrediti i kao integral funkcije impulsnog odziva tkiva, što se smatra jednom od najboljih tehnika zbog čega je korišćena u ovom radu.



Slika 3.1. Tipična kriva signala MR perfuzije sa karakterističnim parametrima

Naime, ukoliko uvedemo funkciju $R(t)$ koju nazivamo i funkcijom ostatka [3], kao opadajuću funkciju kojom se izražava količina kontrastnog sredstva prisutnog u vaskularnom sistemu nakon vremena t , prema izrazu:

$$R(t) = 1 - \int_0^t h(\tau) d\tau \quad (5)$$

gde $h(t)$ označava gustinu verovatnoće kapilarnih tranzitnih vremena. Za $R(t)$ važe sledeći izrazi :

$$R(0) = 1 \text{ i } R(\infty) = 0 \quad (6)$$

Koncentracionu krivu tkiva sada možemo izraziti na sledeći način :

$$C_t(t) = CBF \cdot C_a \cdot R(t) \quad (7)$$

Proizvod $CBF \cdot R(t)$ se zove impulsni odziv tkiva, jer se C_a smatra kao ulazna funkcija koja teži nuli (AIF). Međutim, u praksi postoji određena disperzija AIF-a, tako da AIF dolazi kao funkcija „raširena u vremenu“. Zbog toga se koncentraciona kriva tkiva izražava preko konvolucije impulsnog odziva i AIF-a (AIF se posmatra kao suma individualnih veoma kratkih impulsa) na sledeći način:

$$\begin{aligned} C_t(t) &= CBF \cdot R(t) \otimes AIF(t) \\ &= CBF \cdot \int_0^t AIF(\tau) R(t-\tau) d\tau \end{aligned} \quad (8)$$

gde $\otimes$ označava konvoluciju [3]. Kada se izabere odgovarajući AIF, dekonvolucijom se može odrediti CBF. Pošto je $R(0) = 1$, CBF bi trebao da bude jednak početnoj visini impulsnog odziva tkiva ($CBF \cdot R(t)$). Ovo važi samo u slučaju kada nema kašnjenja i disperzije. MTT se određuje prema teoremi centralne zapremine kao :

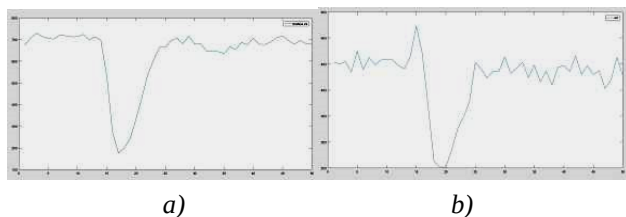
$$MTT = \frac{CBV}{CBF} \quad (9)$$

dok se za TBP uzima trenutak dostizanja maksimuma funkcije impulsnog odziva tkiva.

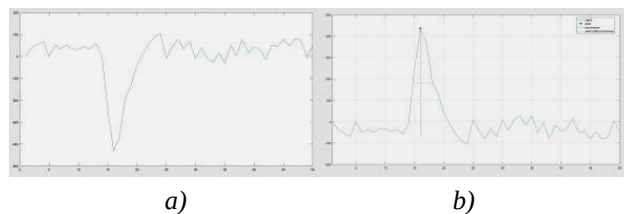
3.1. Klinički podaci

U ovom radu korišćeno je 24 signala DSC-MR perfuzije iz različitih moždanih zona i odgovarajuća arterijska ulazna funkcija subjekta snimljenog na uređaju Magnetom Aera 1.5T u Centru za imaging dijagnostiku, Instituta za onkologiju u Sremskoj Kamenici. Studija je odobrena od strane Etičkog odbora Instituta za onkologiju u Sremskoj Kamenici. Na slici 3.2 dat je reprezentativni primer signala perfuzije i odgovarajuće arterijske ulazne funkcije.

Za određivanje impulsnog odziva tkiva korišćena je metoda deljenja polinoma (slika 3.3 a) a potom je identifikovan maksimum i pozicija maksimuma impulsnog odziva tkiva (slika 3.3 b).

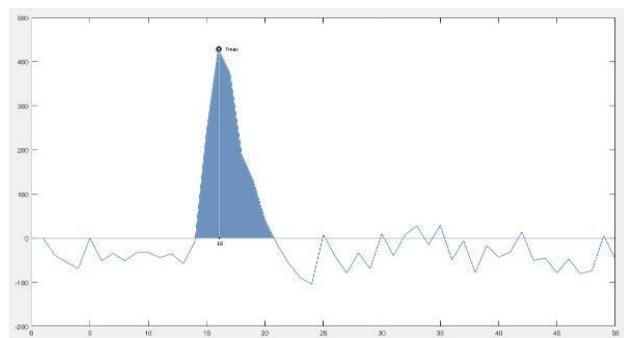


Slika 3.2. Reprezentativni signal perfuzije (a) i odgovarajuća arterijska ulazna funkcija-AIF (b)



Slika 3.3. Impulsni odziv reprezentativnog signala sa slike 3.2. (a). Određivanje CBF na invertovanom signalu impulsnog odziva tkiva (b)

Tako dobijena maksimalna vrednost u odnosu na x osu uzeta je kao vrednost protoka krvi CBF dok je pozicija maksimuma označena kao vrednost TBP (slika 3.3). Zapremina krvi CBV određena je kao integral funkcije impulsnog odziva tkiva tako da obuhvati maksimalne vrednosti odziva (slika 3.4).



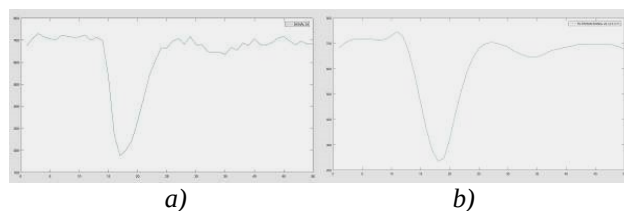
Slika 3.4. Određivanje CBV kao integrala funkcije impulsnog odziva tkiva za signal sa slike 3.2

Srednje vreme prolaska MTT određeno je kao količnik CBV i CBF. Isti postupak ponovljen je za signale perfuzije na koje je prethodno bio primenjen Savitzky-Golay filter koji je odabran jer poboljšava odnos signal šum dok maksimalne vrednosti ostaju očuvane. Analiza je realizovana u programskom paketu Matlab

4. REZULTATI I DISKUSIJA

U Tabeli 1 date su vrednosti CBF, CBV, MTT i TBP, za sve dostupne signale. Signali S15, S 16 izdvojili su se kao signali sa niskim vrednostima CBV i CBF dok je signal S22 pokazao visoke nivoe za oba parametra. Vrednost MTT bila je najniža za signale 15 i 16 dok je najviša vrednost ovog parametra zabeležena za S22. Vrednost TBP bila je približno jednaka za sve posmatrane signale dok su nesto veće vrednosti dobijene za signale 3,5 i 22.

U Tabeli 2 date su vrednosti CBV, CBF, MTT i TBP za signale na koje je predhodno primenjen Savitzky-Golay filter (slika 4.1). Za parametre reda i dužine prozora odabarne su vrednosti 9 i 11 respektivno kao najpogodnije



Slika 4.1. Reprezentativni signal perfuzije pre (a) i nakon predobrade Savitzky-Golay filtrom (b)

Tabela 1. Vrednosti karakterističnih parametara na signalima bez primene filtra

	CBF	CBV	MTT	TBP
S 01	58.72	104.24	1.78	15
S 02	119	218.45	1.84	15
S 03	100.26	569.85	5.69	25
S 04	102.44	180.57	1.77	15
S 05	111.44	452.3	4.06	25
S 06	178.07	390.69	2.2	16
S 07	309.06	828.96	2.69	16
S 08	207.85	1035.38	4.99	16
S 09	174.9	767.49	4.39	16
S 10	91.3	238.97	2.62	15
S 11	531.61	1786.97	3.37	16
S 12	110.87	262.26	2.37	15
S 13	95.15	149.3	1.57	15
S 14	62.93	131.75	2.1	15
S 15	16.05	35.08	2.19	6
S 16	8.65	0.3	0.04	15
S 17	229.89	763.26	3.33	16
S 18	437.93	1489.89	3.41	16
S 19	124.63	390.38	3.14	15
S 20	425.1	1259.93	2.97	16
S 21	97.99	197.03	2.02	15
S 22	378.74	3509.75	9.27	25
S 23	74.28	101.75	1.37	15
S 24	429.73	1397.11	3.26	16

Vrednosti CBF i MTT dobijene sa i bez filtra nisu se značajno razlikovala ($p > 0.1$) dok su vrednost CBV i TBP sa i bez primene filtra u predobradi pokazale statistički značajne razlike ($p < 0.1$).

Oba postupka ukazala su na signale 15 i 16 kao na zone niske perfuzije odnosno zone u kojima u posmatranom vremenskom opsegu nije došlo do značajnije promene signala dok su vrednosti parametara perfuzije signala S22 ukazale na potencijalno patološki izmenjeno tkivo sa relativno visokom vrednošću CBV i umerenim vrednostima CBF i MTT.

5. ZAKLJUČAK

Kvantifikacija signala MR perfuzije podrazumeva modelovanje procesa perfuzije odnosno protoka krvi kroz male krvne sudove i određivanje karakterističnih, klinički relevantnih parametara.

U ovom radu korišćen je metod određivanja karakterističnih parametara na osnovu impulsnog odziva tkiva što je jedan od najčešće korišćenih postupaka. Savitzky-Golay filter korišćen u predobradi pokazao se kao pogodan metod za potiskivanje šuma posebno kada se radi o određivanju CBV i MTT.

Tabela 2. Vrednosti karakterističnih parametara signala perfuzije uz predobradu Savitzky-Golay filtrom

	CBF SG	CBV SG	MTT SG	TBPSG
S 01	57.28	104.71	1.83	16
S 02	116.25	216.04	1.86	15
S 03	111.43	571.55	5.13	25
S 04	96.95	179.7	1.86	15
S 05	95.91	452.22	4.72	25
S 06	166.45	391.32	2.36	16
S 07	289.29	826.18	2.86	16
S 08	209.11	1029.11	4.93	17
S 09	172.45	770.42	4.47	16
S 10	95.33	238.5	2.51	15
S 11	494.14	1785.78	3.62	16
S 12	112.63	261.57	2.33	15
S 13	87.29	149.4	1.72	15
S 14	63.78	131.84	2.07	15
S 15	12.03	36.9	3.07	11
S 16	6.83	7.01	1.03	3
S 17	224.64	765	3.41	16
S 18	415.76	1487.11	3.58	16
S 19	136.48	376.7	2.77	16
S 20	396.75	1259.2	3.18	16
S 21	104.07	198.72	1.91	15
S 22	367.43	3510.16	9.56	25
S 23	86.18	101.91	1.19	15
S 24	420.68	1398.07	3.33	16

7. LITERATURA

- [1] Geon-Ho Jahng, Ka-Loh Li, Leif Ostergaard, Fernando Calamante, "Perfusion magnetic resonance Imaging: A comprehensive update on principles and techniques" Korean J Radiol 2014; 15(5):554-577
- [2] Bianca Maan. "M.Sc. Thesis: Comparison of methods for quantitative analysis of dynamic susceptibility contrast enhanced brain perfusion MRI", University of Twente, Department of Electrical Engineering, Nov 2010.
- [3] L. Knutsson, F. Stahlberg, and R. Wirestam. "Absolute quantification of perfusion using dynamic susceptibility contrast MRI: pitfalls and possibilities." MAGMA, 23(1):1–21, Feb 2010.

Kratka biografija:

Davor Rukavina rođen je u Sremskoj Mitrovici 11.05.1990. godine. Studije je upisao 2009. godine u Novom Sadu na Fakultetu tehničkih nauka, smer Energetika, elektronika i telekomunikacije. Studije je nastavio na Katedri za telekomunikacije i obradu signala, da bi se potom usmerio na telekomunikacione sisteme. Master studije upisuje 2014. godine na Katedri za telekomunikacije i obradu signala, grupa Telekomunikacioni sistemi.

PROJEKTOVANJE I ANALIZA BAZNIH STANICA U LTE MREŽAMA

DESIGN AND ANALYSIS OF BASE STATIONS IN LTE NETWORKS

Vladimir Kosjer, Željko Trpovski, Dejan Nemec, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu je opisana evolucija mobilnih mreža od prve analogne generacije do LTE i LTE-Advanced mreža četvrte generacije. Predstavljene su nove tehnologije koje su zaslužne za razvoj mreža četvrte generacije. Urađena je analiza i projektovanje bazne stanice za LTE mrežu. Nakon toga, izvršeno je merenje nivoa signala na prijemu i nivoa magnetskog polja u blizini bazne stanice i bez nje. Finalno, napravljena je analiza uticaja nejonizujućeg zračenja na čoveka.

Abstract – This paper contains evolution of mobile communication network since first analogue generation until LTE and LTE-Advanced networks of 4th generation. New technologies which led us to 4G are represented and described. Base station for LTE network is analyzed and designed. Test measurements of received signal level, as well as magnetic field level are performed, near the base station, and far from it. In the end, the analysis of non-ionizing radiation on humans is made.

Cljučne reči: Bazna stanica, 4G, LTE i LTE-Advanced mreža, OFDM, SC-FDM, MIMO

1. UVOD

Od samog nastanka govora, ljudi su stremili brzom, preciznom i efikasnom prenosu poruka na velikim udaljenostima. Od tog perioda do današnjeg modernog doba, komunikacija je postala i ostala ključ preživljavanja i napretka društva.

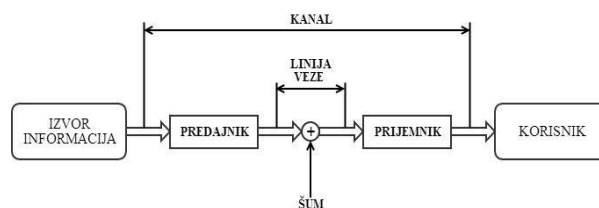
Začetnikom bežične komunikacije može se smatrati Mihael Faradej, koji je 1830. godine eksperimentalno pokazao da postoji veza između elektriciteta i magnetizma. Nakon otkrića tranzistora 1947. godine, kao i prvog integrisanog kola, dolazi do primetnog smanjivanja komponentenata (antena i kola), što dovodi do razvoja prototipa mobilnog telefona 1973. godine. Pronalazač Martin Kuper, smatra se osobom koja je napravila prvi poziv i to konkurentskoj kompaniji Bell Labs.

U početku je razvoj mobilnih komunikacija bilo pitanje od nacionalnog značaja, dok su sa usloznavanjem sistema, brigu o razvoju standarda preuzele međunarodne organizacije, kao što su Evropski institut za telekomunikacione standarde (ETSI – European Telecommunications Standards Institute), Međunarodna telekomunikaciona unija (ITU – International Telecommunication Union) i Partnerski projekat treće generacije (3GPP – Third Generation Partnership Project).

NAPOMENA:

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

Tehnologije za mobilne komunikacije često su podeljene po generacijama. Ono što je zajedničko za sve generacije, kao i sve ostale telekomunikacione sisteme, jeste da se mogu predstaviti Šenonovim generalnim modelom, prikazanim na slici 1.



Slika 1. Model telekomunikacionog kanala [1]

2. POTREBA ZA UNAPREĐENJEM SISTEMA

Dugi niz godina, glasovni pozivi bili su dominantna usluga u mobilnom saobraćaju. Prenos podataka je u početku sporo rastao. Međutim, povećanjem brzine prenosa podataka, naročito nakon pojave pametnih telefona (oko 30% prodatih uređaja u 2011. godini bili su pametni telefoni [2]), potražnja za ovom uslugom počela je naglo da raste. Prema istraživanjima kompanije Erikson, u periodu od januara 2007. godine do jula 2011. godine, količina prenetih podataka rasla je za 18% na tromesečnom nivou, da bi za celokupni period taj rast iznosio više od sto puta. Rast je naročito primetan od 2008. godine, nakon što su u proizvodnju krenuli uređaji bazirani na Android operativnom sistemu kompanije Gugl.

Kao rezultat naglog rasta saobraćaja za prenos podataka, dolazi do prezasićenja 2G i 3G mreže, čije su maksimalne brzine išle do 84 Mb/s kod HSPA+ standarda (eng. High Speed Data Access). Ova dva sistema više ne mogu da odgovore na sve veće zahteve korisnika, što je jasan znak da je postojala realna potreba za povećanim kapacitetom mreže.

Šenon je već 1948. godine otkrio teorijski limit za prenos podataka koji može biti postignut bilo kojim komunikacionim sistemom [3]. Najjednostavniji izraz za kapacitet sistema, nazvan Šenon-Hartlijeva teorema, prikazan je sledećim izrazom:

$$C = B * \log_2(1 + SINR) \quad (1)$$

gde su:

- SINR – odnos snage signala u odnosu na snagu usled interferencije i šuma;
- B – širina spektra komunikacionog sistema izražena u Hz;
- C – kapacitet kanala izražen u bit/s, odnosno maksimalni prenos podataka za jednu ćeliju.

3.1. Rešenje problema nedovoljnog kapaciteta postojećih sistema

Prvi bi predstavljala implementacija manjih ćelija. Izgradnjom novih baznih stanica i smanjivanjem zone koja je pokrivena jednom ćelijom, smanjio bi se broj pretplatnika koji toj ćeliji pripadaju, a samim tim došlo bi i do smanjenog saobraćaja po ćeliji. Prema formuli (1) ovo bi dovelo do povećanja kapaciteta celog sistema.

Drugi način bio bi povećanje širine frekvencijskog opsega. Međutim, pored mobilne telefonije, nacionalni spektar su koristili radio i TV, kao i mnoge državne službe, među kojima je vojska, policija i druge. Kako je spektar ograničen, jasno je bilo da postoje granice do koje ovaj proces može da ide.

Treća tehnika, koja se pokazala kao jedina održiva na duže staze, predstavljala je celokupno unapređenje tehnologije koja bi se koristila u budućnosti. Nova tehnologija bila bi razvijana u pravcu boljeg iskorišćenja kapaciteta kanala, naročito na povećanje parametra SINR, što bi ujedno donelo i veće brzine protoka. Na taj način, rešilo bi se nekoliko postojećih problema i došlo bi do izgradnje savremene mreže, koja bi imala veliki kapacitet za dalja unapređenja.

3. 4G TEHNOLOGIJA – LTE/LTE-ADVANCED

3GPP grupa predstavila je svoju tehnologiju koja bi trebalo da nasledi 3G sisteme, pod nazivom LTE (eng. Long Term Evolution) i njenu unapređenu verziju LTE-Advanced. Dosta stvari bilo je nasleđeno iz prethodnog 3GPP rešenja (najviše HSPA+), ali najznačajniji delovi mreže, bili su zamenjeni.

3GPP je rad na standardizaciji LTE tehnologije započeo već 2004. godine, kada je najviše pažnje bilo posvećeno radio interfejsu. Zahtevi koji su stavljeni pred LTE bili su: unapređen kapacitet sistema i pokrivenost signalom, veće maksimalne brzine (preko 100Mb/s), smanjeno kašnjenje, povećana ponuda i smanjena cena, podrška za više prijemnih i predajnih antena, smanjena potrošnja energije i drugi.

Da bi sve ovo bilo moguće, razvijene su različite tehnologije koje su obezbedile LTE standardu da uspešno izvrši ciljeve koji su pred njega postavljeni i to su:

- OFDM/OFDMA
- SC-FDM
- MIMO
- Adaptivni linkovi.

4.1. OFDM

Ključni element koji je doveo do evolucije sistema u LTE jeste korišćenje radio linka baziranog na ramovima ortogonalnog frekvencijskog multipleksa (OFDM – Orthogonal Frequency Division Multiplexing). Praktična implementacija tehnologije različita je na downlink i uplink delu kao rezultat različitih zahteva između ova dva smera komunikacije i opreme koja se koristi na oba kraja. OFDM je izabran za downlink smer (od bazne stanice ka korisniku), zbog svoje velike otpornosti na interferenciju. Takođe, format je veoma pogodan za postizanje velikih brzina prenosa, što je jedna od ključnih komponenata LTE sistema. Prednosti su i velika spektralna efikasnost, robusnost na fejdging, jednostavna implementacija, mala kompleksnost algoritama i druge.

Ideja koja stoji iza OFDM jeste prenos informacija na velikom broju uskopojasnih podnosilaca. Slične tehnologije koriste ih svega nekoliko, kod HSPA+ su to četiri. Kod OFDM, broj podnosilaca koji će biti emitovan preko istog radio linka varira između nekoliko stotina do nekoliko hiljada.

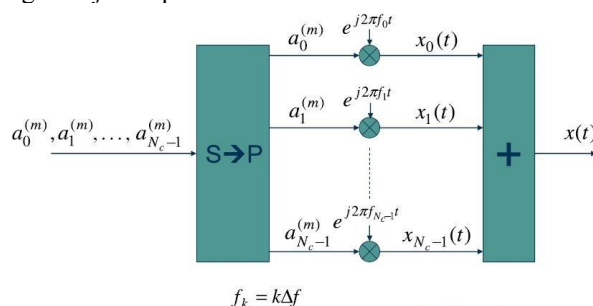
U opštem obliku, vremenski domen OFDM signala tokom određenog vremenskog intervala može se izraziti sledećim izrazom:

$$x(t) = \sum_{k=0}^{N_c-1} x_k(t) = \sum_{k=0}^{N_c-1} a_k^{(m)} e^{j2\pi k \Delta f t} \quad (2)$$

gde su:

- $x_k(t)$ – k-ti modulirani podnosilac frekvencije $f_k = k \cdot \Delta f$;
- $a_k^{(m)}$ – kompleksni modulacioni simbol primenjen na k-ti podnosilac tokom m-tog OFDM simbolskog intervala;
- N_c – ukupan broj modulacionih simbola koji će biti emitovani.

Izgled OFDM modulatora koji je opisan izrazom (2) prikazan je na slici 2. Na ulaz u modulator dolazi niz simbola koji treba da se modulišu, oni ulaze u blok za paralelizaciju i zatim se svaki od simbola konvoluiira sa odgovarajućim podnosiocem.



Slika 2. OFDM modulator [4]

Prilikom prenosa OFDM simbola, dostupna su tri tipa modulacije simbola i to QPSK (2 bit/sim), 16-QAM (4 bit/sim) i 64-QAM (6 bit/sim). Povećanjem reda modulacije, povećava se i brzina radio linka.

Nosioci su gusto smešteni u spektru i rastojanje između njih iznosi $\Delta f = 1/T_u$. Kada se rastojanje odabere, broj podnosilaca određuje se na osnovu procenjenog maksimalnog protoka koji će obezbeđivati željena ćelija.

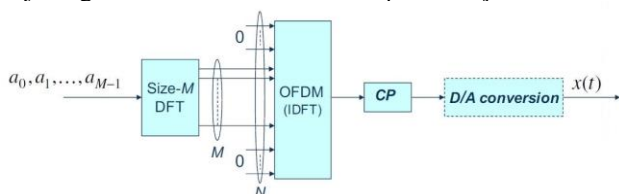
Ideja koja stoji iza OFDM jeste da se svakom korisniku na nivou ćelije, dodeli određeni broj podnosilaca za njegove potrebe. Broj podnosilaca po jednom pretplatniku varira u zavisnosti od kompleksnosti servisa koji se koristi, od svega nekoliko za prenos govora, do nekoliko desetina za zahtevnije servise i aplikacije. Na ovaj način, postoji veliki broj slobodnih podnosilaca, koji se ravnomerno raspoređuje na ostale, pa imamo situaciju da u jednom trenutku može biti uslužen veliki broj korisnika.

4.2. SC-FDM

Dok je većina zahteva koja je potrebna za realizaciju komunikacije od korisnika ka baznoj stanici (uplink) ista kao za downlink, postoje i jedinstveni izazovi koji se pred njom nalaze. Zahtevi koje LTE uplink mora da ispuni su sledeći [5]:

- Ortogonalno emitovanje od strane različitih korisničkih uređaja, zbog smanjenja intercelijske interferencije i povećanja kapaciteta;
- Omogućena podrška za različite brzine slanja podataka, kao i mogućnost da se brzina adaptira promenljivom SINR parametru;
- Dovoljno nizak odnos maksimalne i prosečne snage emitovanog signala PARP (eng. Peak-to-Average Power Ratio), kako bi se izbegli veliki troškovi korisničke opreme i visoka konzumacija energije;
- Sposobnost da se iskoristi frekvencijski diverzitet koji nudi širokopojasni kanal (omogućeno korišćenje opsega širine do 20 MHz), čak i kada se podaci prenose na malim brzinama;
- Podrška za korišćenje više kombinovanih antena, kako bi se povećao kapacitet kanala.

Dizajn SC-FDMA predajnika i prijemnika veoma je sličan onom koji koristi OFDM, ali je PARP parametar znatno bolji, što dovodi do dužeg trajanja baterije i jednostavnijeg dizajna RF pojačavača, a samim tim i cene uređaja. Izgled SC-FDMA modulatora prikazan je na slici 3.



Slika 3. Uprošćen dijagram SC-FDMA modulatora [4]

Najbitnija razlika jeste dodatna diskretna Furijeova transformacija signala na predaji, odnosno inverzna DFT na prijemu. FFT blok koji je realizuje, ubačen je kako bi se minimizovale varijacije u snazi emitovanog signala. Za modulaciju koriste se QPSKA, 16/64QAM.

Postupak demodulacije predstavlja inverznu operaciju onoj na predaji. Prvo se nailazi na FFT blok reda N , zatim se uklanjaju podnosioci koji ne odgovaraju predajnoj sekvenci, i na kraju se dolazi do inverzne DFT transformacije.

4.3. MIMO

Korišćenje MIMO (eng. Multiple Input Multiple Output) tehnike sa više prijemnih i predajnih antena donosi sa sobom mnoštvo unapređenja, koja su potrebna da bi se ambicije LTE sistema uspešno realizovale. Najbitnije je što se dostižu maksimalno projektovane brzine prenosa podataka. LTE standard savršeno uklapa ovaj sistem sa OFDM prenosnom strukturom i maksimalno iskorišćava njihovu kombinaciju. Zbog toga, može se reći da LTE standard zapravo predstavlja MIMO-OFDM sistem [6]. Pored već navedene brzine prenosa podataka, podižu se performanse celokupnog sistema. Čelija sada može da pokriva veću teritoriju (makro ćelije do 40 km), povećan je kapacitet ćelije (makro ćelije preko 2000 korisnika) i povećana je pouzdanost radio linka.

Današnji korisnički uređaji većinom su opremljeni sa dve prijemne antene. Kod baznih stanica broj antena varira u zavisnosti od namene. Brzina prenosa podataka u downlink smeru linearno je proporcionalna broju predajnih antena, te je zbog ove činjenice njihov broj sklon varijacijama. Ukoliko će biti emitovan HSPA+ signal u pitanju su dve antene, dok se četiri i osam antena koristi za LTE i LTE-Advanced, zbog velikih zahtevanih brzina.

4.4. Adaptivni linkovi

Jedna od ključnih karakteristika svakog radio komunikacionog sistema jesu brze i značajne promene mnogih parametara prenosnog kanala. Varijacije u parametrima su česte, a razlozi i uticaj na kanal mogu biti različiti.

Svi oni utiču i na spektralnu efikasnost koja je naročito bitna kod LTE standarda, pošto su uključene različite metode za njeno povećanje. Kako bi nivo željene spektralne efikasnosti bio konzistentan, primenjuje se tehnika koja dinamički menja parametre sistema u zavisnosti od trenutnih karakteristika prenosnog kanala i poznata je pod nazivom adaptivni link. Pored toga, koristi se i kanalsko planiranje (eng. channel-dependent scheduling). Smisao metode je da na osnovu različitih šablona iz prošlosti, predvidi ponašanje sistema u određenoj situaciji ili trenutku.

Sistem međutim, nema mogućnost da savršeno podesi parametre predaje, usled same prirode varijacija u kvalitetu radio linka, koje su nasumične. Umesto toga, pravilnim postavljanjem parametara, imamo mogućnost da trenutne uslove kanala i korišćenog dela spektra iskoristimo efikasnije. Postignuti rezultati znatno su bolji u odnosu na slučaj da su parametri fiksirani na predefinisane vrednosti.

4. PROJEKTOVANJE BAZNE STANICE

Projektovanje bazne stanice za LTE mrežu rađeno je u saradnji sa kompanijom „Telekom Srbija“ AD, izvršna jedinica Novi Sad, zaduženom za radio planiranje i pokrivenost mrežom na teritoriji Vojvodine. Način projektovanja koji je opisan u radu, kao i oprema koja je korišćena, izabrani su tako da odgovaraju realnim zahtevima kompanije Telekom Srbija i situaciji na terenu. Za dodelu frekvencija mobilnim operaterima zaduženi su RATEL (Regulatorna agencija za elektronske komunikacije i poštanske usluge) i odgovarajuće resorno ministarstvo. Opsezi koji su kompaniji „Telekom Srbija“ dodeljeni planom raspodele frekvencija za GSM/DCS-1800 radio sistem (DCS – Digital Cellular System, odnosno GSM-1800), prikazani su u tabeli 1.

Radio sistem	Frekvencijski blok	Uplink (MHz)	Downlink (MHz)	Namednjeni kanali	Broj kanala
GSM-900	3	904.3-913.9	949.3-958.9	72-119	48
DCS-1800	1	1710.1-1720.1	1805.1-1815.1	512-561	50
UMTS/IM T-2000 FDD	1	1920-1935	2110-2125	-	3
UMTS/IM T-2000 TDD	1	1900-1905		-	1

Tabela 1. Frekvencijski opsezi dodeljeni kompaniji „Telekom Srbija“

5.1. Izbor lokacije, izrada plana i projektovanje

Za potrebe ovog rada, odabrana je zgrada na adresi Bulevar oslobođenja 127. Lokacija se činila pogodnom iz nekoliko razloga. Prvi je da predstavlja najviši objekat u blizjoj okolini, krov se nalazi na visini od 66 m (146 m

nadmorske visine). To znači da veća površina neće biti u zoni radio senke, što obezbeđuje veliku pokrivenost koju će buduća ćelija imati. GPS koordinate lokacije su 45°14'36" severne geografske širine i 19°50'31" istočne geografske dužine.

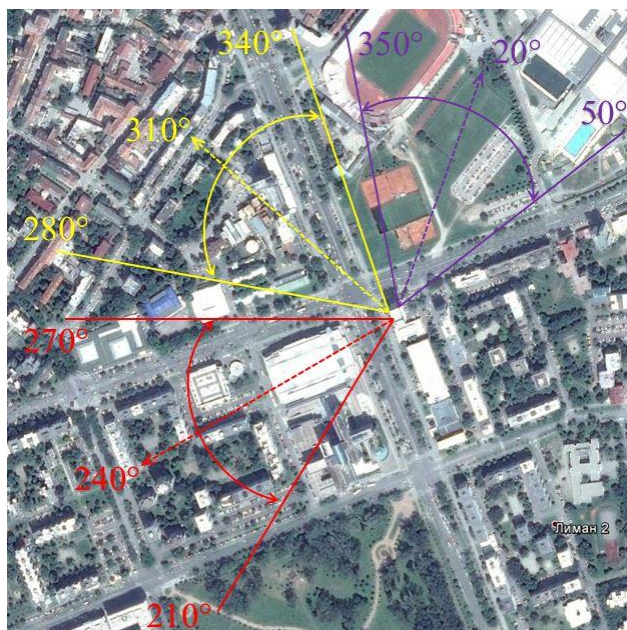
Prilikom projektovanja i planiranja bazne stanice, centralnu figuru predstavljaju parametri antene. Osobine od značaja predstavljaju:

- Visina baze antene - visina donje ivice antene u odnosu na površinu na kojoj se postavlja
- Azimut antene - centralna liniju zračenja antene, određuje se u stepenima u odnosu na sever.
- Sektor antene - površina koja će biti pokrivena zračenjem iz antene. Postoje vertikalni i horizontalni sektor, prave na osnovu dijagrama zračenja antene.
- Mehanički tilt - fizički nagib kućišta antene u odnosu na vertikalnu osu nosećeg elementa antene.
- Električni tilt omogućava daljinsko nagibanje električnih elemenata unutar kućišta antene, što omogućava radio planeru da menja vertikalni nagib i sektor antene, bez izlaska na lokaciju.

Kako bi u potpunosti bio iskorišćen potencijal LTE standarda, potrebno je signal emitovati korišćenjem četiri zračeća elementa. Koristi se model X diversifikacije gde su četiri antene postavljene pod uglom od 45° u obliku slova X, u jednom kućištu. Za bazne stanice „Telekoma Srbije“ na LTE mreži koristi se antena kompanije Kathrein, model 80010510 [7].

Postavljaju se tri antene u tri različita ugla zgrade. Na slici 4. prikazana su tri sektora, jedan za svaku od antena. Za mehanički tilt odabrana je vrednost od 3°, dok je za električni tilt odabrano 2°, radi mogućih korekcija.

Prvi sektor označen je crvenom bojom. Azimut iznosi 240°. Kako antena pravi horizontalni snop širine 60°, granice sektora iznose 210°-270°.



Slika 4. Sektori antena sa odgovarajućim azimutima

Drugi sektor označen je žutom bojom. Azimut iznosi 310°, dok granice sektora iznose 280°-340°. Definiše se zaštitno rastojanje od 10° između dva sektora, usled nesavršenosti dijagrama zračenja, kao i mogućih interferencija.

Treći sektor označen je ljubičastom bojom. Azimut antene iznosi 20°, dok su granice sektora 350°-50°.

Nakon što radio planer završi preliminarne dizajn bazne stanice, posao preuzima građevinski inženjer, čiji je zadatak proračunavanje opterećenja zgrade, kako bi se imao uvid u mogućnosti montaže radio kabineta i sprovođenje napajanja. Energetičar je zadužen za određivanje napajanja od električne mreže do bazne stanice. Za samo napajanje i električne kablove, angažuju se podizvođači. Naposljetku se na licu mesta određuje da li će do kabineta ići optički kabel ili je omogućena optička vidljivost sa radio relejnim linkom RRU (eng. Radio Remote Unit).

5.2. Izrada situacionog plana i stručne ocene

Nakon izbora lokacije za buduću baznu stanicu, ona postaje zona od interesa. Za prijavu i dobijanje dozvole za izgradnju bazne stanice, potrebno je izvršiti nekoliko administrativnih koraka.

Inicijalno se stupa u izradu situacionog plana, koji se predstavlja telu zaduženom za upravljanje objekta (skupština stanara, fizičko ili pravno lice) i dolazi do sklapanja ugovora. Dalje se izrađuje Stručna ocena uticaja na životnu sredinu i sledi slanje zahteva Gradskoj ili Opštinskoj upravi za zaštitu životne sredine. Nakon dobijanja rešenja od uprave, radi se izrada projekta i kreće izgradnja bazne stanice kada joj se dodeljuje jedinstveni kod, odnosno naziv nove bazne stanice.

Bitan parametar Stručne ocene predstavlja nulto merenje nejonizujućeg zračenja na željenom lokalitetu. Ono se vrši pre postavljanja bazne stanice, kako bi se imao uvid u to koliko bazna stanica svojim radom doprinosi ukupnoj količini zračenja na lokaciji od interesa. Prilikom puštanja bazne stanice u rad, pristupa se prvom merenju. Ukoliko se prilikom prvog merenja javi prekoračenje od 10% od dozvoljene količine zračenja, bazna stanica se proglašava za Izvor od posebnog interesa (IPI).

5.3. Merenje

Merenja su vršena mobilnim uređajem Huawei Honor 7, koji je opremljen magnetometrom. Za merenje intenziteta magnetskog polja, korišćena je Android aplikacija pod imenom Physics Toolbox Suite. Prvo merenje izvršeno je na krovu zgrade u Gagarinovoj ulici, bez prisustva bazne stanice i drugih električnih uređaja. Intenzitet magnetskog polja izražen je u mikrotelslama (μT) i on na ovoj lokaciji nije prelazio vrednost od 45 μT. Drugo merenje izvršeno je na krovu zgrade na Bulevaru oslobođenja, na kojoj se nalazi bazna stanica sa tri predajne antene i radio ormarom. Vrednosti koje su ovde pročitane iznose 64 μT. Može se primetiti da je razlika oko 20 μT, što ne predstavlja značajno uvećanje.

Nivo signala meren je pomoću aplikacije pod nazivom Network Cell Info Lite. Mera za snagu prijemnog signala je RSSI (eng. Received Signal Strength Indicator) i izražena je jedinicom decibel-milivat (dBm). Koristi se za merenje apsolutne snage. Signal koji je za 10 dBm niži, slabiji je po snazi deset puta. U tabeli 2. prikazan je kvalitet signala u zavisnosti od snage izražene u dBm.

Prvo merenje izvršeno je iz zgrade koja je od bazne stanice udaljena 200 m, a drugo merenje u neposrednoj blizini bazne stanice. Kao što je i očekivano, nivo signala u blizini bazne stanice izuzetno je dobar i iznosi čak -51 dBm.

RSSI [dBm]	Kvalitet signala
Više od -60	Savršen
Od -61 do -70	Odličan
Od -71 do -85	Dobar
Od -86 do -100	Prihvatljiv
Od -101 do -110	Veoma loš
Manje od -110	Bez signala

Tabela 2. Kvalitet signala u zavisnosti od snage

Sa druge strane, snaga signala u stambenoj zgradi iznosi -86 dBm i spada u klasu signala sa prihvatljivim kvalitetom. Ukoliko poredimo dva navedena merenja lako se dolazi do međusobnog odnosa signala i može se zaključiti da signal koji je izmeren u stambenoj zgradi ima snagu približno hiljadu puta manju od snage signala u neposrednoj blizini bazne stanice.

5. UTICAJ EM POLJA NA ČOVEKA

Milioni ljudi širom sveta koriste mobilne uređaje u svakodnevnom životu. Zbog povećanih potreba, neprekidno se podižu nove bazne stanice. Zbog toga, naučnici, kao i stanovnici širom sveta, zabrinuti su za uticaj koji ova pojava ima na ljudsko zdravlje.

Jedan od najvažnijih parametara koji se bavi dozvoljenim količinama zračenja jeste specifična brzina apsorbovanja energije SAR (eng. Specific Absorption Rate). Definiše se za 1g ili 10g ljudskog tkiva i izražava se u vatima po kilogramu (W/kg). Preporuku o dozvoljenoj meri SAR, doneo je Evropski Komitet za elektrotehničku standardizaciju CENELEC (fr. Comité Européen de Normalisation Électrotechnique), čega se pridržava i Republika Srbija [8]. Ona definiše maksimalni SAR nivo od 2 W/kg za predeo glave i 4 W/kg za ostatak tela.

Epidemiološke studije o izloženosti elektromagnetskom polju radio frekvencija koje je sproveo naučni komitet SCENIHR (eng. Scientific Committee on Emerging and Newly Identified Health Risks), ne pokazuju povećani rizik od dobijanja tumora mozga, kao ni bilo kog drugog tumora regije glave i vrata.

Takođe, proučavan je uticaj izloženosti zračenju na nervni sistem pomoću elektroencefalogram testova. Evidentirani su određeni efekti, koji ipak nisu konzistentni da bi se mogli doneti pouzdani zaključci.

Studija se takođe bavila uticajem na svakodnevni život ispitanika. Istraživanjima nije primećena direktna i pouzdana veza između izloženosti EM polju i kvaliteta života, bilo kod kratkotrajnog ili dugotrajnog izlaganja. Ispitivanja koja su se bavila uticajem na decu, njihov razvoj i ponašanje su ograničena, te se ne može izvesti pouzdan zaključak, dok značajniji dokazi ne postoje. Slično važi i za uticaj na plodnost kod muškaraca [9].

6. ZAKLJUČAK

Ključnu ulogu u mrežama četvrte generacije ima upotreba novih tehnologija, koja je dovela do višestrukog povećanja brzine prenosa podataka, što je bio jedan od glavnih ciljeva. Kapacitet ćelije višestruko je povećan i potreban je manji broj baznih stanica kako bi se zadovoljile rastuće potrebe korisnika.

Sve navedeno, dovelo je do drastičnog smanjenja snage zračenja antena, kako na predajnoj, tako i na prijemnoj strani. Direktna posledica jeste smanjenje izloženosti populacije štetnom zračenju na minimalni nivo. Merenja prikazana u radu, kao i studije vršene na evropskom i svetskom nivou, potkrepljuju sve pomenute činjenice.

Shodno prethodnim navodima, nije utvrđeno da postoji direktna veza između nejonizujućeg zračenja EM polja radio frekvencija i pojave malignih bolesti.

Takođe nije utvrđeno da optimalna upotreba mobilnih telefona, kao i boravak u blizini bazne stanice, mogu dovesti do dugotrajnih štetnih efekata.

7. LITERATURA

- [1] Ž. Trpovski, „Osnovi telekomunikacija“, 2012
- [2] https://www.ericsson.com/res/docs/2012/tmd_report_feb_web.pdf, Ericsson, „Traffic and market data report“, učitano 22.09.2016. godine
- [3] C. E. Shannon, „A mathematical theory of communication“, The Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656, July, October, 1948.
- [4] E. Dahlman, S. Parkvall, J. Skold „4G: LTE/LTE-Advanced for Mobile Broadband“, 2014.
- [5] S. Sesia, I. Toufik, M. Baker, „LTE – The UMTS Long Term Evolution“, 2011.
- [6] H. Zarrinkoub, „Understanding LTE with Matlab“, 2014.
- [7] „790 – 6000 MHz Base Station Antennas, Filters, Combiners and Amplifiers for Mobile Communications“, Kathrein catalogue 2011.
- [8] „Službeni glasnik RS – 2009“, broj 110 od 04.12.2009. godine
- [9] „Potential health effects of exposure to electromagnetic fields (EMF)“, SCENIHR, 2015.

Kratka biografija:



Vladimir Kosjer rođen je u Indiji 1989. godine. Master rad odbranio je 2016. godine na Fakultetu tehničkih nauka iz oblasti Elektrotehnika i računarstvo - Telekomunikacije



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



Dejan Nemec rođen je 1972. god. Diplomirao, specijalizirao i magistrirao je na Fakultetu tehničkih nauka iz oblasti Elektrotehnika i računarstva. Oblast interesovanja su telekomunikacije i obrada signala.

**ALGORITAM ZA PRONALAŽENJE MINIMALNIH BLOKIRAJUĆIH SKUPOVA
ZAŠTITNIH KODOVA SA RETKIM MATRICAMA PROVERE PARNOSTI****FINDING MINIMUM STOPPING SETS OF LOW DENSITY PARITY CHECK CODES**Siniša Gak, Dejan Vukobratović, *Fakultet tehničkih nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj – Pri konstrukciji LDPC (Low Density Parity Check) kodova, jako je bitno testirati konstruisani kod, da bi mogli da znamo da li taj kod zadovoljava određene standarde i da bi, u slučaju potrebe, mogli da dopunimo kod redundantnim bitima, time povećavajući njegovu efikasnost. Dominantan uticaj na mogućnost dekodovanja kodne reči kodovane navedenim kodom ima struktura blokirajućih skupova, tj. veličina minimalnog blokirajućeg skupa za dati kod. U ovom radu proučavamo mogućnost konstruisanja takvog programskog alata, koji bi na efikasan način pronašao sve blokirajuće skupove, koji su manji ili jednaki od neke predefinisane veličine τ . Analiza je radjena s pretpostavkom da je komunikacioni kanal koji prenosi kodne reči binarni kanal s brisanjem bita (BEC) koji su preneti pogrešno.

Abstract – While constructing an LDPC (Low Density Parity Check) codes, it is very important to test the constructed code, so we could know if the constructed code satisfies the criteria that has been set and, if needed, we could add redundant bits, thus improving the code efficiency. The structure that has a dominant influence on a possibility of decoding the code word is a structure of stopping set, namely the size of the minimum length stopping set for a given code. In this paper I have studied the possibility of constructing a software tool that could efficiently find all stopping sets which size is equal or less than some predefined size τ . The analysis has been done using the assumption that communication channel used for transmission of the code words is the binary erasure channel (BEC).

Ključne reči: LDPC kodovi, Blokirajući skupovi, Iterativno dekodovanje, BEC kanal

1. UVOD

Komunikacioni kanal se obično definiše kao skup od tri elementa, ulaznog alfabeta i , izlaznog alfabeta o i, tranzicione verovatnoće $p(i,o)$ definisane za svaki par (i,o) . Verovatnoća $p(i,o)$ predstavlja verovatnoću da ćemo, u slučaju da je na ulazu u kanal bio simbol i iz ulaznog alfabeta, na izlazu dobiti simbol o iz izlaznog alfabeta.

Još 1948. godine Claude E. Shannon je objavio rad [1] u kojem je predstavio teoremu o prenosu informacija preko komunikacionog kanala u prisustvu šuma. U ovom radu, Shannon je pokazao da svaki kanal ima svoj kapacitet takav da je pouzdan prenos podataka moguće postići za

takve brzine prenosa koje su relativno bliske datom kapacitetu. Osnovna prepreka za korišćenje ove teoreme je vreme kao i oprema koji su potrebni za dekodovanje signala oštećenih šumom.

Oslanjajući se na Shannonov rad, Robert Gallager je 1963. godine u svojoj doktorskoj disertaciji [2] predstavio LDPC kodove. LDPC kod je linearni blok kod određen retkom matricom koja ima n kolona i m vrsta. Za razliku od drugih klasa kodova, LDPC kodovi su već opremljeni sa vrlo brzim (probabilističkim) algoritmima za enkodovanje i dekodovanje [3]. Takođe, ovi kodovi prilaze jako blizu kapaciteta kanala za razne vrste kanala, a i pogodni su za praktičnu primenu.

Jedna od osnovnih osobina LDPC kodova jeste ta da dozvoljavaju dekodovanje primljenog koda čak i ako je on pod velikim uticajem šuma. Dekoder, poznavajući osobine koda koji se koristi, je u mogućnosti da ispravi određen broj grešaka u prenosu. Naravno, ovaj broj je ograničen i zavisi od toga koji tačno skup bita u kodnoj reči je pogrešno prenet. Skupovi bita koji, ako su svi pogrešno preneti, blokiraju dekodovanje kodne reči nazivamo blokirajući skup.

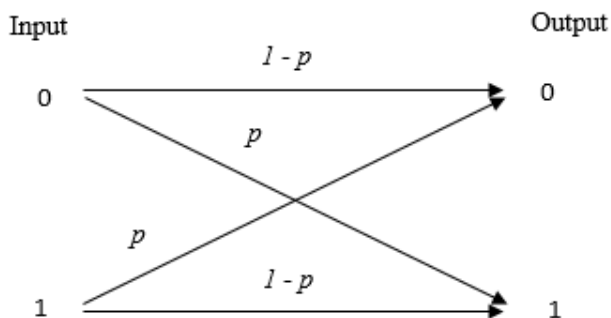
Zadatak ovog rada je konstrukcija programskog alata koji će da pronađe sve blokirajuće skupove čija je dužina manja od unapred definisane maksimalne dužine τ za određeni LDPC kod.

2. BEC KANAL

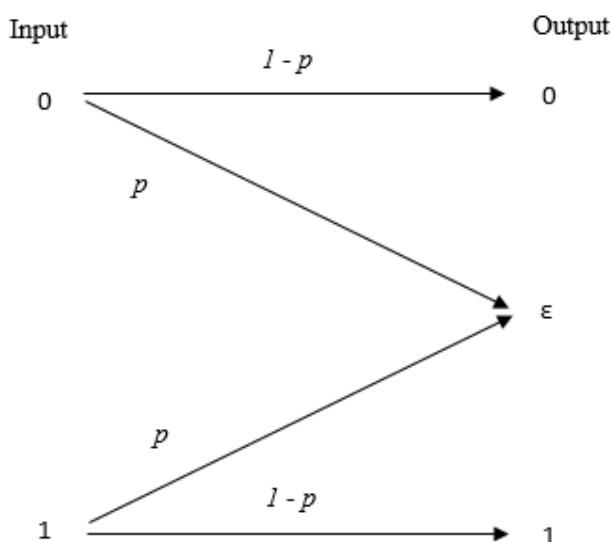
Korišćeni algoritam podrazumeva korišćenje BEC (Binary Erasure Channel) kanala za dekodovanje. Ovaj kanal se često koristi jer u mnogome pojednostavljuje analizu velikog broja problema u teoriji informacija. Za razliku od, recimo lako razumljivog binarnog simetričnog kanala (Slika 1. *Binarni simetrični kanal*) koji je realniji i postavljen tako da se poslati simbol 1 može preslikati u simbol 0 sa verovatnoćom p i u simbol 1 sa verovatnoćom $1 - p$. Takođe, pošto je kanal simetričan, isto važi za ulazni simbol 0 , tj. on se preslikava u simbol 1 sa verovatnoćom p , a u 0 sa verovatnoćom $1 - p$. Ovo znači da je verovatnoća greške za binarni simetrični kanal p . Binarni kanal sa brisanjem na mestu greške (Slika 2. *BEC kanal*) ima ulazni alfabet $\{0, 1\}$, a izlazni $\{0, 1, \varepsilon\}$. Ovde se simbol 0 može preslikati u ε sa verovatnoćom p i u 0 sa verovatnoćom $1 - p$, dok se simbol 1 može preslikati u ε sa verovatnoćom p ili u 1 sa verovatnoćom $1 - p$.

NAPOMENA:

Ovaj rad je proistekao iz master rada čiji mentor je bio Dejan Vukobratović



Slika 1. Binarni simetrični kanal



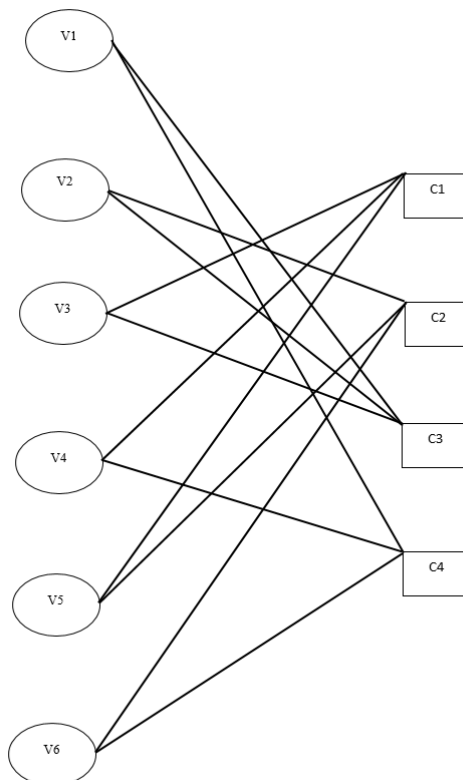
Slika 2. Binarni kanal s brisanjem grešaka u prenosu

3. LDPC MATRICA I TANNER-OV GRAF

LDPC kod je definisan LDPC matricom. To je retka matrica, što znači da je većina njenih elemenata jednaka nuli. Broj kolona ove matrice jednak je dužini kodne reči. Ovakva matrica se može prikazati i kao bipartitni graf, nazvan Tanner-ov graf. Primer jedne takve matrice je dat na slici 3. *Primer LDPC matrice*. Na osnovu ove matrice prikazan je i Tannerov graf (Slika 4. *Tanner-ov graf*). Ovaj graf nastaje tako što na levoj strani nacrtamo onoliko članova koliko imamo kolona u matrici (ove članove zovemo promenjivi članovi), a na desnoj strani onoliko članova koliko matrica ima vrsta (ove članove zovemo članovi za proveru). Veza između i-tog promenjivog člana i j-tog člana za proveru postoji samo ako se u matrici u i-toj koloni j-toj vrsti nalazi 1.

0	0	1	1	1	0
0	1	0	0	1	1
1	1	1	0	0	0
1	0	0	1	0	1

Slika 3. Primer LDPC matrice



Slika 4. Tanner-ov graf

Tanner-ov graf može da se koristi za dekodovanje LDPC kodova. Svaki član za proveru je povezan sa određenim skupom promenjivih članova. Na primer, član za proveru C1 sa slike 4. povezan je sa promenjivim članovima V3, V4, i V5 (možemo primetiti da su u prvoj vrsti matrice sa slike 3, baš na tim pozicijama vrednosti 1). Zbir po modulu dva svih promenjivih članova vezanih za jedan član za proveru mora da bude 0. Recimo za C1 sledi da

$$V3 + V4 + V5 = 0 \quad (1)$$

Ako pretpostavimo da je za prenos korišćen BEC komunikacioni kanal, to znači da vrednosti mogu biti iz skupa $\{0, \epsilon, 1\}$. U ovom slučaju imamo znatno pojednostavljenje kako dekodovanja tako i čitave analize problema. Na primer, ako su za jedan član za proveru poznati svi promenjivi članovi osim jednog (koji je u tom slučaju ϵ) ispravkak ovog simbola se svodi na problem rešavanja jedne linearne jednačine s jednom nepoznatom. Tačnije, simbol koji je stigao kao ϵ je jednak zbiru po modulu 2 svih ostalih simbola.

4. BLOKIRAJUĆI SKUPOVI

Kao što je pokazano u prethodnom poglavlju, dekodovanje kodne reči nad BEC komunikacionim kanalom za LDPC kod je prilično jednostavno. Svodi se na formiranje jednačina provere parnosti za svaki član za proveru. Nakon toga prolazimo iterativno kroz sve jednačine i tražimo one u kojima je samo jedan bit prenet kao ϵ . Određujemo vrednost tog elementa i nastavljamo iteracije. Ovaj proces traje dok svi promenjivi članovi koji su preneti kao ϵ nisu ispravljani, ili dok nam ne ostane takav set nerešenih jednačina provere parnosti da u svim

preostale neresene jednačine imamo po dva ili više brisanih bita. Skup ovakvih bita ili promenljivih članova, koji, ako su preneseni pogrešno (ako su stigli u dekodirer kao ε), blokiraju dekodovanje naziva se blokirajući skup. U [4] data je formalna definicija blokirajućeg seta:

Blokirajući skup S , jeste podskup skupa V , koji je skup svih promenljivih članova, takav da su svi susedi skupa S povezani sa skupom bar dva puta.

Ovo znači da je blokirajući skup S skup bita kodne reči takav da svaka jednačina provere parnosti koja proverava bit u skupu S proverava najmanje dva bita iz S . Broj bita koji se nalazi u blokirajućem skupu naziva se veličina blokirajućeg skupa.

5. EFIKASAN ALGORITAM ZA PRONALAZENJE BLOKIRAJUĆIH SKUPOVA ZA LDPC MATRICU

U [5] je predstavljen efikasan algoritam za pronalaženje svih blokirajućih skupova ograničene veličine korišćenjem mehanizma separacije i evaluacije (engl. Branch and bound algorithm). Algoritam separacije i evaluacije je često korišćen pristup za programsko rešavanje NP – teških problema. U [6] je uveden dodatak na gore spomenuti algoritam koji korake separacije i evaluacije .

Algoritam koristi skupove prinude. Skup prinude je skup uređenih parova (p, s) gde p predstavlja poziciju bita u kodnoj reči, a s njegovu vrednost koja može da bude 0 ili 1. U skupu prinude vrednost bita s označava da li je pretpostavljeno da on pripada blokirajućem skupu (vrednost 1) ili da ne pripada blokirajućem skupu (vrednost 0).

Koraci algoritma su navedeni ispod:

1. Konstruisati listu skupova prinude L i u nju dodati jedan prazan skup prinude.
2. Ako lista L nije prazna izabrati jedan skup prinude F iz L i ukloniti taj skup iz L , u suprotnom algoritam se završava.
3. Ako skup F ima elemenata koliko i kodna reč, i elemenata sa vrednosti 1 manje ili jednako maksimalnoj traženoj veličini blokirajućeg skupa, snimiti blokirajući skup (pozicije sa vrednostima 1 u F) i vratiti se na korak 2.
4. Odraditi iterativno dekodovanje na vrednostima iz skupa F i na osnovu rezultata dekodovanja konstruisati novi skup prinude F' .
5. Ako skup F' ima elemenata koliko i kodna reč, a elemenata sa vrednosti 1 manje ili jednako maksimalnoj traženoj veličini blokirajućeg skupa, i ako je F' validan skup snimiti blokirajući skup (pozicije sa vrednostima 1 u F') i vratiti se na korak 2.
6. Ako ne postoje kontradikcije u dekodovanom skupu F' i ako je donja granica izračunata u koraku evaluacije nad F' manja ili jednaka maksimalnoj traženoj veličini blokirajućeg skupa, onda je potrebno **optimalno** izabrati poziciju koja se ne nalazi u skupu F' i konstruisati dva nova skupa prinude – F'' i F''' . Ovi skupovi se kreiraju tako što se uzmu svi članovi iz F' i na izabranu poziciju se za jedan skup dodaje vrednost 0, a za drugi vrednost 1. Potom se ponovo vraćamo na korak 2.
7. Vratiti se na korak 2.

6. ZAKLJUČAK

Sama implementacija datog algoritma je rađena u C# programskom jeziku uz korišćenje .NET seta programskih alata, u mnogome oslanjajući se na LINQ (Language Integrated Query) biblioteku.

Implementacija se oslanjala na predloženi algoritam u [5] i njegovu optimizaciju u [6]. Pri samoj implementaciji, primećeno je i implementirano par poboljšanja u odnosu na originalno predloženi algoritam. Sam predloženi algoritam ne uzima u obzir mogućnost ponovne analize istih setova prinude (zbog stalnog uklanjanja skupa prinude iz liste L , dovodimo se u mogućnost da ponovo dodajemo isti skup u listu i ponovo ga dekodujemo), kao i mogućnost višestrukog čuvanja istih blokirajućih skupova. Takođe, postoje mogućnosti za dodatna poboljšanja, ako se uzme u obzir da možemo da koristimo definiciju blokirajućeg skupa da proverimo da li je skup blokirajući i na taj način ne moramo da forsiramo dekodovanje čitavog skupa prinude (korak 5 algoritma). Osnovna potreba kreiranja efikasnog alata za pronalaženje minimalnih blokirajućih skupova za LDPC kod je ispunjena. Ovakav alat bi trebao da bude od pomoći dizajnerima ovih kodova, pogotovo imajući u vidu da je minimalni blokirajući skup dominantan pokazatelj kvaliteta dizajniranog koda.

U budućnosti ostaje mogućnost dorade ovakvog algoritma, tako da bi on mogao sam da traži i predlaže redundantne bite za dopunu dizajniranog koda, time povećavajući veličinu minimalnog blokirajućeg skupa, a samim tim i performanse samog koda.

7. LITERATURA

- [1] C. E. Shannon, "A mathematical theory of communication" Bell System Technical Journal, vol. 27, pp. 379 - 423, 1948.
- [2] R. G. Gallager, "Low Density Parity-Check Codes" MIT Press, Cambridge, MA, 1963.
- [3] Amin Shokrollahi "LDPC Codes: An Introduction" Digital Fountain, Inc., April 2, 2003.
- [4] Changyan Di, David Proietti, I. Emre Telatar, Thomas J. Richardson, and Rüdiger L. Urbanke, "Finite-Length Analysis of Low-Density Parity-Check Codes on the Binary Erasure Channel", IEEE Transactions On Information Theory, vol. 48, no. 6, June 2002.
- [5] Eirik Rosnes and Øyvind Ytrehus, "An Efficient Algorithm to Find All Small-Size Stopping Sets of Low-Density Parity-Check Matrices", IEEE Transactions On Information Theory, vol. 55, no. 9, September 2009.
- [6] Eirik Rosnes, Øyvind Ytrehus, Marcel A. Ambroze, and Martin Tomlinson, Addendum to "An Efficient Algorithm to Find All Small-Size Stopping Sets of Low-Density Parity-Check Matrices", IEEE Transactions On Information Theory, vol. 58, no. 1, January 2012.

Kratka biografija:

Siniša Gak rođen je 1988. godine u Zagrebu. Bachelor rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnika i računarstvo – Komunikacione tehnologije i obrada signala, odbranio je 2012. godine.

DIZAJN I VERIFIKACIJA CSI-3 IP MODULA**DESIGN AND VERIFICATION OF CSI-3 IP MODULE**Dejan Križan, *Fakultet tehničkih nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj –Rad predstavlja osnovne tehnike digitalnog dizajna i funkcionalne verifikacije na primeru CSI-3 IP modula. CSI-3 je standardizovan protokol za komunikaciju između senzora kamere i aplikacije, koji je opšte prihvaćen u industriji. Modul poseduje standardne OCP interfejsa za point-to-point komunikaciju sa susednim periferijama. Dodatno, projekat daje uvid u faze razvoja jednog digitalnog dizajna za ASIC tehnologiju pre njegove implementacije na hijerarhijski višem nivou.

Abstract – This paper presents basic techniques of digital design and functional verification taking CSI-3 IP module as an example. CSI-3 is standard protocol designed for the communication between the camera and the host, broadly used in the industry. This module is composed of OCP interfaces intended for the point to point communication with the neighbouring peripherals. In addition, project gives an insight in all steps of the ASIC digital design development before higher level implementation.

Ključne reči: CSI-3, OCP, ASIC, point-to-point, faze razvoja

1. UVOD

Nezavisno od tehnologije, kompleksni digitalni sistemi se sastoje od više povezanih IP modula kojima uglavnom manipuliraju mikroprocesorska jedinica. Svaki IP modul može biti višestruko upotrebljivu funkcionalno srodnim sistemima, što daje veliki značaj njegovom razvoju. Na primeru CSI-3 IP modula biće predstavljeni svi koraci u fazi razvoja jednog takvog digitalnog dizajna.

Početak svakog projekta predstavlja specifikacija modula, koja je polazna tačka za digitalni dizajn i funkcionalnu verifikaciju. Ona mora da sadrži opis svih funkcionalnosti modula, ali i interfejsa koji će se koristiti za komunikaciju sa susednim periferijama sistema. Ovaj dokument piše arhitekta sistema.

Digitalni dizajn se razvija u nekom od HDL jezika što podrazumeva kodovanje sa simulacijama i debugovanjem na RTL nivou. Sledeći korak je sinteza dizajna pri čemu se HDL model prevodi u strukturu sačinjenu od generičkih komponenti. Po završetku dizajna prema specifikaciji modula, potrebno je dokumentovati implementaciju sa svim obrazloženjima. Takav dokument se kasnije koristi za funkcionalnu verifikaciju zajedno sa specifikacijom u cilju realizacije ili nadogradnje test scenarija.

NAPOMENA:

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

Zbog kompleksnosti današnjih IP modula, odnosno broja mogućih stanja, funkcionalna verifikacija je često jedini način provere ispravnosti digitalnog dizajna. Glavni cilj verifikacije je provera konačnog broja funkcionalnosti koja se ogleda kroz pokrivenost HDL koda. U zavisnosti od raspoloživih resursa i vremena, verifikacija se uglavnom realizuje na nivou IP modula ili sistema (SoC). Nezavisno od nivoa i pokrivenosti koda, verifikacija potvrđuje samo skup planiranih funkcionalnosti ali ne i odsustvo bagova. Kvalitetna provera dizajna proističe iz dobre strategije i verifikacionog plana.

2. SPECIFIKACIJA CSI-3 IP MODULA**2.1 Pregled osobina IP modula**

Modul je namenjen za prosleđivanje piksela od CSI-3 transfer nivou video porta. Podaci mogu da dolaze sa senzora kamere pri čemu se bez obrade prosleđuju na video port, ili iz memorije gde se vrši njihovo raspakivanje pre slanja na izlazni port. Softver je zadužen za selektivnu modu rada kao i formata raspakivanja. CSI-3 IP modul poseduje sledeće osobine:

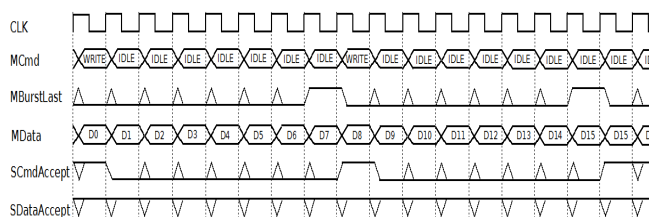
- Maksimalna frekvencija rada je 304MHz za tehnologiju GS80.
- Obrada CSI-3 zaglavlja paketa i oporavak posle preliivanja TX bafera. Detalji o CSI-3 zaglavlju mogu se naći u [2].
- Jedna putanja toka podataka za obradu piksela. Interna procesorska jedinica za raspakivanje piksela podržava 4 formata: raw6, raw8, raw10 i bypass. Kapacitet obrade ove jedinice su 4 piksela po ciklusu.
- Izlazni video port poseduje konfigurabilni takt generator za prilagođavanje izlaznog niza piksela prema zahtevima BYS periferije. Ovaj port poseduje programabilna polja za umetanje horizontalnih i vertikalnih praznina.

2.2 Interfejsi IP modula

Ulazni OCP port služi za prihvatanje podataka sa senzora ili memorije i operativan je na funkcionalnom taktu. Opis signala i faze OCP protokola nalazi se u [1]. Linija podataka ima širinu od 64 bita, a port poseduje sledeća ograničenja:

- Započeta komandna faza upisaburst-a mora biti završena pre početka sledeće. To se postiže deaktiviranjem signala SCmdAccept do početka faze odgovora.
- Maksimalne performanse CSI-3 ulaznog niza su moguće ukoliko slave entitet prihvata podatke u svakom ciklusu. Dakle u fazi odgovora modul mora

biti sposoban da prihvati novu komandnu fazu i postavi odgovor prethodnog transfera kao što je prikazano na slici 2.1.



Slika 2.1. Dva uzastopna burst-a podataka sa različitim oznakama (back to back)

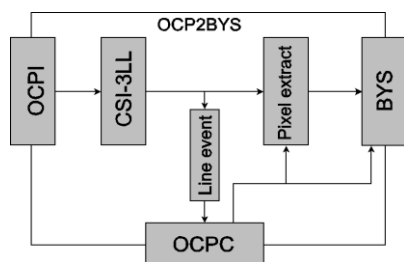
Izlazni video port prosleđuje niz piksela *BYS* periferiji. Zbog sinhronizacije sa pomenutom periferijom pre početka, ali i po završetku svakog *frame*-a šalju se 4 sinhronizaciona piksel takta. Sa svakim taktom port postavlja 4 piksela širine 16 bita na liniji podataka. Za određivanje pozicije piksela u *frame*-a, port poseduje 4 kontrolna signala koji su predstavljeni na slici 2.4. Pri emitovanju praznih piksela port mora da obezbedi piksel takt kao i postavljanje odgovarajućih kontrolnih signala.

Konfiguracioni *OCPC* port zadužen je za konfigurisanje modula i operativan je na polovini funkcionalnog takta.

Modul poseduje: statički port za konfigurisanje takt i reset ćelija, test port i signal za postavljanje prekida.

2.3 Funkcionalni opis *IP* modula

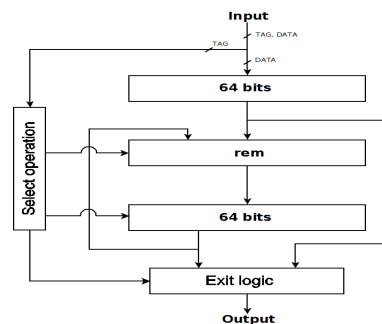
Podaci koji pristižu na ulazni *OCPC* port se prihvataju i prosleđuju *CSI-3LL* bloku. Unutar ove jedinice nalazi se mašina stanja koja obrađuje zaglavlje *CSI-3* paketa, i na osnovu njega formira kvalifikatore za sve podatke. Ovako uređen par podatak-kvalifikator se dostavlja bloku za raspakivanje, koji na osnovu konfiguracije i kvalifikatora vrši ekstrahovanje piksela. Raspakovani pekseli sa pripadajućim kvalifikatorima se prosleđuju izlaznom video portu. Konfiguracioni port sadrži registre sa poljima ekstraktor i *BYS* podmodula. Sa ovim portom interaguje jedinica za detekciju odobrenih prekida koja vrši observaciju *pipeline*-a i internih signala koji su izvori prekida. Blok šema *CSI-3IP* modula prikazana je na slici 2.2.



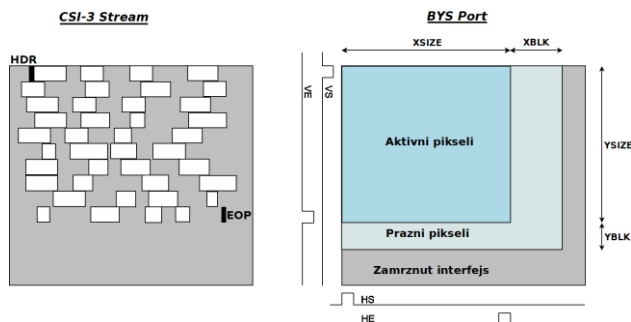
Slika 2.2 Blok šema *CSI-3* modula

Jedinica za raspakivanje piksela prihvata odbirak širine 64 bita, dok na svom izlazu generiše 4 piksela prema selektovanom formatu. Unutrašnja arhitektura jedinice prikazana je na slici 2.3. Oni bitovi odbirka koji ne mogu da se obrade u tekućem ciklusu, smeštaju se u registar i biće iskorišćeni u narednim ciklusima obrade.

U slučajevima kada je piksel takt sporiji od funkcionalnog ili kada se umeću prazni pikseli, *BYS* port blokira ceo *pipeline*. Prikaz tipičnog *CSI-3frame*-a na izlaznom portu prikazan je na slici 2.4.



Slika 2.3. Blok dijagram jedinice za raspakivanje piksela



Slika 2.4. Izlazni port za tipičan *CSI-3* tok podataka

Generator piksel takta je potpuno sinhronizovan sa funkcionalnim taktom. Željena frekvencija se dobija postavljanjem zahteva za aktivnu ivicu takta prema ćeliji koja obezbeđuje isti. Algoritam generisanja zahteva opisan je sa (2.1).

$$\begin{aligned} ACC &= PCLK \\ \text{if}(ACC + PCLK \geq 65536) \text{ Clk_en} &= 1 \text{ else Clk_en} = 0 \\ ACC &= (ACC + PCLK) \% 65536 \\ \text{goto } 2 \end{aligned} \quad (2.1)$$

3. IMPLEMENTACIJA DIZAJNA

3.1 Zahtevi dizajna

Ovo je tipičan *pipeline* dizajn sa 4 konkurentne faze koje su uočljive na slici 2.2. Svaki od blokova na glavnoj putanji podataka sastoji se od kontrolne jedinice (*FSM*) i entiteta za prenos i obradu podataka.

Izlazni pinovi modula moraju prolaziti kroz registre, kako bi se eliminisalo kombinaciono kašnjenje prema susednim periferijama. Podmoduli na *pipeline*-u poseduju opcione bafere koji mogu da smanje kritično kombinaciono kašnjenje u slučaju narušavanja vremenskih ograničenja sinhronog dizajna.

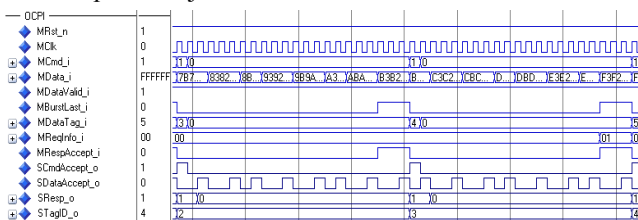
Unutar modula se nalaze instancirane reset i takt ćelije preko kojih se vrši distribucija istoimenih signala. Ove ćelije su posebno dizajnirane na nivou tranzistora i nalaze se u standardnoj biblioteci generičkih komponenti, a logička šema takt ćelije je prikazana na slici 3.3.

Signali test porta se ostavljaju nepovezani. Nakon sinteze, ove signale koristi softverski alat za uvezivanje flipflopova u test lance.

3.2 Realizacija pojedinih funkcionalnih blokova

Početak *pipeline*-a predstavlja izlaz *OCPI* bloka. Sa susednim *CSI-3LL* blokom komunikacija je ostvarena jednostavnim *handshaking* protokolom. Centralni deo

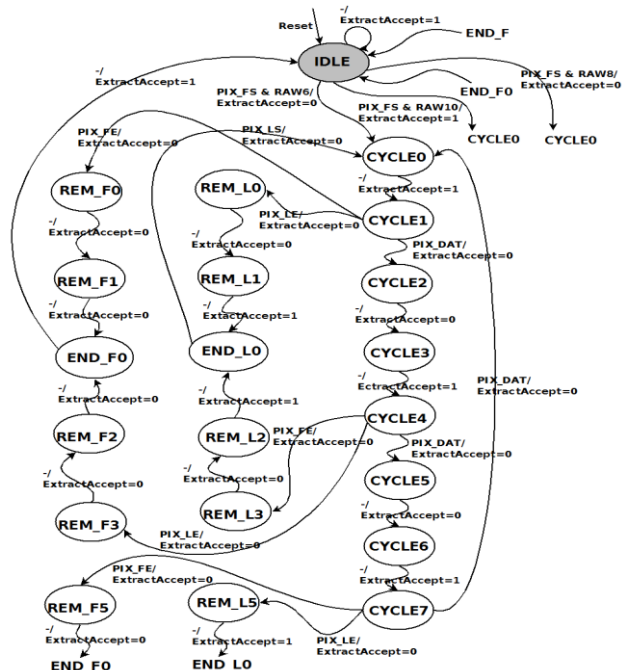
ulaznog porta je mašina stanja, koja postavlja sve kontrolne signale. Primer *back-to-back* transfera za selektovan *raw6* format prikazan je na slici 3.1.



Slika 3.1 Signali implementiranog ulaznog porta pri selektovanom *raw6* formatu

Drugi blok na *pipeline*-u je *CSI-3LL*, čija mašina stanja na osnovu zaglavlja paketa i *MreqInfo* signala sa ulaznog porta postavlja kavalifikator za svaki podatak. Unutar ove jedinice implementiran je bafer za skraćivanje kritične kombinacione putanje koja počinje na izlaznom portu.

Naredni podmodul na glavnoj putanji je zadužen za raspakivanje piksela. Mašina stanja je glavni kontrolni entitet koji na putanji podataka postavlja selektovane formate. Bit-mapa podržanih formata opisana je u [2], a deo implementirane mašine stanja zadužen za *raw6* raspakivanje prikazan je na slici 3.2. Može se uočiti da se detekcijom početka linije ne prelazi u početno stanje, čime se smanjuje logika ali se produžava moguće vreme oporavka u slučaju pogrešnog paketa.



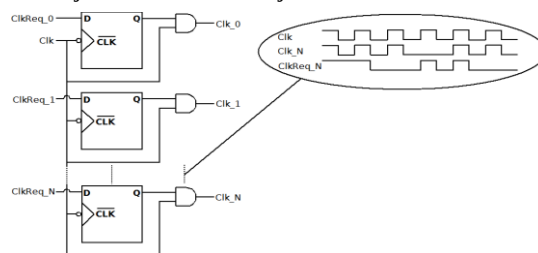
Slika 3.2 Mašina stanja zadužena za raspakivanje *raw6* formata

Konačna destinacija piksela je video port, koji kao i svi prethodni blokovi sadrži *FSM*. Ona je senzitivna samo na kvalifikatore piksela i u zavisnosti od njih postavlja kontrolne signale na izlazu. Dodatno, podmodul poseduje implementirane brojače za aktivne i prazne piksele u slučaju konfigurisanih horizontalnih i vertikalnih praznina. Svi brojači menjaju vrednost na piksel takt koji obezbeđuje interni takt generator.

Jedinica koja sadrži sve registre je konfiguracioni port. Ona ima posebnu vezu sa *CSI-3LL* blokom pošto su u njemu implementirane kopije registara. Sa detekcijom početka

frame-a vrši se kopiranje tekuće konfiguracije koja ne može da se menja do početka narednog *frame*-a (*shadowing*).

Podsistem za detekciju prekida vrši observaciju *pipeline*-a na funkcionalnom taktu. Kako se statusna polja unutar registara postavljaju na konfiguracioni takt, ovde je izvršena njihova sinhronizacija.



Slika 3.3 Čelija za distribuciju takt signala – Na opadajuću ivicu se lećuje zahtev za taktom

3.3 Sinteza dizajna

Nakon implementiranih funkcionalnosti, prelazi se na prevođenje *HDL* modela u mrežu generičkih komponenti. Sinteza za ovaj modul je rađena u *FPGA* tehnologiji, a glavni cilj je provera vremenskih ograničenja, odnosno pronalaženje najdužih kombinacionih putanja. Kritična kombinaciona putanja za buduću nadogradnju počinje u *CSI-3LL* bloku gde se nalazi kopija *BYS* registra, a preko video porta i signala za blokiranje *pipeline*-a završava u *CSI-3LL* mašini stanja. Svi rezultati sinteze prikazani su u tabeli 3.1.

Tabela 3.1 Rezultati sinteze *CSI-3 IP* modula

Alat	Xilinx ISE Design Suite 11.1, L.33
Familija	Virtex5
Mašina	XC5VLX110
Pakovanje	FF676
Optimizacija	Brzina
IOs	314
IOs Buffers	254
BELS	1223
FlipFlops	709

4. FUNKCIONALNA VERIFIKACIJA DIZAJNA

4.1 Strategija verifikacije

Verifikacija *CSI-3 IP* modula je realizovana na nivou dizajna, čiji je cilj provera skupa funkcionalnosti koje su neophodne pre isporuke dizajna na hijerarhijski viši nivo. Za proveru dizajna neophodno je verifikaciono okruženje i testovi. Okruženje se sastoji od aktivnih i pasivnih verifikacionih komponenti. Aktivne komponente su zadužene sa pobudu na nivou signala, a pasivne za prikupljanje odziva i njegovu validaciju. Test je najviši hijerarhijski modul koji generiše željeni scenario koristeći aktivne komponente unutar okruženja.

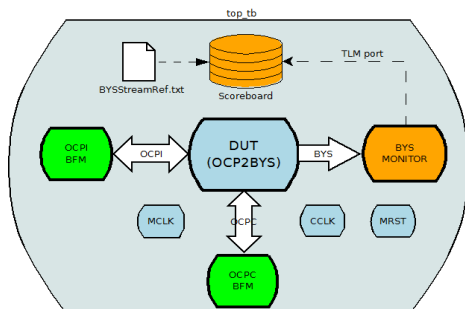
Digitalni dizajn (*DUT*) za verifikaciono okruženje predstavlja „crnu kutiju“, odnosno dostupni su samo njegovi portovi, ali ne i interni signali. Na ovom nivou provere, aktivne komponente poseduju minimalnu randomizaciju, a test scenariji su dirigovani. U tom kontekstu, testovi koriste „zlatne vektore“ za pobuđivanje dizajna, pri čemu svaki test proverava unapred definisan skup funkcionalnosti. Za generisanje „zlatnih vektora“ potrebno je razviti referentni model nezavisno od okruženja. Nakon pozitivnih statusa svih testova, neophodna je provera

pokrivenosti *HDL* koda pri čemu ova analiza nije glavni cilj u ovoj fazi.

Koraci koji uključuju: simuliranje scenarija, debugovanje dizajna, analizu koda sa performansama, rađeni su u softverskom alatu *ModelSim 6.1*.

4.2 Verifikaciono okruženje

Sastoji se od dve aktivne i dve pasivne komponente koje se mogu uočiti na slici 4.1. Unutar okruženja instancirani su moduli za generisanje konfiguracionog i funkcionalnog takta. Za komunikaciju pasivnih komponenti realizovan je pozadinski proces tkz. *TLM* port. Dodatno, okruženje nudi opcije slanja poruka i opciono zaustavljanje simulatora pri debugovanju test scenarija.



Slika 4.1 Verifikaciono okruženje za CSI-3IP modul

Ulazni port *CSI-3* modula povezan je sa *OCPI* aktivnom komponentom koja oponašasusednu periferiju sa hijerarhijski višeg nivoa. Ona sadrži skup procesa koji su dostupni test modulu za prosleđivanje željenih podataka prema *DUT*-u. Procesi unose minimalnu randomizaciju u smislu kašnjenja signala u različitim fazama *OCP* protokola.

Na konfiguracioni port se povezuje *OCPC* komponenta koja sadrži procese upisa i čitanja. Oni se koriste od strane test modula za promenu konfiguracije ili monitorisanje statusa dizajna.

Observaciju video porta vrši *BYS* monitor. Ova pasivna komponenta prikuplja transfere koji se sastoje od piksela i njihovih kvalifikatora. Kolektovan odziv se preko *TLM* porta dostavlja skorbordu.

Skorbord kao pasivna komponenta vrši poređenje podataka pristiglih preko *TLM* porta sa očekivanimreferentnim podacima. U slučaju ne podudaranja istih, akumulira se greška i menja status test scenarija.Referentni podaci su sastavni deo test vektora.

Tabela 4.1 Pokrivenost koda ključnih instanci *CSI-3 IP*

Instance OCP2BYSdizajna	Statement coverage	Condition coverage	Toggle coverage
ocpi_fsm_map	75.2%	66.5%	96.9%
csi3ll_fsm_map	90.5%	81.3%	89.8%
pixel_extract_fsm_map	76%	84.3%	84.9%
extract_data_path_map	100%	-	99.7%
bys_fsm_map	100%	98.9%	87.8%
ocpc_fsm_map	98%	91.1%	85.2%

4.3 Test scenario

Test scenario kao najviši hijerarhijski modul enkapsulira verifikaciono oruženje. On je zadužen za generisanje pobude na nivou transfera i koristi procese aktivnih

komponenti za njihovu distribuciju prema dizajnu. Pored pomenutog okruženja, test modul poseduje pomoćne procese koji učestvuju u kreiranju scenarija. Na samom početku simulacije aktiviraju se konkurentni procesi *tc_body* i *tc_end*. Prvi proces je zadužen za kreiranje scenarija sa glavnim zadacima raspakivanja „zlatnih vektora“ i prosleđivanju transfera prema aktivnim komponentama. Sa druge strane, *tc_end* je zadužen za provere po završetku testa i donošenje odluke o njegovom statusu pre završetka simulacije.

Glavni cilj verifikacije je pozitivan status svih test scenarija koji su prikazani u tabeli 4.2. Pokrivenosti *HDL* koda izmerena nakon pozitivnih statusa testova prikazana je u tabeli 4.1. Trenutna pokrivenost pokazuje potrebu za ekstenzivnijom verifikacijom na nivou modula, koja bi trebalo da pokrije veći broj scenarija od navedene liste u tabeli 4.2. U slučaju integracije *IP* dizajna unutar sistema, neophodno je ponoviti razvijene scenarije i na *SoC* nivou.

Tabela 4.2. Zbirka test scenarija i njihov status

Test scenario	Status
ocp2bys_dv_register	PASS
ocp2bys_dv_pixel_bypass	PASS
ocp2bys_dv_extract_bypass	PASS
ocp2bys_dv_extract_raw6	PASS
ocp2bys_dv_extract_raw8	PASS
ocp2bys_dv_extract_raw10	PASS
ocp2bys_dv_extract_bypass2raw8	PASS
ocp2bys_dv_bys	PASS
ocp2bys_dv_irq	PASS
ocp2bys_dv_back2back	PASS

5. ZAKLJUČAK

Funkcionalna specifikacija *CSI-3 IP* modula je bazirana na istoimenom standardu za komunikaciju senzora kamere i aplikacije. Digitalni dizajn namenjen je implementaciji u standardnoj *ASICGS80* tehnologiji, ali osnovne analize i sinteza su rađeni za *FPGA* korišćenjem *Xilinx* alata. Verifikacija dizajna realizovana je direktnim scenarijima uz primenu „zlatnih vektora“.

Trenutna implementacija dizajna poseduje najbolje performanse samo kada je piksel takt video porta konfigurisan na vrednost funkcionalnog takta. U budućim nadogradnjama, za podizanje performansi pri nižim vrednostima piksel takta neophodna je implementacija bafera sa programabilnim pragom aktiviranja video porta.

6. LITERATURA

- [1] *OCP-IP Association*, "Open Core Protocol Specification 2.2", 2007.
- [2] *MIPI Alliance*, "Camera Serial Interface 3 (CSI-3)", New York, Novembar 2011.

Kratka biografija:



Dejan Križan rođen je u Subotici 1987. god. Fakultet tehničkih nauka upisao je 2006. god. Bečelor rad iz oblasti Elektrotehnike i računarstva – Primenjena mikroračunarska elektronika odbranio je 2010.god.

МОДУЛАРНА WEB АПЛИКАЦИЈА НА .NET ПЛАТФОРМИ**MODULAR WEB APPLICATION ON .NET PLATFORM**

Душица Алексић, Факултет техничких наука, Нови Сад

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – У овом раду описана је модуларна web апликација креирана помоћу ASP.NET MVC радног оквира. Представљене су карактеристике и архитектура таквог решења. Предложена су нека решења за унапређење такве апликације са становишта модуларности.

Abstract – This paper describes a modular web application created using the ASP.NET MVC framework. The characteristics and architecture of this solution are presented. The proposed some solutions to improve such applications from the standpoint of modularity.

Кључне речи: web, ASP.NET MVC, MEF, plugin

1. УВОД

Захтеви клијената и корисника за различитим функционалностима једне апликације, који се дефинишу пре почетка имплементације, али исто тако и захтеви за додатним функционалностима, који долазе након што се апликација испоручи клијенту, између осталог, довели су до тога да се размишља како да се на што бржи и адекватнији начин одговори на исте. Исто тако, ту су и захтеви других корисника који на основу својих потреба захтевају мање или веће измене над већ постојећим функционалностима апликације. У времену када је велика конкурентност на тржишту, задовољство клијената представља један од најзначајнијих фактора који диктира успешност рада неке компаније или произвођача. Тада брзина одговора на поменуте честе промене у захтевима клијената, као и сам квалитет испорученог софтвера, јесте нешто о чему треба водити рачуна.

Јавља се потреба за апликацијом коју је могуће лако надоградити, додати нову функционалност или пак изменити неку постојећу. При томе унос тих измена треба да што мање утиче на остатак апликације током имплементације истих. Модуларна апликација је једно од решења које задовољава наведене потребе. Основна идеја модуларности јесте могућност додавања нових функционалности без превеликог утицаја на остатак апликације. Оно чиме се то постиже је подела софтвера на мање целине, чиме се обезбеђује лакше савладавање комплексности система. Модул представља логички независан део програма. При томе се истиче потреба за кохезијом, са једне стране, и слабој повезаности модула, са друге стране [1].

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је био др Александар Ердџан, ред. проф.

Кохезија представља степен повезаности елемената унутар једног модула. Са становишта модуларности потребно је обезбедити висок степен кохезије. То значи да постоји јака веза између елемената унутар компоненте. *Coupling* представља степен међузависности више модула. Показује колико једна компонента има свест о другој компоненти, о њеним елементима. Оно што је потребно да се обезбеди је што мања зависност између модула (*loose coupling*).

Проширивање софтвера *plugin*-има представља атрактивно решење за развој модуларних и лако проширивих апликација. *Plugin* архитектура софтвера раздваја апликацију на језгро и на целине које то језгро надограђују функционалностима. *Plugin*-и комуницирају са околином преко дефинисаних интерфејса. Апликација треба да обезбеди *plugin manager*, чији задатак је да учита *plugin*-е у апликацију. Једна од предности овакве архитектуре је та што се функционалности имплементирају као одвојене целине, па *plugin*-и могу бити развијани паралелно од стране више тимова.

2. МОДУЛАРНОСТ WEB РЕШЕЊА

Све погодности које нуди софтвер са *plugin*-инима су применљиве и на web апликације. Коришћењем Microsoft платформе за развој web апликације, стављају се на располагање радни оквири као што су: APS.NET Web Forms, ASP.NET Single Page Application, и ASP.NET Web Pages. Онај који пружа најбољу подршку у креирању модуларне web апликације је ASP.NET MVC радни оквир, јер имплементира MVC шаблон. Овај шаблон састоји се од три компоненте:

- *Model* – рукује подацима;
- *View* – одговорна за приказ корисничког интерфејса, а податке добија од *Model*-а;
- *Controller* – рукује улазним подацима, комуницира са *Model* и *View* компонентом.

Коришћењем ASP.NET MVC радног оквира за развој web апликације подржава се раздвајање на слојевиту архитектуру.

У општем случају, оно што *plugin* треба да садржи у оквиру ASP.NET MVC web апликације јесте:

- контролер који треба да мапира своје руте,
- сервис који садржи бизнис логику, и
- web странице.

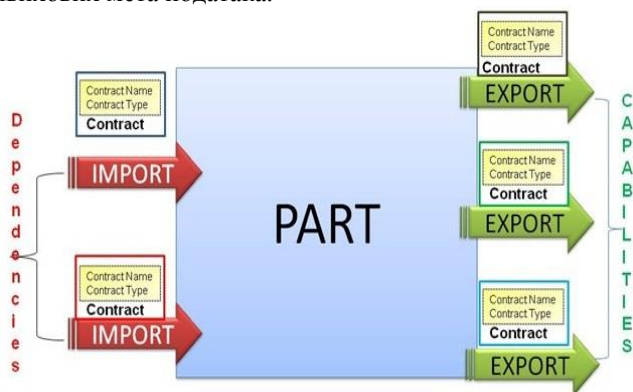
На жалост, *plugin* архитектура има и своје мане. Брзина учитавања и доступност компоненти у апликацији може се одразити на одзив апликације. Код оваквог типа архитектуре поставља се питање на који начин учитати компоненте у апликацију. Најчешће се доводи до тога да апликација није у

могућности да сама открије компоненте које треба да укључи. Ту је такође и питање поновне искористивости компоненте у другим апликацијама.

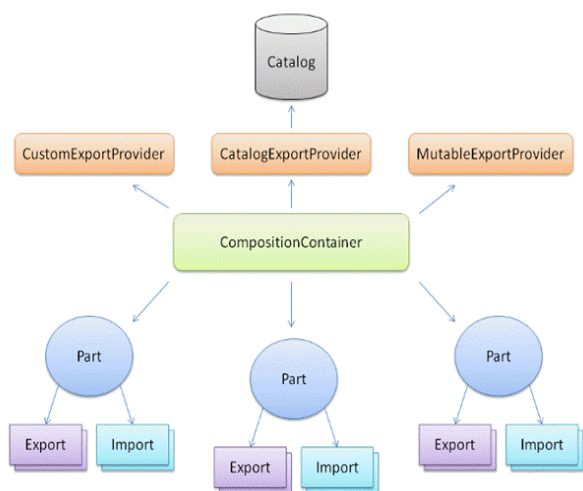
2.1. РЕШЕЊА ЗА .NET ПЛАТФОРМУ

Постоји пар решења за учитавање компоненти у апликацију, креираних за .NET платформу, а овде ће бити поменути MEF и Orchard.

MEF представља решење за креирање проширивих .NET апликација, тј. апликација које се лако прилагођавају новим захтевима клијената. У оквиру апликације, MEF компоненте се откривају у *runtime*-у, што значи да апликација користи компоненте без међусобне зависности путем референци и конфигурационих фајлова. Насупрот томе, MEF омогућава проналажење компоненти на основу њихових мета података.



Слика 1 - Концепт MEF-a



Слика 2 - Архитектура MEF-a

Посматрајући детаљније, MEF садржи композитне делове (*ComposableParts*), где сваки композитни део пружа један или више експорта (*Export*) и може да зависи од једног или више других композитних делова (*Import*).

Ту је такође и уговор (*Contract*) који представља везу између експорта и импорта. Састоји се из два дела: назива и типа уговора (слика 1), и на основу њих се врши повезивање делова. Како би се ти делови стварно повезали, MEF користи контејнер који врши повезивање (*CompositionContainer*).

Тај контејнер проналази композитне делове помоћу каталога (*Catalog*), чија је пак намена да омогући

откривање композитних делова из неког директоријума или DLL фајла (слика 2) [2].

MEF на сваки елемент проширења гледа као на посебни композитни елемент, чиме је омогућен висок ниво флексибилности.

Други начин за учитавање компоненти у апликацију представља Orchard као *open source* CMS радни оквир. Листа доступних модула у апликацији се добија претраживањем *Modules* и *Core* директоријума у циљу проналажења *module.txt* фајлова. [3] У основи, врши се претраживање директоријума у циљу проналажења DLL фајла са именом које одговара имену модула специфицираном у *module.txt* фајлу.

Иако се Orchard апликација у основи базира на механизме које нуди ASP.NET MVC радни оквир, да би апликација могла да се прошири на овакав начин, мора бити креирана коришћењем Orchard радног оквира.

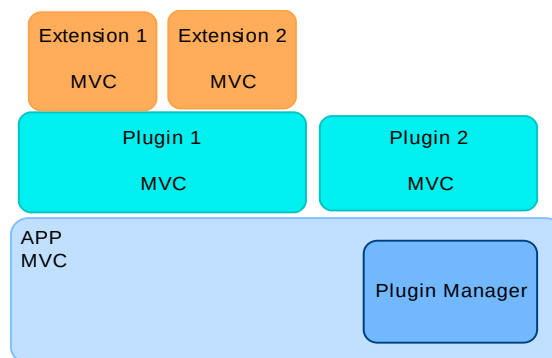
3. ОПИС РЕШЕЊА

Предложено решење за модуларну веб апликацију се заснива на савременом приступу где се *plugin* проширује другим *plugin*-има.

Језгро апликације садржи глобалне сервисе и моделе који се користе кроз целу апликацију. Такође, ту је и *plugin manager* који се бави учитавањем и манипулисањем *plugin*-а. Таква манипулација подразумева, између осталог, и проверу доступности сваког *plugin*-а за одређеног корисника.

Plugin проширује језгро апликације новим функционалностима тако што дефинише своје сервисе и контролере који се потом укључују у апликацију. Могуће је изменити понашање *plugin*-а имплементацијом проширења и тада се проширења понашају као *plugin*-и.

Апликација, као и сваки *plugin* и проширење организовани су по MVC структури (слика 3).



Слика 3 - Модуларност апликације

Архитектура *plugin*-а приказана је на слици 4. Она је подељена на леви део који се извршава на клијентској страни и серверски део приказан на десној страни.

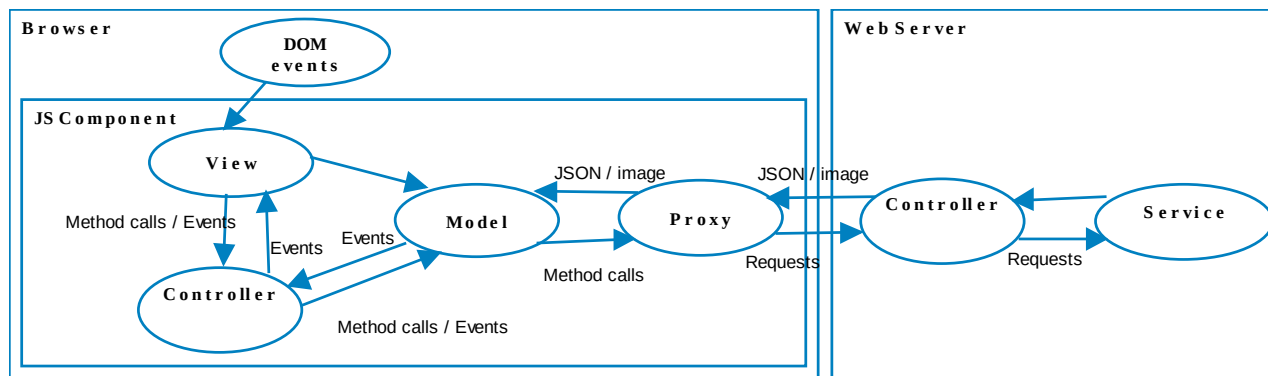
Серверски део *plugin*-а састоји се из:

- *Controllers* – обрађују долазне захтеве, враћајући JSON поруке;
- *Models* – репрезентује модел података;
- *Services* – садрже бизнис логику.

Клијентски део *plugin*-а састоји се из:

- *Controllers* - обрађују долазне захтеве од view-a, враћајући му податке добијене из модела;
- *Models* – комуницира са *proxy*-ем и на основу одговора модификује модел података;
- *Views* - ослушкује DOM догађаје настале услед интеракције корисника, комуницира са *controller*-ом;

- *Proxy* – шаље захтеве ка *Web serveru*, у зависности од одговора враћа моделу JSON податке.



Слика 4 - Архитектура *plugin-a*

Пример апликације рађен је у Microsoft ASP.NET 4.5 верзији радног оквира. Коришћено је проширење ASP.NET MVC4 како би се омогућило раздвајање на *model*, *view* и *controller* компоненте и тиме је стављен јасан нагласак на разумљиву архитектуру. У имплементацији коришћен је JavaScript, док се комуникација са сервисима обавља преко AJAX позива. Коришћена је и jQuery JavaScript библиотека за лакше манипулисање HTML елементима.

3.1. Начин учитавања *plugin-a*

При учитавању *plugin-a* потребно је додати његове информације на серверској страни. Исто тако, потребно је доставити и клијентске фајлове *plugin-a* на клијента.

Како би се учитали *plugin*-и у апликацију на серверској страни коришћен је MEF.

Пошто *plugin* дефинише своје контролере и сервисе, потребно је сваког од њих декорисати *Export* атрибутом. Поред тога, *plugin* контролер мора да мапира своје руте. Због тога имплементира једноставан интерфејс. Унутар њега дефинисана је метода која као повратну вредност враћа колекцију рута, које се односе на *plugin*, а које ће бити интегрисане у систем рутирања апликације.

Сервис са собом носи и информације о називу *plugin-a*. Касније она служи за проверу да ли ће кориснику у апликацији бити доступан садржај који обезбеђује *plugin*.

Пошто су се додали *plugin*-и на серверској страни, након учитавања апликације добављају се информације о *plugin*-има на клијентској страни. Учитавање клијентских фајлова *plugin-a* врши се динамички у редоследу, тј. након што се добије информација о успешности учитавања једног, учитава се следећи *plugin* и сви учитани *plugin*-и се смештају у DOM стабло *browser-a*. Такође, након што се *plugin* учита, врши се додавање *url*-ова темплејт фајлова које садржи *plugin*. То додавање се врши у *appConfig* фајл апликације, где се иначе дефинишу *url*-ови за све темплејт фајлове.

3.2. Проширења *plugin-a*

Проширењима се надограђује или мења понашање *plugin-a*. По структури су исти као и *plugin*-и, као и по учитавању са серверске стране. Могу да дефинишу своје контролере и сервисе и са собом носе скуп html темплејт фајлова које дефинишу. При учитавању у апликацију, учитавају се након *plugin-a*. Тиме се обезбеђује да фајлови који буду учитани из проширења буду и коришћени да прошире *plugin* фајлове или уместо већ учитаних фајлова из *plugin-a*.

Са клијентске стране постоји разлика при учитавању проширења, а разлика је у томе да проширења не дефинишу *url*-ове за своје html темплејт фајлове. У том случају апликација се ослања на механизам „гажења“ фајлова. Уколико проширење обезбеђује измену у корисничком интерфејсу у односу на *plugin* који проширује, тада се клонира html темплејт фајл из *plugin-a* у проширење. При томе се не мења назив фајла, већ се само унесу жељене измене. Када се апликација пакује, на локацији на којој се налазе, html темплејт фајл из проширења ће физички заменити фајл из *plugin-a*, тј. фајл из проширења је фајл са истим називом, али новијим временом креирања у односу на фајл из *plugin-a*. С обзиром да је *plugin* дефинисао *url* html темплејта према називу фајла, када се апликација покрене биће коришћен темплејт из проширења.

4. УНАПРЕЂЕНО РЕШЕЊЕ

У решењу које је представљено уочен је простор за унапређење појединих аспеката апликације. Та унапређења се односе на решење учитавања *plugin-a* са клијентске стране и обједињавање html темплејт фајлова у један.

4.1. Учитавање *plugin-a* са клијентске стране

Приликом интеракције корисника са апликацијом, уочени су недостаци по питању перформанси и одзива апликације. Након што се корисник улогује у апликацију, потребно је да прође одређени временски

интервал како би се учitale све потребне компоненте. То се дешава из разлога што фајлови *plugin*-а са клијентске стране нису довољно брзо учитани. Динамички начин учитавања *plugin*-а, представљен у иницијалном решењу, је прихватљиво решење за *plugin*-е мање величине. Међутим, када су фајлови *plugin*-а већи, нпр. из разлога комплексних функционалности које обезбеђују, повећава се и време потребно за њихово учитавање у апликацију.

Учитавање фајлова у индексну страницу апликације, приликом њеног креирања, обезбедило је доступност *plugin*-а у тренутку када је апликација учитана. Пре него што се фајлови проследи индексној страници, у *plugin manager*-у се за сваки *plugin* провери да ли корисник има дозволу за дати *plugin*. Тиме је обезбеђено да је кориснику након логовања доступан само онај садржај који обезбеђује *plugin* за који има приступ.

Предност оваквог начина учитавања види се и на укупном времену за које се апликација учита. У случају иницијалног решења начина учитавања *plugin*-а било је потребно око 3,92 секунде да се прочитају све потребне компоненте. Након примењене измене, уочено је побољшање од око 45,7%. Где након 20 пута покретања апликације, средња вредност учитавања износи око 2,13 секунди.

4.2. Обједињавање html темплејт фајлова

Истражена је могућност обједињавања html темплејт фајлова и на које све аспекте апликације би такво решење утицало.

Оно што је свакако видљива предност обједињавања html темплејт фајлова јесте сама величина фајла. Пример обједињавања је урађен над темплејт фајловима из апликације која садржи 22 фајла, чија укупна величина износи 56,3 KiB. При обједињавању су уклоњени коментари, као и размаци између елемената у садржају фајла. Као резултат тога креиран је један фајл величине 45,8 KiB. Уштеда која је остварена је око 18,7% и она се директно одражава на време учитавања темплејта.

Такође предност обједињавања html темплејта је и са аспекта кеширања и слања захтева у *browser*-у. Након примене обједињавања, шаље се један захтев за довлачење и кешира се један фајл који је мање величине. Тиме се заузима мање меморије у кешу, а самим тим и време за довлачење и кеширање је мање. У решењу без обједињавања се за сваки фајл шаље посебан захтев и заузето је више меморије у кешу *browser*-а.

Када у *plugin*-у, након обједињавања, постоји један фајл који садржи све html темплејте које тај *plugin* користи, јавља се питање на који би начин проширења обезбедила измену. Више није могуће ослањати се на „гажење“ и клонирање појединачних фајлова *plugin*-а. У том случају у проширењу креира се само један фајл који садржи само оне делове који се мењају, без клонирања целих *plugin* html темплејт фајлова. Како се html темплејт фајл *plugin*-а састоји из више *script* тагова који имају свој *id*, само онај *script* таг у којем се жели направити промена наћи ће се у темплејту проширења.

Такво решење могуће је реализовати на више начина. Први начин је да у html темплејту проширења *script* таг има исти *id* као таг из *plugin*-а. У том случају, када се прочита *plugin*, *template manager* пролази кроз html темплејт *plugin*-а и памти вредности *id*-ева, који се налазе у њему и назив фајла из ког се чита. У овом случају то је *plugin.htm*. Након тога, када се прочита проширење, *manager* за сваки *id* у њему проверава да ли се налази у листи *plugin id*-ева коју је претходно направио. Уколико пронађе *id* са истом вредности, за тај *id* измениће назив фајла из ког се чита у *проширење.htm*. На тај начин апликација ће знати у ком фајлу да тражи потребни *id*.

Други начин је да *id script* тага у проширењу има јединствену вредност. Тада је поред њега потребно да има и атрибут *override*, чија ће вредност бити иста као вредност *id plugin* тага који се мења. У том случају *manager* ће у листу ставити све *id*-еве из *plugin*-а, а затим и све из проширења. Тада ће се за жељени *id* гледати у листи да ли постоји *override* са истом вредношћу. Уколико постоји, тада ће се користити *id* који иде са тим *override* атрибутом. Биће приказан садржај тага из проширења уместо садржаја из *plugin*-а.

5. ЗАКЉУЧАК

У овом раду представљена је *web* модулarna апликација, креирана помоћу ASP.NET MVC радног оквира. Описани су њена архитектура и основне карактеристике. Представљени су неки од начина учитавања компоненти у апликацију креираних за .NET платформу. Додатно, предложена су решења за унапређење већих апликација, а тичу се клијентског дела учитавања компоненти у апликацију. Анализиране су погодности тих решења, као и резултати добијени након њихове имплементације у апликацији.

6. ЛИТЕРАТУРА

- [1] ISO/IEC/ IEEE 24765, 2010, Systems and software engineering – Vocabulary
- [2] Mark Seemann, Dependency Injection in .NET, Manning, 2011
- [3]

<http://docs.orchardproject.net/en/latest/Documentation/Orchard-module-loader-and-dynamic-compilation/> (приступљено септембар 2016.)

Кратка биографија:



Душица Алексић рођена је у Лозници 1991. год. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Примењене рачунарске науке и информатика одбранила је 2016. год.

**MODELOVANJE DISTRIBUTIVNIH ELEKTROENERGETSKIH SISTEMA
UPOTREBOM MICROSOFT GRAPH ENGINE KOMPONENTE****MODELING OF DISTRIBUTIVE ELECTRIC POWER SYSTEM
USING MICROSOFT GRAPH ENGINE COMPONENT**

Nedeljko Stojaković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Rad prikazuje performanse dobijene modelovanjem jednog distributivnog elektroenergetskog sistema upotrebom Microsoft Graph Engine komponente. Model koji se koristio u radu opisan je CIM (Common Information Model) standardom za modelovanje elektroenergetskih sistema. Na takvom modelu je primjenjen generički pristup podacima, opisan DAF/GDA (Data Access Facility/Generic Data Access) standardom. Dobijeni model elektroenergetskog sistema je smješten na serveru, pri čemu je unutrašnja organizacija servera skrivena od korisnika iz želje da mu se pristup omogući preko GDA interfejsa.

Abstract – This paper presents the performance obtained by modeling a distributive power system using Microsoft Graph Engine component. The model that was used in this paper is described with CIM (Common Information Model) standard for modeling electrical power system. For such a model is applied generic data access, described with DAF/GDA (Data Access Facility/Generic Data Access) standard. The resulting model of the power system is located on the server, where server internal organization is hidden for client, from desire to facilitate access to him through GDA interface.

Cljučne reči: Microsoft Graph Engine (Trinity), TSL, Neo4j, CIM, DAF/GDA, .NET, C/C++, C#.

1. UVOD

Zbog velike potrebe i značaja upotrebe elektroenergetskog sistema (Power Energy System), kao ozbiljan i složen zadatak ističe se upravljanje elektroenergetskim sistemima. Primjena novih tehnologija je omogućila povezivanje velikog broja računara u složen distribuiran sistem koji se danas sve više koristi. Distribuiran sistem treba da sakrije od korisnika razlike među računarima u tom sistemu i način komunikacije.

Radi lakšeg razvijanja softvera za modelovanje nekog elektroenergetskog sistema često se koristi objedinjeni jezik za modelovanje ili UML (Unified Modeling Language) [1]. Sa takvim modelom pristupamo implementaciji sistema sa svim funkcionalnostima koji su nam od značaja. Iz potreba razmjene informacija i integracije sistema, vodeći stručnjaci i kompanije su razvili i definisali standarde i preporuke.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Darko Čapko, docent.

Konkretno, za izradu modela elektroenergetskih sistema definisani su CIM (Common Information Model) [2] i standard za generički pristup podacima iz domena elektroenergetskih sistema GDA (Generic Data Access) [3]. Širenjem takvih sistema i određenih ograničenja u pogledu hardvera, razvijena su rešenja u vidu distribuiranih orijentisanih graf baza podataka. Jedno od tih rešenja je i Microsoft rešenje pod nazivom Graph Engine (ranije poznat kao Trinity).

**2. OBJEKTO-ORIJENTISANE GRAF BAZE
PODATAKA**

Graf je apstraktni matematički objekat koji je, neformalno govoreći, sastavljen od tačaka, odnosno čvorova, i linija, odnosno grana, koje spajaju po dva čvora. Veoma je česta upotreba grafova za opis modela i struktura podataka. Proučavanje algoritama koji rješavaju probleme upotrebom grafova predstavlja veoma značajan dio informatičke nauke [4]. Zbog široke upotrebe grafova u računarskom okruženju, suočavamo se sa izazovima, na svim nivoima, koje nam donosi modelovanje podataka kroz graf.

Jedno od ponuđenih rešenja objekto-orijentisane graf baze podataka jeste Graph Engine komponenta (u nastavku GE), razvijena od strane Microsoft kompanije. Dizajn GE komponente zasnovan je na uvjerenju da mreže sa velikom brzinom postaju pristupačnije kao da cijene DRAM opadaju i sva in-memory rešenja obezbjeđuju najniži trošak za veliki spektar aplikacija. Pored GE, na tržištu postoji još sličnih rešenja, kao što je Neo4j [5] i HyperGraphDB [6] koji su fokusirani na podršci online procesiranja transakcija na graf podacima. Razlika u odnosu na GE je u tome što oni nisu distribuirani, tj. ne podržavaju grafove koji su podjeljeni na više računara. To predstavlja ograničenje za grafove kojima mogu da rukuju na odgovarajući način.

Sa druge strane tu su MapReduce [7], Pregel [8], GraphLab [9]. Za razliku od Neo4j-a i HyperGraphDB-a, ne podržavaju online procesuiranje upita, ali su optimizovane za analizu velike količine podataka podjeljenih na hiljade računara. Međusobno, sve ove baze pokrivaju nedostatke drugih i imaju neke svoje prednosti. GE nije sistem koji dolazi sa ugrađenim alatima za obradu grafova.

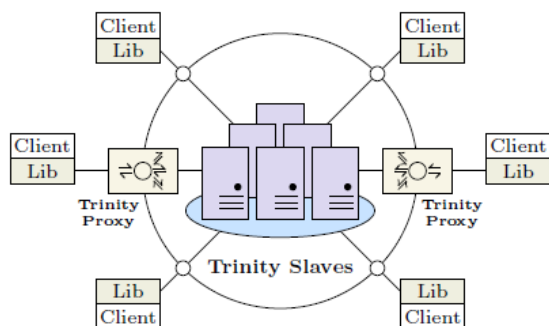
Sa svojim fleksibilnim podacima i mogućnostima modelovanja proračuna, GE daje mogućnost razvoja različitih modela za različite varijacije graf aplikacija.

Drugim riječima, lako se pretvara u sistem koji podržava konkretnu graf aplikaciju.

3. MICROSOFT GRAPH ENGINE

Uz optimalno upravljanje memorijom i mrežnom komunikacijom, GE podržava brzu pretragu grafa, i to radi efikasno poput paralelnog programiranja. Posebno, GE usklađuje pristup u *online* i *offline* proračunima da bi što bolje optimizovali performanse memorije i komunikaciju. Time se omogućava efikasno procesiranje *online* upita, kao i *offline* analiza velikih grafova. GE je infrastruktura za skladištenje i proračune smještena na vrh klastera dobro povezanih računara.

Kao infrastruktura skladišta, GE organizuje memorije više računara u globalni adresibilni, distribuiran memorijski adresni prostor (*Memory Cloud*) za podršku velikih grafova. Pored toga, podržava objektno-orijentisano rukovanje podacima. Arhitektura GE klastera (slika 1) sadrži više komponenti koje komuniciraju kroz mrežu.



Slika 1. Trinity/Graph Engine klaster [10]

Prema ulogama, komponente se mogu podijeliti u tri grupe: server (*server*), klijent (*client*) i proksi (*proxy*). Uloga servera je dvostruka, da skladišti podatke i obavlja proračune nad podacima. Proračuni obično uključuju slanje i primanje poruka od drugih GE komponenti. GE proksi rukuje jedino porukama i ne posjeduje ništa od podataka. Obično služi kao međusloj između servera i klijenta. Proksi nije obavezan. GE klijent je korisnički sloj između GE sistema i krajnjih korisnika. Komunicira sa serverima i proksijima kroz API (*Application Programming Interface*) [10].

3.1. Trinity Specification Language

Trinity Specification Language (TSL) je deklarativni jezik koji je dizajniran da obezbjedi fleksibilnost sistema [10]. Predstavlja most između logičke strukture podataka i načina na koji se predstavljaju u *cloud*-u, a to su tako zvane ćelije. GE dozvoljava korisnicima da pomoću TSL-a definišu graf obrazac, komunikacione protokole i računске paradigme.

TSL spaja sledeće tri stvari zajedno: modelovanje podataka, modelovanje protokola i specifikaciju formata razmjene podataka. TSL dopušta korisnicima razmjenu podataka bez serijalizacije i deserijalizacije na efikasan način šifrovanja strukture podataka. GE kompajlira TSL skriptu u .NET biblioteku. Generisana biblioteka je proširenje modula koji sadrži skup proširenih metoda GE sistema. Korisnik definiše svoju strukturu koja se sastoji

od tri tipa polja: ugrađeni tipovi podataka, drugih struktura i tipova kontejnera. [10] Sama sintaksa TSL-a je slična definisanju struktura u C/C++ ili C#. Tip ćelije definiše korisnik koja se takođe sastoji iz ista tri dijela kao struktura.

Ključ je 64-bitni globalni identifikator koji je jedinstven. Obrazac dijela vrijednost iz ključ-vrijednost para može biti određeno sa TSL ćelijom. GE nakon kompajliranja TSL skripte generiše ključ-vrijednost interfejs (*SaveCell*, *LoadCell*, *itd.*) i metode za pristup podacima za svaki tip ćelije (*accessor*).

Svaka ćelija dobija svoj *accessor*. Podaci u ćeliji su smješteni u obliku niza bitova. Protokoli određuju ugovor između strana koje razmjenjuju poruke i određeni su sa tri stvari: tip protokola, poruka sa zahtjevom (*request*) i odgovor na poruku (*response*). Mogući tipovi protokola su sinhroni (*Syn*), asinhroni (*Asyn*) i HTTP. Za svaki tip definisana su pravila za zahtjev i odgovor [10].

3.2. Pristup ćelijama i podacima

Kao što ranije spomenuto, nakon kompajliranja TSL projekta, generišu se određene metode za rukovanje i manipulacijom ćelijama, pa samim tim i podacima unutar ćelija. *Accessor* radi kao pokazivač, ne može biti sačuvan da bi se kasnije upotrebio zato što memorijski blok na koji pokazuje *accessor* može biti prepisan operacijom drugog *accessor*-a. Objekat za pristupanje ćelijama (*Cell accessor*) je objekat za jednokratnu upotrebu.

U C# je to obezbjeđeno upotrebom *using* konstrukcije. U suprotnom može doći do curenja memorije. Još jedno ograničenje koje postoji kod *Cell accessor*-a je da se ne mogu koristiti na umetnute načine, jer mogu da prouzrokuju zastoje (*deadlock*). Svako operaciji koja vrši izmjenu sadržaja ćelije je zagarantovana atomičnost. Ugrađeni atomični operatori za manipulisanje ćelijama su *AddCell*, *SaveCell*, *LoadCell*, *RemoveCell*. GE obezbjeđuje više šablona za pristupe [10]. Prije upotrebe neke od metoda za pristup podacima, potrebno je naći kompromis između praktičnosti i performansi.

4. DIZAJN I REALIZACIJA SISTEMA

Svrha ovog modela je da predstavi efikasno rešenje za pretraživanje i analizu podataka sadržanih u modelu podataka elektrodistributivne mreže modelovanih pomoću *Microsoft Graph Engine* komponente. Pretraživanje podrazumjeva pronalaženje određenih informacija iz modela koje su zatražene sa klijentske strane. Informacije predstavljaju informacije o elementima i objektima unutar elektrodistributivnog sistema. Informacije o modelu, koje su bile smještene u bazu podataka kao GIS podaci, su upotrebijene za izgradnju modela elektrodistributivne mreže u obliku grafa.

Primjenom određenih operacija i postupaka kreiran je graf konektivnosti. Elektrodistributivni sistem je zasnovan na CIM [2] standardu koji je modelovan kroz koncept *Microsoft Graph Engine*.

Kako je u GE osnovna jedinica podataka ćelija, da bi realizovali model potrebno je definisati čvor grafa kao ćeliju na sledeći način:


```

cell struct TVertex
{
    short Table;
    List<TTag> Tags;

    List<CellId> Adjs;
}

struct TTag
{
    TTagType TType;
    string TagID;
    string TableName;
}

enum TTagType
{
    FEEDEROBJECT = 0,
    CIRCUIT = 11,
    LVNETWORK = 12,
    NETWORKTYPE = 1,
    WINDINGADJACENT = 2,
    ...
}

```

Radi pojednostavljenja, pomoću ćelije TVertex su predstavljeni svi elementi unutar elektrodistributivnog sistema (trafostanice, prekidači, čvorovi konektivnosti,...). Na taj način, za čvorove, imamo samo jedan tip ćelija TVertex. Informacije koje su nam od značaja za modelovanje su matrica povezanosti elemenata List<CellId> Adjs, lista tagova List<TTag> Tags čija je namjena u početku bila da se opiše pripadnost nekog čvora određenom kontejneru.

Tip liste tagova je struktura TTag koja nosi informacije o tipu taga (TType), jedinstvenom identifikatoru (TagID) i nazivu taga (TableName). Tip taga je enumeracija sa definisanim tipovima tagova. Dobijeni model elektroenergetskog sistema kreiran pomoću GE skladišten je na serveru, a unutrašnja organizacija podataka skrivena je od klijenta.

Razlog takve organizacije je ideja da se klijentu omogući pristup preko GDA interfejsa. Pored ovoga, kao ćelije su definisani i kontejneri unutar CIM modela na sledeći način:

```

cell struct Container
{
    TTagType TType;
    CircuitType CirType;
    string ContainerName;
    string ContainerID;
    List<long> ListVertex;
    List<string> ListContainer;
}

enum CircuitType
{
    HV,
    MV,
    LV,
    UNDEFINED
}

```

Za kreiranje kontejnera kao ćelije, iskorišćena je lista tagova koju svaki čvor nosi kao informaciju. Informacije koje imamo o kontejneru su tip kontejnera po tipu taga ili tipu *circuit*-a. *Circuit* nam daje informaciju o tome da li jedan čvor istovremeno pripada različitim naponskim nivoima (HV-High Voltage, MV-Middle Voltage, LV-Low Voltage). Takođe, informacija o imenu kontejnera, preuzeto od imena taga, identifikator preuzet od identifikatora taga, listu ListVertex koja predstavlja listu identifikacionih brojeva opreme (čvorova) koja se nalazi u tom kontejneru i listu ListContainer koja predstavlja listu identifikacionih brojeva kontejnera unutar tog kontejnera.

Pri korišćenju *Microsoft Graph Engine* komponente, svaka ćelija dobija jedinstven identifikacioni broj (*cellId*). Takođe, potrebna komunikacija između komponenti je definisana preko GE protokola.

Drugi dio rada se odnosio na implementaciju GDA interfejsa, pri čemu su implementirane samo *GetValues()* i *GetExtentValues()* metode.

Funkcionalnost ovih metoda je definisana DAF/GDA standardom [3].

5. TESTIRANJE I PRIKAZ REZULTATA

Prikaz testiranja i rezultata će biti posmatran u dva odvojena dijela. Prvi dio se odnosi na kreiranje grafa pomoću GE komponente, a drugi na implementaciju GDA interfejsa.

5.1 Testiranje Microsoft Graph Engine-a

Za testiranje se koristio graf koji ima približno 4 miliona čvorova koji su realizovani u *Microsoft Graph Engine* ćelije. Taj graf inicijalno, nakon izgradnje, zauzima približno 5 GB memorije. Test prikazuje serijalizaciju i deserijalizaciju grafa iz fajl sistema u memoriju. Inicijalno trajanje serijalizacije je definisano na 30 sekundi. U tabeli 1 su prikazani rezultati serijalizacije na *LocalStorage*-u.

Tabela 1. Testiranje serijalizacije

	Broj ćelija	Ukupno ćelije [MB]	Ukupno rez.mem. [MB]
1.	3 867 318	618,35	1670,18
2.	482 401	79,92	1093,58
3.	240 001	39,20	915,52

Iz prethodne tabele možemo vidjeti da za relativno veliki broj čvorova/ćelija, ukupno zauzeće memorije nije ogromno. Za primer 1 iz tabele 1, ukupna rezervisana memorija je približno 1.6 GB koja je i podrška za brzo dodjeljivanje novih podataka, od čega se samo za ćelije iskoristi oko 600 MB. Za primer 2 iz tabele 1, trajanje deserijalizacije je trajalo 1.77 sekundi.

Takođe vidimo da za oko 15 puta manji broj ćelija, ukupna memorija koja se zauzme je skoro duplo manja (primer 1 i 3, tabela 1), ali je memorija koja se iskoristi samo za ćelije oko 15 puta manja. Od početnog grafa izdvojen je potpuno uvezan graf koji sadrži približno 500 000 ćelija za testiranje algoritma za najkraći put.

Ukupno vrijeme izvršavanja algoritma je iznosilo 1.11 sekundi, pri čemu je ukupni broj poslanih poruka iznosio 513 974. Najveći dio vremena je utrošen na pristup ćelijama (38.6%) i slanje poruka kao rezultata algoritma (35.9%).

5.2 GDA metode

GetValues() metoda – Podaci o entitetima su smješteni u vidu *ResourceDescription*-a. Ova metoda na efikasan način dobavlja traženi *ResourceDescription* kojima su predstavljeni podaci o traženom entitetu. Svaki identifikacioni broj entiteta u sebi sadrži zakodiranu informaciju o tipu entiteta.

Entiteti su grupisani po tipu entiteta što omogućava da se pretraživanje vrši nad manjim skupom entiteta.

Na taj način brzo se pronalazi *ResourceDescription* odgovarajućeg entiteta. Kako bi se skup informacija koja metoda vraća sveo na one koje su potrebne, u zaglavlju metode potrebno je da se prosledi lista atributa koje treba

smjesiti u *ResourceDescription* rezultat. Ukoliko je lista atributa prazna, kao rezultat izvršavanja funkcije biće vraćen *ResourceDescription* koji sadrži samo identifikacioni broj i nema ni jedan drugi atribut. Prosleđivanje *null* pokazivača za parametar liste će rezultovati greškom, tj. desiće se izuzetak. Metoda za svaki zadati atribut provjerava da li je on definisan za tip traženog entiteta i da li traženi entitet ima definisanu vrijednost za taj atribut.

Ukoliko zadati entitet nema definisanu vrijednost za određeni atribut, a pri tome taj atribut jeste odgovarajući za tip tog entiteta, u rezultujućem *ResourceDescription*-u će biti dodan taj atribut sa nekom osnovnom, podrazumjevanom vrijednošću. Ukoliko zadati atribut nije definisan za tip traženog entiteta, metoda će rezultovati greškom. Metodom *GetValues()* se selektuje jedan određeni entitet u modelu. Kod ove metode za pronalaženje elemenata, njegova povezanost sa drugim elementima nema nikakvu ulogu.

GetExtentValues() – Ova metoda za zadati tip entiteta, na efikasan način pronalazi *ResourceDescription*-e iz odgovarajuće kolekcije koji su tog datog tipa. Takođe, zadaje se lista atributa koje treba da sadrži svaki *ResourceDescription* koji je dobijen kao rezultat izvršavanja ove metode.

Slično kao i kod *GetValues()* metode, ukoliko je lista atributa prazna, kao rezultat izvršenja metode će biti vraćeni *ResourceDescription*-i koji sadrže samo identifikator i nemaju nijedan drugi atribut. Prosleđivanje *null* pokazivača za parametar liste će rezultovati greškom, tj. desiće se izuzetak. Metoda za svaki zadati atribut provjerava da li je on definisan za traženi tip i da li svi entiteti imaju definisanu vrijednost za taj atribut.

Ukoliko neki entitet nema definisanu vrijednost za određeni atribut, a pri tome taj atribut jeste odgovarajući za traženi tip, u *ResourceDescription*-u tog entiteta će biti dodan taj atribut sa nekom podrazumjevanom vrijednošću. Ukoliko zadati atribut nije definisan za zadati tip entiteta, metoda će rezultovati greškom.

6. ZAKLJUČAK

Radom je predstavljeno uspješno modelovanje jednog distributivnog sistema upotrebom *Microsoft Graph Engine* komponente. Dodavanje *Microsoft Graph Engine* komponente u .NET i dalje rukovanje njome je vrlo jednostavno i relativno se brzo možemo naviknuti na takav način rada.

Pored toga, jedna od prednosti ove komponente je što radi direktno sa RAM memorijom, nije previše zahtjevnja za sam računar i omogućava brzo rukovanje podacima kroz ćelije. To je neophodan i optimalan način za analizu podataka u obliku grafa.

Pomoću GDA interfejsa korisniku je omogućen pristup željenim podacima. Takođe, implementacija GDA interfejsa je bila uspješan test primjene DAF/GDA standarda.

7. LITERATURA

- [1] Booch, G., Rumbaugh, J., Jacobson, I., The Unified Modeling Language User Guide, 2. Edition, Addison Wesley, May 2005.
- [2] International standard IEC 61970 – 301: Energy management system application program interface (EMS – API) – Part 301: Common Information Model (CIM) Base
- [3] Draft IEC 61970: Energy Management System Application Program Interface (EMS-API) – Part 403: Generic Data Access
- [4] Vojislav Petrović, Teorija grafova, Novi Sad, 1998.
- [5] Ian Robinson, Jim Webber, Emil Eifrem, Graph Databases, First Edition, June 2013.
- [6] B. Iordanov. Hypergraphdb: a generalized graph database. In Proceedings of the 2010 international conference on Web-age information management, WAIM '10, pages 25–36, Berlin, Heidelberg, 2010. Springer-Verlag.
- [7] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. Commun. ACM, 51(1):107–113, Jan. 2008.
- [8] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In Proceedings of the 2010 international conference on Management of data, SIGMOD '10, 2010.
- [9] https://turi.com/products/create/open_source.html (datum pristupa 25.09.2016.)
- [10] Bin Shao, Haixun Wang, and Yatao Li. Trinity: A Distributed Graph Engine on a Memory Cloud. In SIGMOD 2013: Proceedings of SIGMOD 2013, New York, New York, USA, June 22-27, 2013, pages 505-516.

Kratka biografija



Nedeljko Stojaković rođen je u Bos. Gradišci, Republika Srpska (BiH) 1991. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnika i računarsvo – Automatika i upravljanje sistemima odbranio je 2016. godine.

IONIC ОКВИР ЗА РАЗВОЈ ХИБРИДНИХ МОБИЛНИХ АПЛИКАЦИЈА**IONIC HYBRID MOBILE APPLICATION FRAMEWORK**

Душко Алексић, Факултет техничких наука, Нови Сад

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – Тема мастер рада спада у дисциплину програмирања мобилних апликација. У њему су анализирани оквири за развој хибридних мобилних апликација. Међу анализираним оквирима је изабран Ionic оквир. Описан је поступак развоја хибридне мобилне апликације коришћењем овог оквира, као и апликација која је резултат овог процеса.

Abstract – The subject of the master thesis belongs to the discipline of mobile app development. The thesis analyses hybrid mobile app development framework and chooses Ionic framework as the framework best suited for development of mobile apps. It describes the process of developing mobile app using this framework and the app that is the result of this process

Кључне речи: мобилне апликације, хибридне апликације, Ionic, Cordova, AngularJS, JavaScript

1. УВОД

Већину мобилних уређаја покрећу два доминантна оперативна система: Android [1] и iOS [2]. Савремени мобилни оперативни системи омогућили су релативно једноставан развој мобилних апликација, које је могуће извршити на великом броју различитих типова уређаја (мобилним телефонима, табличним рачунарима, паметним сатовима, телевизорима, аутомобилима, итд.). Временом се искристалисало неколико парадигми за развој мобилних апликација.

1.1 Класификација мобилних апликација

Постоје три основна типа мобилних апликација, која се разликују по начину развоја и извршавања. То су:

- матичне (енг. *native*)
- веб
- хибридне

Матичне апликације пружају најбоље перформансе и најбоље свеукупно мобилно искуство. Ово постижу захваљујући томе што су развијане за одређену платформу. Мана овог типа мобилних апликација је да развој апликације за више платформи захтева поновно писање апликације на програмском језику специфичном за ту платформу што значајно повећава трошкове развоја.

Веб мобилне апликације су веб апликације дизајниране да раде на малим екранима. Као такве, веб апликације су способне да раде на различитим

системима без посебног прилагођавања и могу их отворити било који модерни претраживачи. Предност ових апликација је лакша дистрибуција и подршка него код матичних апликација. Највећа мана овог типа мобилних апликација је да захтевају интернет конекцију како би корисник имао приступ апликацији. Хибридне апликације комбинују најбоље особине матичних и веб мобилних апликација. За имплементацију пословне логике се користе постојеће веб технологије попут HTML-a, CSS-a и JavaScript-a. Велика предност хибридних апликација у односу на матичне јесте да се једном написан код може извршавати на више различитих платформи. Тиме се значајно смањују трошкови израде апликације. Како би се омогућио лакши развој хибридних апликација, настали су развојни оквири који садрже скуп компоненти корисничког интерфејса и метода за приступ и манипулацију матичним функцијама мобилних уређаја.

1.2 Оквири за развој хибридних апликација

У последњих неколико година јавља се велико интересовање за хибридне оквири за развој мобилних апликација. Неки од популарних оквира су:

- Ionic оквир [3]
- Appcelerator Titanium [4]
- Sencha Touch [5]
- Framework 7 [6]
- Kendo UI [7]
- Onsen UI [8]

Један од најпопуларнијих оквира за развој хибридних апликација је Ionic. Поседује добру документацију и велику заједницу. Укључује велики број компоненти корисничког интерфејса које је лако прилагодити потребама. Интегрисан је део за приступ матичним функцијама мобилних уређаја, што није случај код Framework 7, Kendo UI, Onsen UI оквира. Такође, за разлику од Sencha Touch и Kendo UI оквира, Ionic је потпуно бесплатан за коришћење.

Због ових карактеристика, Ionic оквир ће бити коришћен за имплементацију апликације описане у овом раду.

У другом одељку описан је Ionic оквир. Функционални захтеви су специфицирани у трећем одељку, а у четвртном одељку је специфициран дизајн апликације. Имплементација је описана у петом одељку. У последњем одељку анализиране су предности и мане Ionic оквира.

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је био доц. др Стеван Гостојић

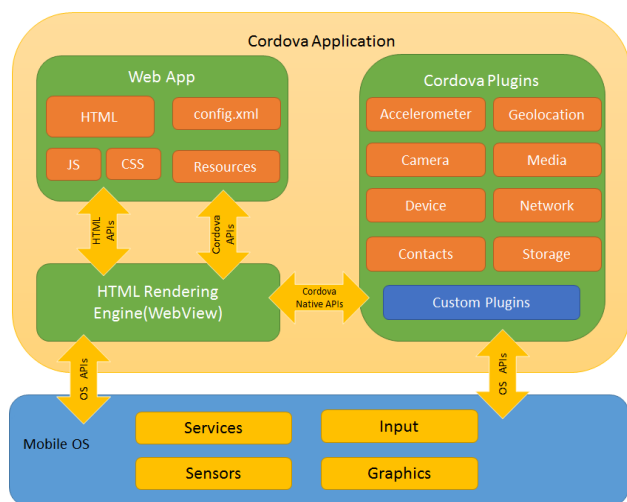
2. IONIC

Ionic је оквир за развој хибридних мобилних апликација. Коришћењем стандардних веб технологија попут *HTML*-а, *CSS*-а и *JavaScript*-а, *Ionic* омогућава развој мобилних апликација за више мобилних платформи. Апликација се извршава унутар контејнера који је специфичан за сваку платформу. Да би мобилна апликација била хибридна, мора имати две одлике. Прва је да садржи матични контејнер који омогућава приступ матичним функцијама мобилног уређаја. Матични контејнер који користи *Ionic* назива се *Cordova* [9]. Друга је да садржи компоненте које служе за креирање корисничког интерфејса и имплементацију пословне логике. У ту сврху *Ionic* користи *AngularJS* [10] оквир.

2.1 Cordova

Cordova је матични контејнер хибридних мобилних апликација. Омогућава приступ матичним функцијама платформе употребом *JavaScript*-а. Апликација се извршава унутар контејнера специфичног за сваку платформу [9].

Архитектуру мобилне апликације развијене помоћу *Cordova* оквира чине неколико компоненти које чине архитектуру апликације на највишем нивоу апстракције (слика 2.1) [9]. То су веб апликација, *WebView* [9] и прикључци.



Слика 2.1 Архитектура *Cordova* апликације [9]

2.2 AngularJS

AngularJS је *JavaScript* оквир за развој *single page* веб апликација. Олакшава раздвајање пословне логике, модела података и шаблона за приказ података, манипулисања историјом веб претраживача и *dependency injection*.

Овај оквир користи неколико концепата који *AngularJS* чине веома моћним овиром за развој веб апликација. Најважнији од њих су:

- **шаблон** (енг. *template*) – *HTML* странице са додатним ознакама;
- **директива** (енг. *directive*) – омогућава проширење синтаксе *HTML*-а са новим елементима и атрибутима;
- **модел** (енг. *model*) – подаци који су се обрађују;

- **контролер** (енг. *controller*) – пословна логика иза погледа;
- **опсег** (енг. *scope*) – контекст у коме је модел смештен;
- **израз** – приступ променљивама и методама из опсега;
- **поглед** (енг. *view*) – *HTML* контејнер који приказује апликацију;
- **повезивање података** (енг. *data binding*) – синхронизација података између модела и погледа;
- **модул** (енг. *module*) – контејнер за различите делове апликације као што су контролери, директиве, сервиси, филтери, итд.;
- **сервис** (енг. *service*) – пословна логика независна од погледа;
- **филтер** (енг. *filter*) – дефинише формат за приказ података;

2.3 Основе Ionic оквира

У овом поглављу биће описан поступак инсталације *Ionic* оквира, генерисање новог пројекта и структура пројекта. Такође, описана је и инсталација *Cordova* прикључака за приступ матичним функцијама уређаја. На крају је приказано како се апликација покреће на мобилном уређају и на претраживачу.

2.3.1 Генерисање новог пројекта

Како би имплементирали нову апликацију, прво се мора генерисати нови *Ionic* пројекат. За то ће послужити раније инсталирани алати, чије команде се извршавају из конзоле. Потребно је извршити команду:

```
$ ionic start <putanja> [naziv šablona]
```

Као параметри се прослеђују путања са називом новом пројекта и врста шаблона. Резултат ове команде је нови пројекат.

2.3.2 Структура апликације

Након генерисања новог пројекта помоћу конзолне команде, направљена је иницијална структура пројекта која се неће много мењати у току даљег рада на пројекту. Када се додају функције специфичне за платформу, креираће се додатни фолдер под називом *platforms*. Иницијална структура састоји се од:

- **hooks** – садржи прилагођене *Cordova* пречице за извршавање специфичних команди;
- **plugins** – садржи инсталиране *Cordova* прикључке;
- **scss** – садржи стилове апликације које је могуће модификовати;
- **www** – садржи пословну логику апликације, структуру корисничког интерфејса и стилове
- **.bowerrc** – садржи вредност путање где ће бити инсталиране додатне библиотеке (обично има вредност *www/lib*);
- **bower.json** – садржи листу додатних библиотека које су потребне;
- **config.xml** – поред информација о апликацији и аутору, садржи и додатна подешавања пројекта која је могуће променити и
- **ionic.project** – дефинише назив и јединствени идентификатор апликације

2.3.3 Покретање апликације

Као што је већ споменуто, *Ionic* је овир за креирање хибридних мобилних апликација. Ова карактеристика омогућава апликацији да се по потреби понаша или као мобилна апликација или као веб апликација.

Да би се апликација извршила у претраживачу, потребно ју је покренути следећом командом:

```
$ ionic serve
```

Команда покреће локални сервер и отвара нови прозор претраживача. Након чувања измена аутоматски се освежава приказ.

Како би се апликација покренула на физичком уређају (и могла да користи све функције које уређај пружа), неопходно је извршити команду:

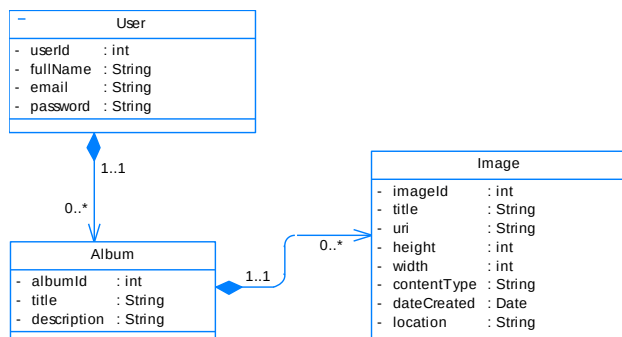
```
$ ionic run <naziv platforme>
```

3. СПЕЦИФИКАЦИЈА ЗАХТЕВА

Као пример употребе *Ionic* оквира имплементирана је фото-албум мобилна апликација. Апликација подржава регистрацију и пријаву корисника. Омогућава кориснику креирање, измену и брисање албума. Такође, омогућава прављење фотографија, смештање фотографија у жељени албум, приказ фотографија и брисање фотографија.

4. СПЕЦИФИКАЦИЈА ДИЗАЈНА

За фото-албум мобилну апликацију су нам потребне три класе: *User* (корисник), *Album* (албум) и *Image* (фотографија). Слика 4.1 приказује дијаграм класа фото-албум апликације.



Слика 4.1 Дијаграм класа фото-албум апликације

Класа *User* представља корисника апликације. Садржи *userId* (јединствени идентификатор), *fullName* (пуно име корисника), *email* (e-mail адреса корисника) и *password* (шифра корисника).

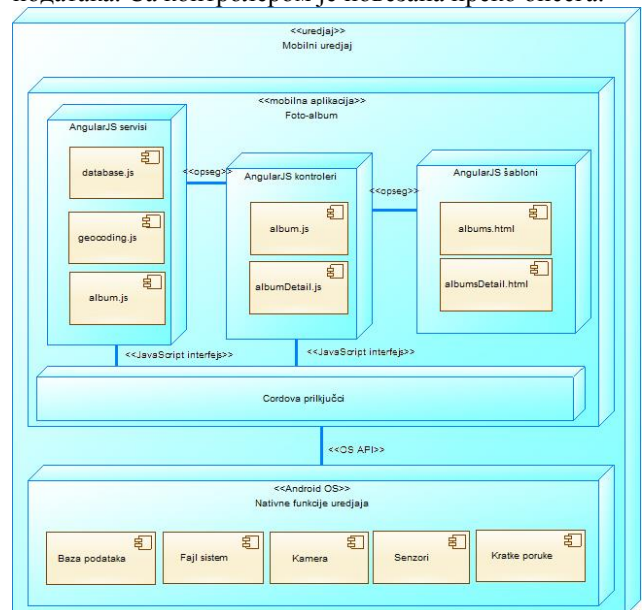
Класа *Album* садржи *albumId* (јединствени идентификатор), *title* (назив албума) и *description* (опис албума).

Класа *Image* представља сачувану фотографију. Садржи *imageId* (јединствени идентификатор), *title* (назив фотографије), *uri* (путању на којој је сачувана фотографија), *height* (висину фотографије), *width* (ширину фотографије), *contentType* (формат фотографије), *dateCreated* (датум креирања фотографије) и *location* (локација на којој је направљена фотографија).

Дијаграм распореда приказује податке о распореду компоненти апликације (слика 4.2). Главни чвор на дијаграму је мобилни уређај. Садржи две компоненте:

оперативни систем и мобилну апликацију. Оперативни систем чине скуп компоненти матичних функција уређаја (база података, фајл систем, камера, сензори, итд.).

Компонента *Cordova* прикључака чини скуп прикључака који омогућавају приступ матичним функцијама уређаја. Пословна логика апликације састоји се од сервиса и контролера имплементираних у *AngularJS* оквиру. Преко *JavaScript* интерфејса приступају *Cordova* прикључцима. Подаци се приказују и преузимају од стране корисника помоћу компоненте која представља шаблоне за приказ података. Са контролером је повезана преко опсега.



Слика 4.2 Дијаграм распореда мобилне апликације

5. ИМПЛЕМЕНТАЦИЈА

Ово поглавље описује кључне делове имплементације фото-албум мобилне апликације помоћу *Ionic* оквира. Посебно је приказано како апликација приступа матичним функцијама уређаја.

5.1 Рад са базом података

Како би сачували информације о кориснику, албумима и фотографијама, потребно је искористити базу података. Да би се ова функционалност могла искористити потребно је инсталирати *Cordova* прикључак за рад са базом. За инсталацију прикључка потребно је извршити следећу команду:

```
$ ionic plugin add cordova-sqlite-storage
```

Након успешне инсталације, могућ је рад са базом података. Основна функција за извршавање упита над базом је *execute*. Као параметре има *database* (базу података), *query* (упит) и *parameters* (низ параметара).

5.2 Кратке поруке

Када се заврши акција коју је изазвао корисник, потребно је обавестити корисника о статусу акције. У ту сврху се користи *toast* порука. То је компонента матичног дела систем. То значи да је потребно инсталирати одговарајући *Cordova* прикључак. Следећом командом врши се његова инсталација:

```
$ ionic plugin add cordova-plugin-x-toast
```


За приказ повратне поруке позива се метода *show*. Као параметри се прослеђују порука која се жели приказати, временски интервал приказа (може бити *short* или *long*) и позиција поруке на екрану (може бити *top*, *center* или *bottom*).

5.3 Фотографисање

Фото-албум апликација чува корисничке фотографије у структури албума. Да би корисник могао правити фотографије помоћу свог уређаја, потребно је инсталирати *Cordova* прикључак. Следећа команда ће инсталирати прикључак за приступ камери уређаја:

```
$ ionic plugin add cordova-plugin-camera
```

Активност камере уређаја се приказује позивом методе *getPicture*. Метода поред метода које се позивају након успешно или неуспешно извршене операције, прима објекат који садржи опције.

5.4 Преузимање локације уређаја

У спецификацији фото-албум апликације, класа *Image* садржи атрибут *location*. Да би се при чувању фотографија преузела локација уређаја, потребно је искористити посебан *Cordova* прикључак. Инсталација се извршава следећом командом:

```
$ ionic plugin add cordova-plugin-geolocation
```

Како би се преузела локација, потребно је укључити дозволу за локацију у подешавањима уређаја. Затим је потребно је позвати методу *getCurrentPosition*. Као параметар треба проследити објекат који садржи информацију о *timeout*-у и нивоу прецизности локације. Повратна вредност методе је објекат који садржи информацију о географској дужини и географској ширини.

5.5 Рад са фајл системом

Након рада на фотографији, потребно ју је сачувати. Да би се направио фајл и сачувао у меморији уређаја, неопходно је искористити *Cordova* прикључак за рад са фајл системом. Следећом командом се врши инсталација:

```
$ ionic plugin add cordova-plugin-file
```

Прикључак обезбеђује методе за креирање, читање, упис фајлова, креирање директоријума, итд.

6. ЗАКЉУЧАК

Задатак овог рада био је анализа *Ionic* оквира и његова примена за развој хибридних мобилних апликација. *Ionic* оквир се показао јако добро за развоју корисничког интерфејса зато што се користе опробане технологије као што су *HTML*-а и *CSS*-а. Компонете показују велику флексибилност и могуће их је лако прилагодити жељеном дизајну. Уз помоћ прикључака могућ је приступ матичним функцијама уређаја.

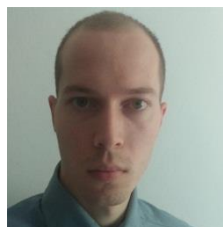
Када се све узме у обзир, стиче се утисак да овакав тип апликација даје најбоље резултате ако се користи за конзумирање и приказ података. Једноставно, процесна моћ *JavaScript*-а који се извршава у оквиру „танког“ матичног контејнера је далеко мања од процесне моћи матичне апликације. Свака захтевнија обрада података може осетно успорити рад апликације што може имати утицај на крајњег корисника.

Што се тиче даљег развоја апликације описане у овом раду, могуће ју је унапредити имплементацијом синхронизације података са базом података у облаку, интеграцијом апликације са друштвеним мрежама и проширивањем скупа ефеката који се могу применити на фотографији.

6. ЛИТЕРАТУРА

- [1] Android, <https://www.android.com/>
- [2] iOS, <https://www.apple.com/ios>
- [3] Ionic framework, ionicframework.com/
- [4] Appcelerator Titanium, www.appcelerator.com/
- [5] Sencha Touch, <https://www.sencha.com/products/touch/#overview>
- [6] Framework 7, <https://framework7.io/>
- [7] Kendo UI, www.telerik.com/kendo-ui
- [8] Onsen UI, <https://onsenui.io/>
- [9] Cordova, <https://cordova.apache.org/>
- [10] AngularJS, <https://angularjs.org/>
- [11] Node.js, <http://nodejs.org/>
- [12] npm, <https://www.npmjs.org/>

Кратка биографија:



Душко Алексић рођен је у Лозници 1991. године. Дипломски рад из области Електротехнике и рачунарства одбранио је 2014. године. на Факултету техничких наука у Новом Саду. Положио је све ипите предвиђене студијским програмом.

ZAŠTITA VISOKONAPONSKIH ASINHRONIH MOTORA PROTECTION OF HIGH VOLTAGE ASYNCHRONOUS MOTORS

Aleksandar Kerekeš, Veran Vasić, Đura Oros, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu prikazana je primena savremenih mikroprocesorskih zaštitnih uređaja čija funkcija je da zaštite visokonaponske asinhronne motore. Rad objašnjava zbog čega se vrši zaštita motora i na šta je potrebno obratiti posebnu pažnju pri primeni ovakvog oblika zaštite. Ukratko su opisani standardi koji se primenjuju pri realizaciji relejne zaštite, kao i mogućnosti koje ova zaštita pruža korisniku. Takođe rad se osvrće na bezbednost podataka u sistemu i na potrebu zaštite od sajber napada koji mogu prouzrokovati veliku štetu u današnjim pogonima zaštićenim ovakvim oblikom relejne zaštite. Na kraju je dat primer primene ovakve zaštite kao i rezultati sa terena u obliku vremenskih karakteristika snimljenih direktno na ovakvim zaštitnim uređajima koji su bili povezani na sistem sa 6kV motorima.

Abstract – This paper describes the application of modern microprocessor protection devices whose function is to protect the high-voltage asynchronous motors. The work describes why the protection of the engine is carried out and which facts have to be specially considered when this form of protection is implemented. It briefly describes the standards that are applied in the implementation of relay protection, as well as the opportunities that this protection provides to the user. Also the work refers to the security of data in the system and the need to protect it against cyber attacks that can cause great damage in today's drives protected with this form of relay protection. At the end an example of application of this sort of protection is given as well as the results obtained in the field in form of temporal characteristics recorded directly on these protective devices that were connected to the system with 6 kV motors.

Ključne reči: Relejna zaštita, Mikroprocesorski zaštitni uređaji, MPZU, zaštita 6kV asinhronih motora, prekostrujna zaštita motora, kratkospojna zaštita, podnaponska zaštita, Relay protection.

1. UVOD

Početkom dvadesetog veka počinje primena relejne zaštite, gde se pojam relej koristi za uređaj koji je napravljen tako da reaguje na određeni način kada se na njegove ulaze dovede određena mehanička, električna ili neka druga pobuda, a taj proces je ponovljiv bez oštećenja uređaja. Neki autori [1] opisuju (mikroprocesorske) releje kao uređaje koji mogu da zamene dva ili više elek-

tromehanička releja. Kombinacijom nekoliko zaštitnih funkcija u jednom kućištu, digitalni releji postaju ekonomski povoljniji pri nabavci, implementaciji, a naročito pri održavanju, u odnosu na elektromehaničke releje.

Danas imamo slučaj da su delovi od kojih je sastavljen elektroenergetski sistem znatno osetljiviji na kvarove u odnosu na ranije proizvedene. Ono što se tu postavlja kao izazov jeste da se upotrebom mikroprocesorske zaštite obezbede sve potrebne zaštitne funkcije kako bi sistem bio pouzdan. To veoma često uvodi zahteve za merenje nekih novih veličina, kao što su senzori za detekciju strujnog luka (eng. Arc Flash sensor) i sl.

Zaštita kompletne opreme (generatora, transformatora, prenosnih vodova i sabirnica) od kratkog spoja je od suštinskog značaja.

Opšti princip jeste da se detektuje kvar (kratak spoj) i donese odluka o izolovanju mesta kvara (oprema sa kvarom) od ostalog dela sistema tako da oprema u ostatku sistema ne pretrpi oštećenja. Osnovna komponenta zaštite je topivi osigurač, koji u svom najjednostavnijem obliku detektuje kvar, pregoreva i na taj način izoluje kvar. U svom najsophisticiranijem obliku, mikroprocesorski zaštitni relej može da detektuje kvar i izvrši analizu i promeni uklopna stanja prekidača u sistemu napajanja tako da se omogući izolovanje kvara.

Pored tih osnovnih funkcija, ovi releji poseduju i mnoge druge mogućnosti kao što je povezivanje u jedan centralizovani sistem, sa kojeg postoji mogućnost manipulacije sklopnom opremom, ostvarivanja međusobne selektivnosti releja otpičkom komunikacijom, realizacija kompletne relejne logike za automatsko upravljanje unutar samog releja. Na slici 1 prikazan je današnji izgled mikroprocesorskih releja različitih proizvođača.



Slika 1: Današnji izgled mikroprocesorskih zaštitnih različitih proizvođača

Ništa od zaštitnih funkcija ovi releji ne bi mogli da realizuju ako ne bi imali mogućnosti merenja visokih napona i velikih struja. Stoga se putem mernih transformatora vrši redukcija sa pravih vrednosti na standardizovane ulazne vrednosti ovih releja.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Đura Oros, vanr. prof.

2. STRUJNI I NAPONSKI MERNI TRANSFORMATORI

Strujni transformatori vrše transformaciju struje sa vrednosti koja teče kroz vod, generator ili transformator na vrednosti koje omogućavaju upotrebu releja i mernih instrumenata, koji su dimenzionisani najčešće tako da im je naznačena struja 1A ili 5A. Na teritoriji Evrope ovi transformatori su najčešće induktivni.

Prema preporukama IEC (Current transformers – Publication 185/1966) i standardima JUS N.H9. 101, 102 i 103 u šest klasa tačnosti: 0,1; 0,2; 0,5; 1; 3 i 5. Oznaka klase tačnosti jednaka je apsolutnoj vrednosti granice strujne greške u procentima pri nazivnom opterećenju i 120% primarne struje [2].

Naponski merni transformatori transformišu primarne napone elemenata elektroenergetskog sistema (EES) na vrednosti koje su u skladu sa naznačenim vrednostima napona mernih i zaštitnih uređaja, i vrše galvansko odvajanje.

Za napone do 110kV upotrebljavaju se induktivni naponski transformatori, a za više napone, radi ekonomičnosti, kapacitivni NT, kao kombinacija kapacitivnog razdelnika napona i induktivnog naponskog transformatora.

Većina propisa za merne transformatore predviđa pet klasa tačnosti: 0,1; 0,2; 0,5; 1; i 3,0. Oznaka klase tačnosti jednaka je apsolutnoj vrednosti granice naponske greške u procentima između $0,8U_n$ i $1,2U_n$ i opterećenja od 25% do 100% prema IEC preporukama (publikacija 186 i 186A iz 1969. i 1970. godine) [2].

Potrebno je istaći da struje i naponi sa sekundarnih strana mernih transformatora predstavljaju glavni dovod informacija vezanih za stanje električnih veličina na štićenom objektu.

3. ZAŠTITNI RELEJI

Osnovni element zaštitnih relejnih sistema je relej. Relej je uređaj koji ima: merni sistem koji služi za detekciju kontrolisane veličine, sistem za upoređivanje (koji upoređuje izmerenu veličinu sa podešenom) i izvršni član koji izaziva trenutnu ili vremenski zakašnjenu promenu u komandnim i signalnim krugovima. Prema principu delovanja mernog člana releji mogu biti [3]:

- 1) Elektromagnetski
- 2) Indukcioni
- 3) Elektrodinamički
- 4) Magnetnoelektrični
- 5) Termički
- 6) Statički
- 7) Mikroprocesorski
- 8) Adaptivni

Mikroprocesorski releji imaju veliku prednost (u odnosu na ranije korišćene releje) jer imaju mogućnost da se povežu na sistem računarskih mreža (SCADA system - Supervisory control and data acquisition system), koji danas otvara mnoge mogućnosti kao što su daljinska promena parametara releja, prikupljanje podataka celokupnog sistema u realnom vremenu i sl.

Ono što je loša strana ovakvog računarskog sistema jeste to da je moguća njegova zloupotreba putem računarskih napada na sistem (sajber napadi).

3.1 Zaštita od sajber i fizičkih napada sistema sa relejnom zaštitom

U novije vreme postoji trend sve veće implementacije sistema tzv. pametnih mreža. Ovakav sistem je i danas u razvoju jer se nastoji da se računarskim sistemima predvide i minimalizuju kvarovi i opasna pogonska stanja. Aktivnosti koje ugrožavaju bezbednost takozvanih pametnih mrežnih sistema (SG - Smart grid sistem) se mogu podeliti na sajber i fizičke napade. Mere fizičke zaštite SG kao i zaštite podataka koje postoje, pored toga i razvijeni algoritmi za informacionu bezbednost sistema ne uzimaju u obzir posledice koje ovakvi napadi mogu izazvati na fizičkim svojstvima sistema, kao što ni algoritmi za fizičku bezbednost ne obuhvataju posledice koje su moguće na nivou informacione strukture sistema. Stoga se nameće zaključak da metodi fizičke i informacione zaštite SG mreže nisu dovoljni da obezbede zaštitu od napada. Pojavljuje se potreba za razvojem sajber-fizičkih modela i metoda kojima bi se uspešno zaštitio sistem od svih vrsta napada uključujući i hibridne napade [4]. Današnja literatura objašnjava korišćenje raznih metoda kako bi se izbegli napadi na ovakve SG sisteme, jedan od predloženih metoda jeste da se filtrira frekvencija komponenti male snage [5]. Neki drugi pristup bi bio da se primene protokoli koji se zasnivaju na „Zero knowledge proof“ [6]. Na taj način se omogućava obračun potrošnje električne energije bez otkrivanja detalja o mernim podacima. I možda još jedno interesantno rešenje jeste primena mehanizma kontrole pristupa [7], na osnovu čijeg principa se omogućava (ili zabranjuje) da pojedine komponente SG sistema mogu da pristupe podacima korisnika.

Najznačajniji skup standarda kojima se definiše informaciona zaštita SG sistema danas sadržana je u sledećim dokumentima [8]:

- 1) ISO/IEC 27001
- 2) ISO/IEC 27002
- 3) IEC 62351
- 4) NERC CIP (US Standard)
- 5) NISTIR-7628 (US Guidelines).

4. ZAŠTITA ASINHRONIH MOTORA

Kao i kod ostalih električnih mašina, tako i kod motora razlikujemo stanja sa kvarom i opasna pogonska stanja. Te dve pojave je potrebno razdvojiti iz prostog razloga što se pri uspostavljanju istih vrše različiti postupci za povratak pogona u rad.

Potencijalna opasna stanja, ili kvarovi koji se obično dešavaju jesu:

- 1) Kratak spoj između dve faze ili kratak spoj sa zemljom (zemljospoj)
- 2) Zaključan rotor (greška pri pokretanju motora)
- 3) Prenizak ili previsok napon napajanja
- 4) Neodgovarajući redosled faza
- 5) Neodgovarajući ambijentalni uslovi (visoka koncentracija prašine, visoka ili niska temperatura)
- 6) Više uzastopnih (nedovršenih) polazaka motora
- 7) Gubitak faze napajanja
- 8) Mehanički kvarovi, kvarovi u ležajevima i sl.

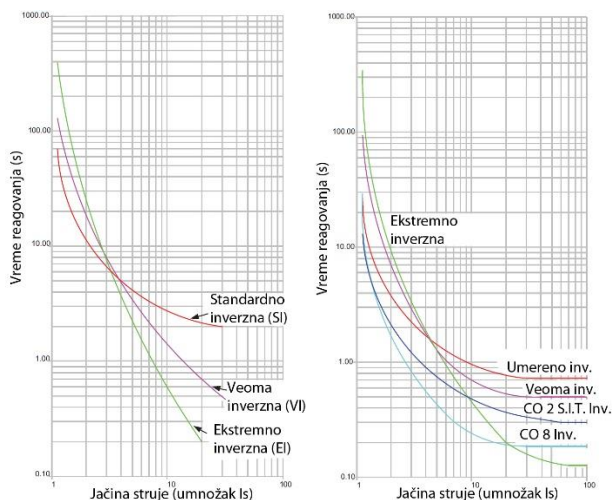
9) Preopterećenje (usled visokog momenta inercije ili velikog opterećenja)

Na slici 2, prikazane su standardne krive koje se koriste za zaštitne releje. Levi dijagram prikazuje krive propisane IEC standardom, dok su desne krive propisane američkim ANSI/IEEE standardom.

Kombinacijom ovih krivih i njihovim karakterističnim podešenjima dolazi se do najboljeg rešenja, koje se zatim programira na releju.

Kako bi se izbegli pomenuti karovi, i njihova dalja eskalacija koja može prouzrokovati potpuno uništenje opreme, pristupa se realizaciji standardnih tipova zaštite u elektroenergetskim postrojenjima:

- 1) Zaštita od preopterećenja motora
- 2) Zaštita od kratkog spoja motora između namotaja
- 3) Zaštita od kratkog spoja sa zemljom
- 4) Podnaponska zaštita
- 5) Diferencijalna zaštita namotaja motora (koristi se kod velikih mašina)
- 6) Zaštita od nesimetrije struje



Slika 2: Prikaz karakteristika propisanih standardom

Sve pomenute zaštite se uvode samo iz razloga da se što pre uoči kvar i značajno umanju šteta pri kvarovima, koji su u jednom radnom pogonu neizbežni.

6. PRIMER PRIMENE RELEJNE ZAŠTITE MOTORA

U ovom primeru opisana je relejna zaštita primenjena na motoru mlina, novog postrojenja unutar flotacije Majdanpek, koji pripada RTB (Rudarsko topioničarskom basenu Bor).

Za proračune pri stacionarnim stanjima kod asinhronih motora je moguće koristiti standardnu ekvivalentnu šemu motora koja služi za proračune karakterističnih veličina motora, na osnovu kojih mogu biti određeni parametri zaštite motora.

Nazivni podaci motora:

390kW, 6kV, 47A, 985o/min, 0.86

U ovom primeru proračunata su sledeća podešenja relejne zaštite:

Prekostrujna zaštita I> (ANSI 51)

Motor mlina se napaja sa sabirnicom na kojima se nalazi prekidač sa prekostrujnom i kratkospojnom zaštitom.

S obzirom da se u praksi uglavnom postupa tako da se motori dimenzionišu na veće snage od onog što je predviđeno, predlaže se da podešenja motora budu usklađena sa nominalnom strujom krivom koja odgovara dozvoljenom vremenu preopterećenja motora.

Long Inverse kriva (IEC LTI/B)

Isett=47A

T = 2 s

Kratkospojna zaštita I>> (ANSI 50)

Vremenski nezavisna kriva. Nju je potrebno podesiti na vrednostima struje između 165% i 190% od struja pri zakočenom rotoru. [9]

Izak=6x47A= 282A

Isett = 1,9x282A=535,8A

tp = 0 s

Zemljospojna zaštita Io> (ANSI 51N)

Realizacija ove zaštite izvršena je putem obuhvatnog transformatora.

Za ovu zaštitu standardno se koristi vremenski nezavisna kriva.

Standardno podešenje ove zaštite jeste na 30% od nominalne vrednosti struje. [10]

U ovom primeru data su podešenja koja je investitor zahtevao.

I0_sett = 5,00A

tp = 20s

Aktivacija zaštite uključuje alarm, ali ne isključuje prekidač motora.

Zaštita od nesimetrije struja (ANSI 46)

Ova zaštita se realizuje u dva stepena, gde prvi stepen (vremenski nezavisna kriva) aktivira alarm i ona je podešena obično na 20% inverzne komponente sa zakašnjenjem od 30s. Drugi stepen ove zaštite (vremenski zavisna kriva - IDMT) sa podešenjima TMS=1.0 šalje signal za isključenje prekidača.

Prikazana podešenja su u skladu sa zahtevima investitora.

Isett =5A

tp = 10s

Podnaponska zaštita:

Standardna podešenja ove zaštite se vode time da se motor uspešno može startovati sa naponom koji je iznad 80%Un, s tim u vezi ova zaštita je podešena na sledeću vrednost. [10].

Usett=80% UN

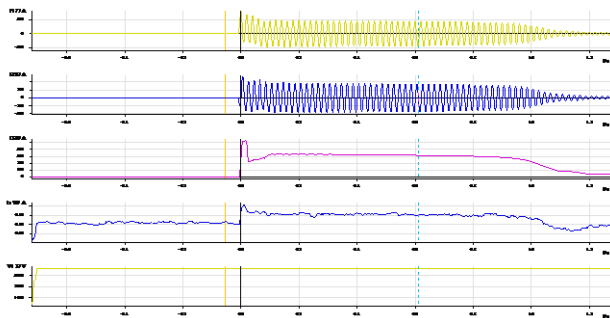
tp = 4s

Na dijagramima sa slike 3 se mogu videti veličine polaznih struja koje su snimljene na terenu. Sa tog dijagrama se može očitati vrednost polazne struje asinhronih mašina, kao i vrednosti napona pri polasku motora. Ovaj dijagram samo potvrđuje tačnost proračunatih podataka vezanih za zaštitne funkcije releja korišćenog na ovom motoru.

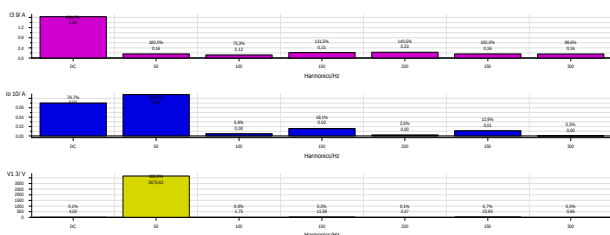
Ovakvi dijagrami se mogu koristiti kao veoma koristan alat za proveru podešenih vrednosti releja. Ukoliko se sa dijagrama utvrdi da one nisu adekvatne, vrši se korekcija podešenih parametara.

Ono što je pored vremenskih dijagrama potrebno razmotriti jeste svakako vrednost svih harmonika u ovoj situaciji, gde nam ta informacija može biti od velikog

značaja. Na slici 4 prikazani su razloženi harmonici struje namotaja, struje zemljospoja kao i napona pri polasku motora.



Slika 3: Dijagrami karakterističnih veličina pri polasku motora



Slika 4: Dijagrami vrednosti pojedinačnih harmonika karakterističnih veličina pri polasku motora

7. ZAKLJUČAK

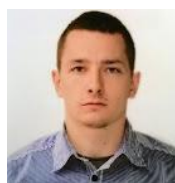
Realizacija zaštite visokonaponskih motora primenom mikroprocesorskih releja u načelu ne izgleda kao složen i komplikovan proces. U praksi se često dešava da studija podešenja koja se bazira na teorijskim proračunima ne predstavlja adekvatno rešenje. Ne retko je potrebno i odlično poznavanje ostatka elektroenergetskog postrojenja kako bi se uspešno realizovao zaštitni sistem visokonaponskih motora. Današnje savremene tehnologije primene mikroprocesorskih zaštitnih relea nam omogućava pored jednostavnog podešenja pojedinih vrsta zaštite i snimanje dijagrama karakterističnih veličina motora u vremenu koje se dobijaju startovanjem motora. Na osnovu snimljenih dijagrama može se izvršiti provera podešenosti parametara relea, odnosno izvršiti korekcija prvobitno podešenih parametara za ostvarenje željenih zaštita. Na taj način postizemo veću pouzdanost rada pogona sa visokonaponskim motorima i bezbednost i motora i ostalih komponenti energetskog sistema na koji je motor povezan.

8. LITERATURA

1. Mooney, Joe, Microprocessor-Based Transmission Line Relay Applications. American Public Power Association's Engineering & Operations Workshop. Salt Lake City, Utah: Schweitzer Engineering Laboratories, Inc. p. 1.) (March 25–28, 1996)

2. Vojislav Bego, Glava 7 - MJERNI TRANSFORMATORI. U: Mjerenja u elektrotehnici (treće prošireno izdanje), Zagreb: Tehnička knjiga Zagreb; 1975. 08-1119/2-1975
3. Duško Bekut, Glava 1 UVOD. U: Relejna zaštita. Novi Sad: Fakultet tehničkih nauka; 2009. 621.316.925(075.8). ISBN 978-86-7892-169-8
4. Yilin Mo, T.H.-J. Kim, K. Brancik, D. Dickinson, Heejo Lee, A. Perrig, B. Sinopoli, "Cyber-Physical Security of a Smart Grid Infrastructure," in Proc. IEEE, Vol. 100, No. 1, Jan. 2012, pp. 195-209.
5. S. R. Rajagopalan, L. Sankar, S. Mohajer, and H. V. Poor, „Smart meter privacy: A utility-privacy framework,” IEEE SmartGridComm 2011, To appear. Available at <http://arxiv.org/abs/1108.2234>
6. A. Rial and G. Danezis, „Privacy-preserving smart metering,” technical Report MSR-TR-2010-150, Microsoft Research, 2010.
7. R. Bobba , H. Khurana , M. AlTurki , F. Ashraf, „PBES: a policy based encryption system with application to data sharing in the power grid,” Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, Sydney, Australia, March 10- 12, 2009.
8. „CEN-CENELEC-ETSI Smart Grid Coordination Group Smart Grid Information Security” CEN-CENELECETSI Smart Grid Coordination Group, November 2012.
9. Reimert Donald „Protective relaying for power generation systems“ ISBN-13: 978-0-8247-0700-2 (alk. paper), ISBN-10: 0-8247-0700-1 (alk. paper), 2006 by Taylor & Francis Group, LLC
10. Simon Richards „NETWORK PROTECTION & AUTOMATION GUIDE“, Third edition, ISBN: 978-0-9568678-0-3, May 2011, Published by Alstom Grid

Kratka biografija:



Aleksandar Kerekeš rođen je u Pančevu 1992. god. Srednju elektrotehničku školu završio je u Zrenjaninu, a osnovne studije je završio 2014. na Fakultetu tehničkih nauka u Novom Sadu iz oblasti Elektrotehnike i računarstva – Energetska elektronika i električne mašine.

PRORAČUN SIMETRIČNIH TOKOVA SNAGA DISTRIBUTIVNIH MREŽA PRIMENOM POSTUPKA SUMIRANJA STRUJA I KOREKCIJA NAPONA**CURRENT SUMMATION AND VOLTAGE CALCULATION LOAD FLOW FOR DISTRIBUTION NETWORKS**

Kristian Haška, Nikola Vojnović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu obrađen je postupak sumiranja struja i korekcija napona za proračun simetričnih tokova snaga radijalnih distributivnih mreža i mreža sa petljama. Model mreže je izveden direktnom primenom strujnog i naponskog Kirchhoff-ovog zakona. Na osnovu algoritma za proračun tokova snaga koji je izveden, napisan je računarski program za proračun tokovova snaga distributivnih mreža s proizvoljnim brojem čvorova i petlji.

Abstract – This paper deals with power flow calculations of radial and weakly meshed distribution networks using current summation and voltage calculation method. The network model is derived by direct application of Kirchhoff's current and voltage laws. The corresponding calculation algorithm for power flow calculation is derived. Based on this algorithm a computer program is also developed.

Ključne reči: Distributivna mreža, Gauss/Seidell-ov metod, Kirchhoff-ovi zakoni, Thévenin/Norton-ova teorema, princip supstitucije, proračun tokova snaga.

1. UVOD

Proračuni tokova snaga najčešći su proračuni elektroenergetskih mreža. Uobičajeni proračuni elektroenergetskih mreža, pa i problemi tokova snaga, sastoje se od sledeće četiri komponente: matematički model mreže, matematički metod za proračun modela mreže, utvrđivanje procedure – algoritma za proračun stanja mreže, računarski program.

Prenosne mreže jesu trofazne mreže s relativno velikim brojem "petlji" – "upetljane". Proračuni tokova snaga tih mreža utvrđeni su pre više decenija [1]. Matematički modeli tih mreža napisani su saglasno sa metodom nezavisnih napona [2, 3]. Ti proračuni su orijentisani na čvorove. Njihova rešenja su zasnovana na klasičnom Newton/Raphson-ovom metodu za rešavanje sistema nelinearnih jednačina.

S malim brojem izuzetaka, distributivne mreže jesu "radijalne" ili "mreže s malim brojem petlji" – "slabo upetljane mreže". U osnovi većine postupaka proračuna tokova snaga ovih mreža leže nepoznate struje grana mreže [4, 5]. Zbog toga se kaže da su ti postupci orijentisani na grane.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Vladimir Strezoski.

Ovde je obrađen postupak sumiranja struja i korekcije napona radijalnih mreža i mreža sa petljama. U ovom radu reč je o stacionarnim tokovima snaga uravnoteženih distributivnih mreža sa petljama. Radijalne mreže su tretirane kao specijalan slučaj mreža s petljama. Mreže su u simetričnim režimima direktnog redosleda. Zato njihovi režimi mogu da se tretiraju koristeći se pogonskim šemama direktnog redosleda. Oni se odnose na režime jedne faze, npr. faze a [3].

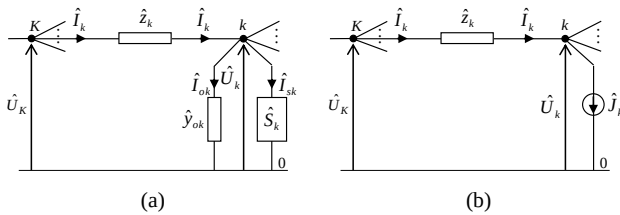
U ovom radu je reč o izvođenju zasnovanosti proračuna tokova snaga distributivnih mreža na jedan neposredan način. Prvo su obrađene radijalne mreže – deo 2, a teorijska zasnovanost proračuna tokova snaga mreža sa petljama je zatim svedena na problem radijalnih mreža – deo 3. To je učinjeno sintezom tri električne institucije: 1) dva Kirchhoff-ova zakona [2], 2) princip supstitucije [3] i 3) Thévenin/Norton-ova teorema [2, 3], kao i jedne matematičke institucije – Gauss/Seidell-ov iterativni metod za rešavanje sistema nelinearnih jednačina [3]. U delu 4 je dat primer proračuna. Delovi 5 i 6 sadrže zaključke i korišćenu literaturu, respektivno.

2. TOKOVI SNAGA RADIJALNIH DISTRIBUTIVNIH MREŽA

Razmatra se radijalna mreža sa n trofaznih čvorova i $(n-1)$ -om trofaznom granom, $n \geq 2$. Model mreže je monofazan (pofazan), saglasan s pogonskom šemom mreže za simetričan režim direktnog redosleda. On se odnosi na režim faze a. Sa 0 je označena tačka (čvor) referentnog – nultog potencijala (zemlja). To je $(n-1)$ -vi čvor mreže.

Normalizovana pogonska šema mreže može da se konstituiše koristeći se isključivo monofaznim Γ segmentima prikazanim na slici 1. Svaki Γ segment se sastoji od redne grane s parametrom $\hat{z}_k$ i otočne grane s parametrom $\hat{y}_{ok}$. Jedan Γ segment je asociran jednoj sekciji voda ili jednom transformatoru. Sa K i k su označeni indeksi čvorova na početku (prvi čvor) i kraju razmatranog segmenta (drugi čvor), odnosno čvor bliži korenu i čvor dalji od korena radijalne mreže, respektivno. S obzirom na radijalnost mreže, prvi čvor K segmenta k ujedno je i drugi čvor jedinstvenog Γ segmenta prethodnika s kojeg se napaja razmatrani segment, ili je to koren mreže. Čvor K može da bude prvi čvor više drugih Γ segmenata koji se napajaju preko istog prethodnika. Čvor k je ujedno i prvi čvor jednog ili više Γ segmenata – sledbenika koji se napajaju preko razmatranog segmenta i/ili je u njemu priključen potrošač/generator s faznom

kompleksnom snagom $\hat{S}_k = P_k - jQ_k$, za definiciju: $\hat{S}_k = \hat{U}_k^* \hat{I}_k$. Moguća zavisnost snage potrošača od napona neće se isticati. Fazor fazne struje potrošača označen je sa $\hat{I}_{sk}$. Usled radialnosti mreže, svaki Γ segment i reprezentu njegovih rednih i otočnih parametara indeksirani su istim indeksom njegovog drugog čvora k .



Slika 1. Pogonska šema segmenta Γ_k trofazne distributivne mreže (a) i njegovo ekvivalentno kolo (b)

Kada je u pitanju Γ segment koji predstavlja sekciju voda, onda je sa $\hat{z}_k$ označena njena redna pogonska impedansa za simetričan režim direktnog redosleda. Kada je u pitanju transformator, onda je to njegova impedansa kratkog spoja (idealni transformatori iz ekvivalentnih šema transformatora eliminisani su primenom sistema relativnih vrednosti i θ transformacije [3]). Struja redne grane označena je sa $\hat{I}_k$. Sa $\hat{y}_{ok}$ je označen admitantni reprezent otko segmenta Γ_k . On predstavlja sumu admitantnog reprezenta otočnog parametra kraja sekcije ili transformatora kojem je segment Γ_k pridružen, otočnih parametara na počecima sekcija ili transformatora koji se napajaju preko razmatrane sekcije ili transformatora (ako ih ima) i reprezentata otočnih elemenata direktno priključenih u čvoru k . Sa $\hat{I}_{ok}$ je označena struja otko segmenta Γ_k .

Eksitaciju mreže, koja se unapred specificira, čine kompleksni napon korena i kompleksne snage potrošnje/proizvodnje svih ostalih čvorova. Za tako specificiranu eksitaciju, uz poznate parametre, režim mreže je određen ali nije poznat. On je osnovni predmet proračuna tokova snaga. Režim radialne distributivne mreže neće se promeniti ako se svi Γ segmenti sa slike 1a, koristeći se principom supstitucije, zamene ekvivalentnim Γ segmentima koji su prikazani na slici 1b.

Model razmatrane mreže se formuliše u sledeća tri koraka [6]:

- 1) Utvrđivanje struja čvorova koje su jednake strujama idealnih strujnih generatora s kojima su supstituisani potrošači/generatori i otočne admitanse čvorova: $\hat{J}_k$ (čvor k na slici 1); ovde su one, koristeći se relacijama $\hat{J}_k = \frac{\hat{S}_k}{\hat{U}_k^*} + \hat{y}_{ok} \hat{U}_k$, $k=2, 3, \dots, n$, iskazane u zavisnosti od kompleksnih snaga čvorova $\hat{S}_k$ i struja u otočnim granama Γ segmenata $\hat{y}_{ok}$, $k=2, 3, \dots, n$.
- 2) Utvrđivanje relacija bilansa struja u svim čvorovima kola (osim prvog – korena), koje su saglasne sa strujnim Kirchhoff-ovim zakonom:

$$\hat{I}_k = \hat{J}_k + \sum_{j \in \alpha_k} \hat{I}_j, k = n, n-1, \dots, 3, 2; \quad (2.1)$$

pri čemu je sa α_k označen skup indeksa svih Γ segmenata kojima je segment Γ_k prethodnik.

- 3) Utvrđivanje relacija prema naponskom Kirchhoff-ovom zakonu za konture koje odgovaraju Γ segmentima i sadrže jednu nezavisnu rednu granu:

$$\hat{U}_k = \hat{U}_K - \hat{z}_k \hat{I}_k, k = 2, 3, \dots, n; \quad (2.2)$$

Ako se struje idealnih strujnih generatora $\hat{J}_k$, $k=2, 3, \dots, n$ uvrste u relacije (2.1), dobija se sledeći matematički model razmatrane radialne distributivne mreže:

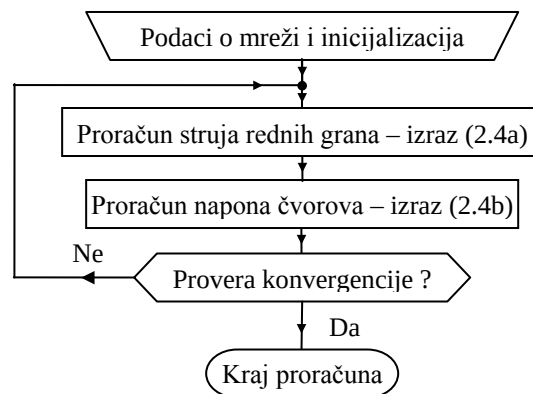
$$\hat{I}_k = \frac{\hat{S}_k}{\hat{U}_k^*} + \hat{y}_{ok} \hat{U}_k + \sum_{j \in \alpha_k} \hat{I}_j, k = n, n-1, \dots, 3, 2; \quad (2.3a)$$

$$\hat{U}_k = \hat{U}_K - \hat{z}_k \hat{I}_k, k = 2, 3, \dots, n. \quad (2.3b)$$

Za rešavanje ovog modela koristi se Gauss/Seidel-ov metod:

$$\hat{I}_k^{h+1} = \frac{\hat{S}_k}{\hat{U}_k^{*h}} + \hat{y}_{ok} \hat{U}_k^h + \sum_{j \in \alpha_k} \hat{I}_j^{h+1}, \quad k = n, n-1, \dots, 3, 2; \quad (2.4a)$$

$$\hat{U}_k^{h+1} = \hat{U}_K - \hat{z}_k \hat{I}_k^{h+1}, \quad k = 2, 3, \dots, n. \quad (2.4b)$$



Slika 2. Blok-dijagram algoritma za proračun tokova snaga radialnih distributivnih mreža

Procedura, koja je određena relacijama (2.4a), predstavlja proceduru sumiranja struja u okviru h -te iteracije postupka sumiranja struja i korekcija napona za proračun tokova snaga radialnih mreža, ("čišćenje unazad – sweep up, backward step"). Relacije (2.4) predstavljaju primenu naponskog Kirchhoff-ovog zakona na redne grane Γ segmenta razmatrane mreže, u tekućoj iteraciji rešavanja linearizovanog modela mreže. Ova procedura, koja je određena korišćenjem relacija (2.4b), predstavlja proceduru korekcija napona u okviru h -te iteracije postupka sumiranja struja i korekcija napona za proračun tokova snaga radialnih mreža ("čišćenje unapred – sweep down, forward step"). Procedura se završava u h -toj iteraciji, kada su zadovoljeni kriterijumi konvergencije:

$$|\Delta \theta_k^h| \leq \varepsilon_1 \wedge |\Delta U_k^h| \leq \varepsilon_2, k = 2, 3, \dots, n, \quad (2.5)$$

gde su $\Delta\theta_k^h$ i ΔU_k^h razlika uglova, odnosno modula kompleksnih napona čvora k u h -toji i $(h-1)$ iteraciji, respektivno. Opisani postupak dat je blok dijagramom prikazanim na slici 2.

3. TOKOVI SNAGA DISTRIBUTIVNIH MREŽA S PETLJAMA

Proračun simetričnih tokova snaga trofaznih mreža sa petljama se sastoji od postupka sumiranja struja i korekcija napona, kao i od *kompensacije za petlje*. On predstavlja kombinaciju postupaka prikazanih u referencama [4, 5]. Razmatra se mreža koja se sastoji od n trofaznih čvorova, $n \geq 2$ (nije ubrojan čvor referentnog potencijala 0 – zemlja), sa p petlji. Dakle radijalizovana (kompenzovana) mreža ima $n+p$ čvorova i $n-1+p$ trofaznih rednih grana. Model mreže je monofazan, saglasan s pogonskom šemom mreže za simetričan režim direktnog redosleda. Eksitaciju mreže, koja se unapred specificira, čine kompleksni napon korena i kompleksne snage potrošnje/proizvodnje svih ostalih čvorova. Time je režim mreže određen, ali nije poznat. On je osnovni predmet proračuna tokova snaga mreže s petljama.

Da bi mreža s petljama mogla da se tretira isto kao radijalna mreža, potrebno je da se otvore sve njene petlje. Izbor mesta gde se otvaraju petlje slobodan je [4, 5]. Otvaranjem svih petlji, mreža se svodi na radijalnu mrežu – radijalizovana mreža. Režim u toj mreži se razlikuje od režima mreže s petljama (originalne mreže). Da bi režim u radijalizovanoj mreži bio isti s režimom originalne mreže, potrebno je da se promene topološke strukture mreže, koje su se desile njenom radijalizacijom, kompenzuju odgovarajućim kompenzacionim strujama – supstituišu idealnim strujnim generatorima. Kompenzacione struje tih generatora se injektiraju u čvorove u kojima su petlje otvorene. Čvor izabran za otvaranje jedne petlje cepa se na dva čvora, tj. umesto jedinstvenog čvora, za svaku petlju se uvode po dva različita čvora: originalni čvor – čvor u kojem je otvorena petlja i novogenerisani čvor, koji je nastao cepanjem originalnog čvora. Pošto su režimi isti, umesto modelovanja i rešavanja mreže s petljama, može da se napiše i reši matematički model radijalizovane mreže sa insertovanim odgovarajućim idealnim strujnim generatorima u čvorovima otvaranja petlji. Model takve mreže mora da se proširi s relacijama koje opisuju jednakost napona parova čvorova nastalih otvaranjem petlji. Te relacije se zamenjuju odgovarajućim Thévenin-ovim relacijama.

Neka je sa $\mathbf{P}_{\text{pet}}$ označen skup parova indeksa, od kojih svaki par odgovara jednom originalnom i jednom novogenerisanom čvoru u kojima je otvorena jedna petlja. Suma snaga $\hat{S}_l$ i $\hat{S}_m$ para čvorova (l, m) , u kojima se otvara petlja, jednaka je snazi originalnog čvora l pre otvaranja petlje. Isto važi i za otočne admitanse originalnog čvora l i para čvorova (l, m) , $(l, m) \in \mathbf{P}_{\text{pet}}$. Analogno s modelom radijalnih mreža, model kompenzovane mreže može da se proširi novim nepoznatim varijablama – kompenzacionim strujama, kao i dodatnim relacijama kojima se iskazuje jednakost napona čvorova u kojima se otvaraju petlje. Time se dobija sledeći model (sistem jednačina):

$$\hat{I}_k = \frac{\hat{S}_k}{\hat{U}_k^*} + \hat{J}_{ck} + \hat{y}_{ok}\hat{U}_k + \sum_{j \in a_k} \hat{I}_j,$$

$$k = n+p, \dots, 3, 2; \quad (3.1a)$$

$$\hat{U}_k = \hat{U}_K - \hat{z}_k \hat{I}_k, \quad k=2, 3, \dots, n+p; \quad (3.1b)$$

$$\hat{U}_l = \hat{U}_m, \quad (l, m) \in \mathbf{P}_{\text{pet}}, \quad (3.1c)$$

pri čemu je sa $\hat{J}_{ck}$ označena kompenzaciona struja u čvoru k , $k=2, 3, \dots, n+p$. Model (3.1), koristeći se opštom Thévenin-ovom teoremom, može da se prikaže u sledećoj ekvivalentnoj formi:

$$\hat{I}_k = \frac{\hat{S}_k}{\hat{U}_k^*} + \hat{J}_{ck} + \hat{y}_{ok}\hat{U}_k + \sum_{j \in a_k} \hat{I}_j, \quad k = n+p, \dots, 3, 2; \quad (3.2a)$$

$$\hat{U}_k = \hat{U}_K - \hat{z}_k \hat{I}_k, \quad k = 2, 3, \dots, n+p; \quad (3.2b)$$

$$\hat{\mathbf{E}}_T - \hat{\mathbf{Z}}_T \hat{\mathbf{J}}_c = \mathbf{0}. \quad (3.2c)$$

Sa $\hat{\mathbf{E}}_T$ i $\hat{\mathbf{Z}}_T$ su označeni vektor Thévenin-ovih elektromotornih sila (dimenzija $p \times 1$) i Thévenin-ova matrica impedansi (dimenzija $p \times p$), viđeni iz svih parova čvorova otvaranja petlji; sa $\hat{\mathbf{J}}_c$ je označen vektor kompenzacionih struja (dimenzija $p \times 1$):

$$\hat{\mathbf{E}}_T = \begin{bmatrix} \vdots \\ \hat{E}_{lm} \\ \vdots \end{bmatrix}, \quad \hat{\mathbf{J}}_c = \begin{bmatrix} \vdots \\ \hat{J}_{lm} \\ \vdots \end{bmatrix}, \quad (l, m) \in \mathbf{P}_{\text{pet}} \quad (3.3)$$

Relacije (3.2c) ekvivalentne su relacijama (3.1c). Sada može da se utvrdi sledeći proračun od kojeg se sastoji jedna iteracija Gauss/Seidel-ovog metoda za proračun razmatrane mreže sa p petlji:

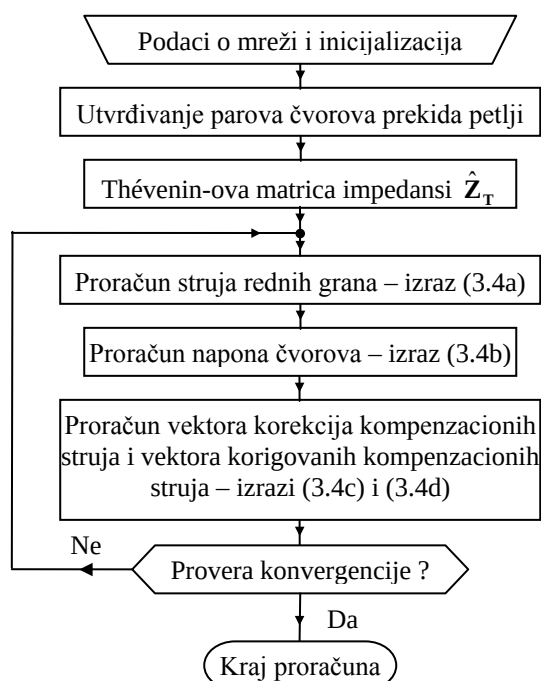
$$\hat{I}_k^{h+1} = \frac{\hat{S}_k}{\hat{U}_k^{h*}} + \hat{J}_{ck}^h + \hat{y}_{ok}\hat{U}_k^h + \sum_{j \in a_k} \hat{I}_j^h, \quad k = n+p, \dots, 3, 2; \quad (3.4a)$$

$$\hat{U}_k^{h+1} = \hat{U}_K^{h+1} - \hat{z}_k \hat{I}_k^{h+1}, \quad k=2, 3, \dots, n+p; \quad (3.4b)$$

$$\Delta \hat{\mathbf{J}}_c^h = \hat{\mathbf{Y}}_N (\hat{\mathbf{U}}_1^{h+1} - \hat{\mathbf{U}}_2^{h+1}), \quad \left[\hat{\mathbf{Y}}_N = (\hat{\mathbf{Z}}_T)^{-1} \right] \quad (3.4c)$$

$$\hat{\mathbf{J}}_c^{h+1} = \hat{\mathbf{J}}_c^h + \Delta \hat{\mathbf{J}}_c^h \quad (3.4d)$$

Sa $\hat{\mathbf{U}}_1^{h+1}$ i $\hat{\mathbf{U}}_2^{h+1}$ označeni su vektori $(h+1)$ -vih aproksimacija napona originalnih i novogenerisanih čvorova u kojima se otvaraju petlje, respektivno, oba dimenzija $p \times 1$. Elementi na istim pozicijama ta dva vektora odgovaraju paru čvorova otvaranja jedne petlje (originalni i novogenerisani). Sa $\hat{\mathbf{Y}}_N$ je označena Norton-ova matrica admitansi, odnosno inverzna Thévenin-ova matrica impedansi $\hat{\mathbf{Z}}_T$, s dimenzijama broja petlji ($p \times p$). Sa $\Delta \hat{\mathbf{J}}_c^h$ je označen vektor korekcija h -tih aproksimacija kompenzacionih struja $\hat{\mathbf{J}}_c^h$, a sa $\hat{\mathbf{J}}_c^{h+1}$ vektor korigovanih – $(h+1)$ -vih aproksimacija tih struja. Njihove dimenzije su $p \times 1$. Procedura se završava u h -toj iteraciji, kada su zadovoljeni kriterijumi konvergencije – relacija (2.5). Opisani postupak dat je blok dijagramom prikazanim na slici 3.



Slika 3. Blok-dijagram algoritma za proračun tokova snaga distributivnih mreža sa petljama

4. PRIMER PRORAČUNA

U ovom delu data su dva primera proračuna.

4.1. Povećanje broja čvorova

U ovom primeru razmatraju se tri radijalne 20kV distributivne mreže. One se sastoje od: sto jednog, petsto jednog i hiljadu jednog čvora i to jedan fider sa sto čvorova, pet fidera sa po sto čvorova i deset fidera sa po sto čvorova, respektivno. Parametri svih sekcija su isti i iznose: $\hat{z} = (0,41 + j0,36)\Omega$ i $\hat{y} = (0 + j1,5 \times 10^{-7})S$. Trofazne kompleksne snage potrošnje svih čvorova su jednake: $\hat{S} = 3 \times (90kW + j45kVAr)$. Rezultati proračuna dati su u tabeli 1. Na osnovu rezultata iz tabele zaključujemo da sa porastom broja čvorova i vreme postupka proračuna tokova snaga linearno raste.

Tabela 1. Vreme i broj iteracija proračuna tokova snaga

Broj čvorova	101	501	1001
Vreme[s] / broj iteracija	$3,0204 \times 10^{-5}$ / 5	$1,5468 \times 10^{-4}$ / 5	$3,1198 \times 10^{-4}$ / 5

4.2. Mreža sa petljama

U ovom primeru razmatra se 20kV distributivna mreža koja se sastoji od dvesta jednog čvora, koji se nalaze na dva fidera po sto čvorova. Parametri sekcija sa parnim rednim brojem jednaki su parametrima sekcija iz prethodnog primera, dok su parametri sekcija sa neparnim rednim brojem prepolovljeni. Da bi se simulirala mreža sa petljom između parova čvorova dodaju se kratkospojnici. Najpre se kratkospojnikom povezuju čvorovi 10 i 11. Zatim se broj petlji u mreži povećava, i to tako što se dodaju novi kratkospojnici između čvorova čiji je redni broj veći za dvadeset u odnosu na redne brojeve čvorova koji su prethodno vezani kratkospojnikom (30 i 31 sve do 190 i 191). Trofazne kompleksne snage potrošnje svih čvorova su jednake: $\hat{S} = 3 \times (90kW + j45kVAr)$. Rezultati

proračuna dati su u tabeli 1. Na osnovu rezultata iz tabele zaključujemo da sa porastom broja petlji u mreži raste broj iteracija odnosno vreme proračuna tokova snaga.

Tabela 2. Vreme i broj iteracija proračuna tokova snaga

Broj petlji	Vreme[s] / broj iteracija
0	$1,2656 \times 10^{-4}$ / 5
1	$1,5000 \times 10^{-4}$ / 7
2	$1,6563 \times 10^{-4}$ / 8
3	$1,7031 \times 10^{-4}$ / 8
4	$1,7500 \times 10^{-4}$ / 9
5	$1,7813 \times 10^{-4}$ / 9
6	$1,8906 \times 10^{-4}$ / 10
7	$1,9063 \times 10^{-4}$ / 10
8	$1,9063 \times 10^{-4}$ / 10
9	$1,9219 \times 10^{-4}$ / 10
10	$1,9219 \times 10^{-4}$ / 10

5. ZAKLJUČAK

Ovim radom je pokazano da proračuni tokova snaga radijalnih distributivnih mreža i distributivnih mreža s petljama, koji su zasnovani na postupku sumiranja struja i korekcija napona, predstavljaju vrlo sofisticiranu sintezu primene sledeće tri električne institucije – Kirchhoff-ovi zakoni, Thévenin/Norton-ova teorema i princip supstitucije, kao i jedne matematičke (numeričke) institucije – Gauss/Seidel-ov metod za rešavanje sistema nelinearnih simultanih jednačina. Pored toga, problem tokova snaga distributivnih mreža s petljama postavljen je i rešen tako da je izvršena sinteza dobrih osobina rešenja predloženih u referencama [4] i [5].

6. LITERATURA

- [1] W. F. Tinney, C. E. Hart: Power Flow Solution by Newton's Method, *IEEE Trans. on PAS*, Vol. 86, No. 11, Nov. 1967, pp. 1449-1460.
- [2] B. Popović: *Osnovi elektrotehnike 1 i 2*, Nauka, Beograd, 1998.
- [3] V. C. Strezoski: *Analiza elektroenergetskih sistema*, skripta, Fakultet tehničkih nauka Novi Sad, 2011.
- [4] D. Shirmohammadi, H. W. Hong, A. Semlyen, G. X. Luo: A Compensation-Based Power Flow Method for Weekly Meshed Distribution and Transmission Network; *IEEE Trans. on PS*, Vol.3, No.2, May 1988, pp. 753-762.
- [5] G. X. Luo, A. Semlyen: Efficient Load Flow for Large Weekly Meshed Distribution and Transmission Network; *IEEE Trans. on PS*, Vol.5, No.4, November 1990, pp. 1309-1316
- [6] R. V. Strezoski: *Teorijska zasnovanost proračuna tokova snaga distributivnih mreža*, master rad, Fakultet tehničkih nauka Novi Sad, 2013.

Kratka biografija:



Kristian Haška rođen je u Novom Sadu 1992. godine. Živi u Kulpinu. Diplomski rad je odbranio 2015. godine na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Elektroenergetski sistemi.

**VIŠEKRITERIJUMSKA PODELA MODELA PODATAKA DISTRIBUTIVNE MREŽE
MULTI-CRITERIA PARTITIONING OF DISTRIBUTION NETWORK DATA MODEL**

Aleksandar Svorcan, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Predmet ovog rada jeste razvoj mehanizma za višekriterijumsku podelu modela podataka distributivne mreže. Implementirani mehanizam za podelu modela podataka podržava kako podelu celokupnog modela podataka, tako i inkrementalne izmene modela. Rešenje koje je rezultat ovog rada je nezavisno od modela podataka, i omogućava konfiguraciju kriterijuma za podelu modela podataka. U okviru ovog rada su analizirane performanse implementiranog rešenja, i predstavljeni su dobijeni rezultati.

Abstract – The subject of this thesis is implementation of multi-criteria partitioning of distribution network data model. Implemented solution supports partitioning of complete data model, as well as partitioning of incremental model updates. Criteria for partitioning data model are configurable, which makes mechanism for partitioning independent of partitioned data model. Performance of implemented solution is analyzed, and results are presented.

Ključne reči: *Common Information Model, teorija grafova, distributivna elektroenergetska mreža, Smart Grid*

1. UVOD

Pametni gradovi predstavljaju aktuelnu inicijativu primene novih rešenja u oblastima informacionih i komunikacionih tehnologija u svrhu optimizacije korišćenja gradskih resursa. Važan deo ove inicijative predstavljaju UMS (*Utility Management System*) sistemi. Ovi sistemi predstavljaju klasu sistema koji se koriste u oblasti distribucije vode, gasa ili električne energije i u njih spadaju WQEMS (*Water Quality and Energy Managment System*) u sektoru distribucije vode i EMS (*Energy Management System*) i DMS (*Distribution Management System*) u sektoru distribucije električne energije, čija je namena da korisnicima omoguće nadzor i upravljanje distributivnom mrežom. Da bi se takvim sistemom moglo upravljati i nad njim vršiti različite proračune i simulacije potrebno je sve njegove elemente i njihove karakteristike ugraditi u model podataka distributivne mreže. Model podataka čiji se red veličine broja entiteta meri u milionima je nepraktičan za razmenu između komponenti jednog UMS sistema, pa ga je poželjno po određenim kriterijumima podeliti u nezavisne celine.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Darko Čapko, docent.

Kao predmet ovog rada implementiran je mehanizam višekriterijumske podele modela podataka distributivne mreže u blokove. Rešenje je implementirano na primeru elektrodistributivne mreže, uz oslonac na CIM (*Common Information Model*), DAF (*Data Access Facility*) i GDA (*Generic Data Access*) standarde. Prilikom implementacije rešenja koje je predstavljeno u ovom radu klase modela podataka i veze između njih su apstrahovane grafom i primenjen je algoritam pretrage grafa metodom „prvo u širinu“.

2. PAMETNI GRADOVI

Pametni gradovi (*Smart Cities*) predstavljaju aktuelnu inicijativu u sferi urbanističkog razvoja. Oni predstavljaju vezu između stvaranja ekološke budućnosti i daljeg ekonomskog razvoja i rasta populacije gradova. Osnovna karakteristika pametnih gradova je strateško korišćenje novih visokotehnoloških rešenja u oblasti informacionih i komunikacionih tehnologija primenjenih na različite aspekte njihove infrastrukture [1] u cilju optimizacije korišćenja gradskih resursa - distribucije električne energije, vode, gasa, transportnih sredstava i ostalih javnih servisa. UMS sistemi su jedna od okosnica *Smart City* inicijative, i u nastavku će biti detaljnije objašnjen pojam pametne elektronenergetske mreže, jer je upravo model *Smart Grid*-a korišćen kao primer pri implementaciji rešenja koje je predmet ovog rada.

Smart Grid je napredna električna mreža koja može efikasno da integriše potrebe i zahteve svih entiteta priključenih u jedinstven elektroenergetski i korisnički sistem [2]:

- tradicionalnih proizvođača i novih tipova proizvođača električne energije (individualni proizvođači na bazi alternativnih izvora energije),
- tradicionalnih distributera i drugih komunalnih službi,
- svih tipova tradicionalnih potrošača i novih tipova potrošača (električni automobili i dr.),
- proširenih zahteva za merenjem i upravljanjem potrošnjom (unapređenja tarifnih sistema, havarijska rasterećenja i dr.),
- naprednih zahteva raznih aplikacija (pametne zgrade i dr.).

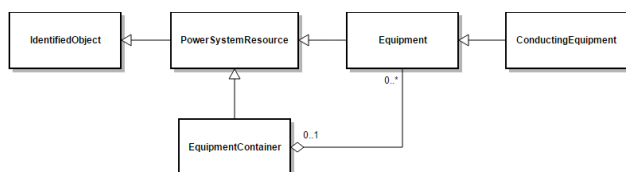
**3. MODEL PODATAKA
ELEKTROENERGETSKE MREŽE**

CIM [3] je apstraktni UML model kojim se predstavljaju svi bitni elementi elektroenergetskog sistema. U njemu su

sadržani opisi svih klasa i njihovih karakteristika, kao i međusobne veze između različitih klasa.

Postojanje CIM modela olakšava integraciju sistema za upravljanje prenosom električne energije. Pre nastanka CIM-a svaka kompanija je koristila svoj nestandardni model podataka i samim tim je razmena podataka između sistema različitih elektroenergetskih kompanija bila otežana. Zbog ovog problema EPRI (*Electronic Power Research Institute*) je kreirao grupu standarda IEC 61970, čiji se deo IEC 61970-301 odnosi na modeliranje elektroenergetskih sistema [3].

Klase CIM modela su grupisane u pakete. Svaki paket predstavlja skup logički povezanih klasa, koje su zajedno sa vezama između njih, grafički prikazane na dijagramima klasa. Core paket sadrži osnovne klase potrebne za funkcionisanje EMS sistema, kao što su: *IdentifiedObject*, *PowerSystemResource*, *EquipmentContainer*, *Terminal*, *ConductingEquipment*... Core paket nije zavistan od ostalih paketa, ali oni većinom zavise od njega. Primera radi, sva oprema u električnoj mreži nasleđuje klasu *PowerSystemResource*, a klase naslednice klase *EquipmentContainer* se koriste za grupisanje opreme. Na slici 1 su prikazane osnovne klase CIM modela i veze između njih.



Slika 1. Prikaz strukture osnovnih klasa CIM modela

U cilju standardizacije pristupa podacima u UMS sistemima OMG (*Object Management Group*) grupa je kreirala UMS DAF (*Data Access Facility*) specifikaciju [4]. DAF specifikacija definiše interfejs za postavljanje upita nad podacima UMS sistema i čitanje njihovih rezultata.

IEC 61970-403: *Generic Data Access (GDA)* [5] je standard zasnovan na OMG UMS DAF specifikaciji koji omogućava generički pristup podacima baziranim na CIM-u. GDA dozvoljava klijentu pristup podacima koje održava neka druga aplikacija ili sistem bez potrebe za poznavanjem logičke šeme modela podataka. GDA je namenjen za asinhronu komunikaciju klijenta i servera i primenjuje se u aplikacijama koje zahtevaju pristup podacima u realnom vremenu, kao što je na primer pristup SCADA podacima. GDA servis podeljen je u tri kategorije: pristup za čitanje podataka, pristup za izmenu podataka i promena reakcije na događaje. GDA modul za pristup podacima radi njihove izmene proširuje modul GDA pristupa za čitanje podataka sledećim operacijama:

- modifikacija vrednosti skupa atributa jednog objekta,
- modifikacija vrednosti skupa atributa više objekata jednim upitom,
- kontrola životnog ciklusa objekata.

Izmena podataka omogućena je metodom *apply_updates()* iz *ResourceUpdateService* interfejsa. Ova operacija ima parametar tipa *DifferenceModel* koji

sadrži razlike stanja modela podataka pre i posle uspešnog izvršavanja operacije *apply_updates()*.

4. TEORIJA GRAFOVA

Teorija grafova [6] je oblast matematike, veoma zastupljena i u informatici, a bavi se istraživanjem osobina grafova. Neformalno govoreći, grafovi se sastoje od tačaka, odnosno čvorova (vrhova), i linija između njih (grana). Stoga je veoma česta primena grafova za reprezentaciju modela i struktura podataka.

Graf G je uređeni par (V, E) skupova, takvih da je $E \subseteq [V]^2$, tj. da su elementi skupa E dvočlani podskup skupa V . Elementi skupa V se nazivaju *čvorovima* grafa, a elementi skupa E se nazivaju *granama* grafa. Graf je neusmeren (neorijentisan) ako je grana između čvorova u i v istovremeno i grana između čvorova v i u .

Pojam stabla predstavlja jedan od najvažnijih pojmova u teoriji grafova. Stablo predstavlja graf u kome su svaka dva čvora povezana tačno jednim nizom grana. Čvor koji ima samo jednog suseda u stablu se naziva list ili viseći čvor. Čvor u koji u orijentisanom grafu ne ulazi nijedna grana, a iz njega izlazi jedna ili više grana naziva se korenom stabla. U neorijentisanom grafu se za koren stabla uzima čvor koji je po nečemu poseban. Stabla sa korenskim čvorom nazivamo korenska stabla.

Kod korenskih stabala su izuzetno važni pojmovi roditelj i deca. Čvor u je roditelj čvoru v ako je grana uv orijentisana od čvora u ka čvoru v . U tom slučaju za čvor v kažemo da je dete čvora u . Preci čvora v su svi čvorovi koji leže na putu od korena stabla do čvora v , osim čvora v . Potomci čvora v su svi čvorovi kojima je čvor v predak.

4.1. Pretraga grafa algoritmom prvo u širinu (BFS)

Jedan od osnovnih problema u teoriji grafova predstavlja pronalazak načina za sistematski obilazak svih čvorova i grana grafa. Ovaj problem leži u osnovi bilo kakvog rada sa grafovima jer je potreban i za osnovne operacije kao što su npr. ispis, kopiranje i bilo kakva konverzija grafa.

Algoritmom „prvo u širinu“ [7] se čvorovi grafa posećuju sistematski, nivo po nivo. Obilazak se započinje od čvora nultog nivoa - korena stabla ili nekog proizvoljno odabranog čvora v . Zatim se obilaze svi susedi čvora v , odnosno čvorovi prvog nivoa. Nakon što su svi čvorovi prvog nivoa obišteni, obilaze se susedi prvog suseda čvora v , osim onih koji su već posećeni. Zatim se obilaze svi susedi drugog suseda čvora v , takođe izuzimajući one koji su već posećeni. Ovaj postupak se nastavlja sve dok se ne obišu svi čvorovi drugog nivoa, i zatim se prelazi na naredni nivo. Kraj pretrage predstavlja trenutak kad se poseti poslednji čvor poslednjeg nivoa.

5. BLOKOVSKA ORGANIZACIJA MODELA PODATAKA

Model podataka UMS sistema sadrži karakteristike miliona različitih entiteta. Zbog velikog broja entiteta javlja se problem sa brzinom pristupa pojedinačnim entitetima, razmenom podataka između različitih komponenti sistema i obradom razmenjenih podataka.

Kao rešenje ovog problema odabrano je deljenje modela podataka u nezavisne celine – blokove. Prednosti blokovske organizacije modela podataka ogledaju se u tome što je na taj način olakšan pristup podacima, jer svaka od komponenti koja obrađuju podatke ne mora da dobavlja ceo model, već samo njegov deo od interesa. Na ovaj način se rasterećuje komunikaciona infrastruktura, kao i sama implementacija komponente jer sa ovakvim pristupom problemu svaka od komponenti ne mora da implementira logiku za pretragu modela za podacima od interesa, već je dovoljno samo da iterira po podacima u traženim blokovima.

U cilju veće granularnosti podele modela, u ovom rešenju se svaki blok posmatra kao par susednih blokova. Na ovaj način je moguće posebno označiti elemente koji su bitni za prikaz na grafičkom korisničkom interfejsu, i oni će biti smešteni u blok sa parnim indeksom para blokova u koji bi inače bili raspoređeni.

6. IMPLEMENTIRANO REŠENJE PODELE MODELA PODATAKA U BLOKOVE

Mehanizam za deljenje modela podataka u blokove koji je predmet ovog rada podrazumeva obradu svakog pristigle izmene modela (*DifferenceModel*-a) prilikom poziva metode *apply_updates()* GDA interfejsa, i njegovo ažuriranje sa informacijama o pripadnosti entiteta odgovarajućim blokovima.

Tok transakcije operacije izmene modela sadrži sledeće korake:

- 1) pravljenje parcijalne kopije modela na koju će biti primenjen neobrađen *DifferenceModel*, prilikom čega se svi novododati entiteti dodaju u prvi blok,
- 2) primenjivanje neobrađenog *DifferenceModel*-a na kopiju modela,
- 3) validacija ispravnosti izmene modela,
- 4) pronalaženje ažuriranih parova blokova,
- 5) pronalaženje entiteta koji predstavljaju posebne parove blokove u ažuriranim blokovima,
- 6) kreiranje novih parova blokova,
- 7) hijerarhijsko smeštanje entiteta iz ažuriranih blokova u odgovarajuće blokove, počev od entiteta koji predstavljaju posebne blokove,
- 8) ažuriranje *DifferenceModel*-a informacijama o blokovim,
- 9) primena ažuriranog *DifferenceModel*-a na originalni model podataka.

Prilikom implementacije predstavljenog rešenja, jedan od činilaca je bila i nezavisnost mehanizma za deljenje modela podataka od samog modela podataka. Stoga je odlučeno da se kriterijumi za podelu modela podataka učitavaju pri startovanju aplikacije iz konfiguracione XML datoteke.

Pravila za podelu modela podataka se definišu na nivou svake klase entiteta, i predstavljeni su interfejsom *ITypeRule* koji definiše sledeće aspekte svakog pravila:

- tip entiteta za koji je pravilo definisano,
- lista pravila po kojima posmatrani entitet predstavlja poseban par blokova,

- lista pravila po kojima posmatrani entitet pripada grupi entiteta koja predstavlja poseban par blokova,
- lista atributa tipa reference, koji predstavljaju referencirane entitete koji se smeštaju u isti par blokova kao i entiteti posmatranog tipa,
- lista pravila koja definišu kada je posmatrani entitet bitan za prikaz na grafičkom korisničkom interfejsu, odnosno kada će biti smešten u parni blok u okviru para blokova kojem pripada.

6.1. Algoritam za smeštanje entiteta u blokove

Prilikom implementacije logike za smeštanje elemenata u odgovarajuće blokove model podataka je apstrahovan grafom. Kao čvorovi grafa su posmatrani entiteti modela, dok su grane grafa bile reference između entiteta. Ovo je omogućilo rešavanje ovog problema primenom algoritma za pretragu grafa „prvo u širinu“. Entiteti su posmatrani kao korensko stablo, gde je koren stabla entitet koji ispunjava kriterijume da predstavlja poseban par blokova.

7. REZULTATI

Rešenje koje je implementirano kao predmet ovog rada je testirano u odnosu na postojeće rešenje ovog problema. Postojeće rešenje je koristilo drugačiji pristup za rešavanje ovog problema – *DifferenceModel* nije primenjivan na kopiju modela podataka, već su elementi smešteni u blokove tako što su se pravile i poredile mape reference iz modela podataka iz *DifferenceModel*-a, i svi kriterijumi smeštanja su bili definisani direktno u kodu rešenja.

Prednosti implementiranog rešenja su sledeće:

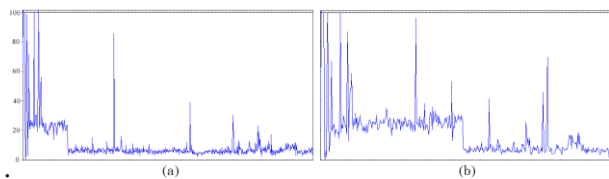
- nezavisnost mehanizma za deljenje modela podataka od samog modela podataka zbog postojanja konfiguracione datoteke
- jednostavniji i pouzdaniji kod jer se ispravnost *DifferenceModel*-a validira pre određivanja blokova, te nije potrebno obrađivati mnogobrojne moguće greške, koje čak nije ni moguće predvideti,
- veća granularnost bitnosti elemenata za prikaz na korisničkom interfejsu,
- mehanizam za ponovno deljenje modela podataka u slučaju izmene kriterijuma za deljenje modela podataka.

Mana implementiranog rešenja u odnosu na postojeće je to što se prilikom dodavanja manjeg broja elemenata u neki blok vrši ponovno smeštanje svih entiteta u tom bloku jer se zbog visokog stepena slobode u definisanju konfiguracije postoji mogućnost izmene bitnosti hijerarhije bitnosti entiteta za prikaz u tom bloku.

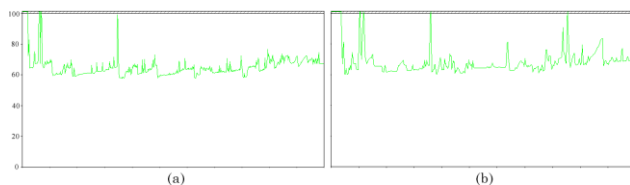
Testiranje je vršeno primenom velikog broja *DifferenceModel*-a na referentni model podataka elektroenergetske mreže korišćenjem jednog, a zatim drugog rešenja. Uporedni rezultati testiranja opterećenja procesora i radne memorije su prikazani na slikama 2 (opterećenje procesora) i 3 (opterećenje radne memorije).

Ukupno vreme izvršavanja već postojećeg rešenja bilo je 2 časa i 44 sekunde, dok je rešenje koje je predmet

ovog rada isti zadatak izvršilo za 1 čas, 16 minuta i 9 sekundi.



Slika 2. Grafik opterećenja procesora: (a) već postojećeg rešenja, (b) rešenja koje je predmet ovog rada.



Slika 3. Grafik opterećenja radne memorije: (a) već postojećeg rešenja, (b) rešenja koje je predmet ovog rada.

Dobijeno poboljšanje u vremenu izvršavanja aplikacije je posledica izmenjenog pristupa rešavanju problema, odnosno određivanja blokova nad već primenjenom izmenom na model podataka. Na ovaj način se izbegava pristupanje velikom broju mapiranja koje reprezentuju veze između *DifferenceModel*-a i modela podataka kao u referentnom rešenju. Porast u opterećenju radne memorije je takođe posledica ovakvog pristupa, jer se u implementiranom rešenju kreira dodatna kopija izmenjenog dela modela nad kojom se vrši deljenje modela podataka u blokove.

Logika u implementiranom rešenju je podeljena u skladu sa paradigmom objektno-orijentisanog programiranja, na više klasa sa jasno definisanim zadacima. Ovakav pristup značajno olakšava razumevanje i testiranje koda, ali istovremeno i prouzrokuje kreiranje i brisanje velikog broja objekata male veličine i kratkog životnog ciklusa, posledica čega je povećano opterećenje procesora.

5. ZAKLJUČAK

UMS sistemi, kao značajan deo *Smart City* inicijative, omogućuju optimalno korišćenje gradskih resursa - distribucije električne energije, vode, gasa, transportnih sredstava i ostalih komunalnih servisa. Ovo se postiže nadzorom i upravljanjem distributivnom mrežom resursa u realnom vremenu, kao i omogućavanjem različitih planerskih i simulacionih funkcionalnosti.

Model podataka jednog UMS sistema mora da sadrži karakteristike svih entiteta kojima se nadzire i upravlja, pa se samim tim red veličine broja entiteta u ovakvom modelu podataka meri milionima. Zbog velikog broja entiteta javlja se problem sa performansama u pristupu i razmeni podataka, kao i nepotrebnoj obradi podataka koji nisu od interesa.

Jedno od mogućih rešenja ovog problema je blokovska organizacija modela podataka, što omogućava brži pristup podacima, kao i obradu i razmenu samo blokova od interesa umesto celokupnog modela podataka. U ovom radu je implementirano jedno rešenje konfigurabilne višekriterijumske podele modela podataka u blokove, uz

podršku za inkrementalne izmene modela. Kao referentni model za implementaciju rešenja iskorišćen je model podataka *Smart Grid* sistema, uz oslonac na CIM, DAF i GDA standarde. Performanse implementiranog rešenja su testirane sa različitim inkrementalnim izmenama modela podataka, i merene su performanse koje su poređene sa performansama već postojećeg rešenja. Rešenje koje je implementirano kao predmet ovog rada je brže od referentnog rešenja, ali istovremeno i više opterećuje procesor i radnu memoriju računara od referentnog rešenja.

Glavni pravac za nastavak razvoja implementiranog rešenja je dalja detaljna analiza i optimizacija performansi, kao i analiza različitih konfiguracija kriterijuma za smeštanje podataka u blokove u cilju pronalaženja optimalne za rad ostatka sistema. Osim toga, dalji razvoj se može kretati u pravcu implementacije dodatnih funkcionalnosti automatskog generisanja minimalnog skupa entiteta bitnog za prikaz na grafičkom korisničkom interfejsu.

6. LITERATURA

- [1] S. Musa. *Smart City Roadmap*, University of Maryland University College, 2016
- [2] S. Borlase. *Smart Grids: Infrastructure, Technology, and Solutions (Electric Power and Energy Engineering)*, 1st Edition, CRC Press, 2012, ISBN 978-1439829059
- [3] IEC 61970-301 (2009-04) Ed. 2.0 *Energy management system application program interface (EMS-API)*, 2009, ISBN 978-2-88910-593-9
- [4] *Utility Management System (UMS) Data Access Facility Specification, Version 2.0.1*, Object Management Group, 2005, <http://www.omg.org/spec/UMS/>, datum pristupa: 1. oktobar 2016.
- [5] IEC 61970-403 (2008-06) Ed 1.0 *Energy management system application program interface (EMS-API) – Part 403: Generic data access*, 2008, ISBN 2-8318-9851-X
- [6] R. Diestel. *Graph Theory*, 4th Edition, Springer, 2012, ISBN 978-3-642-14278-9
- [7] S. Skiena. *The Algorithm Design Manual*, 2nd Edition, Springer, 2008, ISBN 978-1-84800-069-8

Kratka biografija:



Aleksandar Svorcan rođen je 1992. godine u Novom Sadu gde i dalje živi. Školske 2010/2011 godine upisuje Fakultet tehničkih nauka, smer Računarstvo i automatika. U julu 2015. godine brani diplomski rad. U oktobru 2015. godine upisuje master akademske studije na Fakultetu tehničkih nauka, smer Primenjeno softversko inženjerstvo

SISTEM ZA PERZISTENCIJU PODATAKA U PROGRAMSKOM JEZIKU CLOJURE

DATA PERSISTENCE SYSTEM IN CLOJURE PROGRAMMING LANGUAGE

Novak Boškov, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu je implementiran sistem za perzistenciju podataka u programskom jeziku Clojure. Sistem je implementiran kao interni DSL (eng. Domain Specific Language) koristeći načela i praksu samog programskog jezika kao i funkcionalne paradigme programiranja. Jezik je nezavisan od konkretnog tipa ili implementacije baze podataka u koju se podaci smeštaju.

Abstract – This paper presents implementation of a data persistence system in Clojure programming language. System is implemented as a internal DSL (Domain Specific Language) using principals and practice of programming language itself as well as functional programming paradigm. Language is relevant regardless of database type or implementation.

Ključne reči: DSL, Clojure, funkcionalno programiranje, baze podataka, perzistencija

1. UVOD

Ovaj rad se bavi konstrukcijom DSL-a za predstavljanje podataka kao i implementacijom sistema za njihovo čuvanje u različitim bazama podataka. DSL je skraćeno za *Domain Specific Languages* [1] tj. jezike specifične za domen.

Sistem za perzistenciju kojim se bavi ovaj rad (u daljem tekstu *Larva*) je baziran na konceptima Clojure programskog jezika kao i njegovog pogleda na funkcionalno programiranje. Clojure programski jezik spada u porodicu Lisp jezika. Lisp (ili LISP) je porodica programskih jezika originalno specificirana 1958. godine od strane Džona Mekartija (John McCarthy) i grupe za veštačku inteligenciju sa Tehnološkog instituta Masačusets (MIT) [2]. Na pravcu svog razvoja Lisp prolazi kroz različite faze pojednostavljivanja da bi na kraju njegova baza bila svedena na šemu za predstavljanje parcijalno rekurzivnih funkcija, jednu vrstu simboličkih izraza koji se od tada pa nadalje nazivaju *S-expressions*, a odgovarajuće funkcije *S-functions*. Čitav Lisp sistem biva sveden na izvestan broj matematičkih ideja i notacija nad kojima se gradi kompletan sistem za programiranje u kome univerzalna *S-funkcija apply* ima teorijsku ulogu univerzalne Turingove mašine dok sa praktičnog stanovišta predstavlja interpreter. Lisp ima praktično nesaglediv uticaj na sve današnje programske jezike (posebno one dinamičke). Neki od koncepata koje Lisp uvodi od svog nastanka a koji i danas predstavljaju manje-više napredne programske konstrukcije su: parcijalne funkcije, propozicioni izrazi i predikati,

kondicionalni izrazi, rekurzivne definicije funkcija i dr. Jedan od koncepata koji su danas veoma aktuelni u programerskoj zajednici su i „lambda funkcije“. U svom originalnom radu iz 1960. godine Mekarti jasno ukazuje na ovaj problem kada govori o funkcijama i formama. Mekarti uviđa da se u matematički izraz „funkcija“ često uzima nedovoljno precizno te da mu je za Lisp neophodno razlikovanje *S-izraza* koji predstavljaju funkcije od ostalih. Rešenje za ovaj problem nalazi u lambda-kalkulusu Alonza Črča [3] od koga odstupa u minimalnoj meri.

Mekarti u svom radu daje pet funkcija koje su potrebne za implementaciju kompletnog Lisp programskog okruženja. Te funkcije su: *atom*, *eq* (jednakost dva simbola), *car* (vraća glavu liste simbola), *cdr* (vraća rep liste simbola) i *cons* (spajanje dva simbolička izraza). U njegovom radu je dat i dokaz da su ove funkcije zajedno sa univerzalnom funkcijom *apply* minimalan skup potrebnih funkcija za konstrukciju Lisp-a.

Mekarti takođe uvodi i koncept *garbage collector-a*. Ovaj koncept dolazi na prirodan način kao posledica predstave *S-izraza* u memoriji. Svaki ovakav izraz se predstavlja kao jedan term koji je sačinjen od samog sadržaja i mesta koje sadrži adresu sledećeg term-a. Dakle, term je u uporebi samo ako se do njega može doći ulančavanjem operacija *car* i *cdr* iz nekog drugog korišćenog registra. Ako ovo nije slučaj onda taj term (registar) više nije od interesa i treba ga prepustiti *garbage collectoru* kako bi se vratio u listu slobodnih registara.

Zbog Mekartijevog velikog doprinosa programiranju, Pol Grejm (*Paul Graham*) u svom članku [4] kaže da je Mekarti učinio za programiranje ono što je Euklid za geometriju.

Clojure programski jezik, kreiran od strane Riča Hikija (Rich Hickey) 2007. godine kao dijalekat Lisp-a preuzima praktično sve osnovne ideje. Hiki svoj jezik dizajnira tako da bude gostujući (*hosted language*) i uvodi perzistentne strukture podataka koji mu pomažu u uspostavljanju funkcionalne paradigme. Kao posledicu ovih koncepata Clojure ima odlučnu podršku za konkurentno i paralelno programiranje.

1.1. Motivacija

Osnovna motivacija ovog rada je doprinos trenutnom stanju u Clojure ekosistemu. Svo bogatstvo Lisp-a i funkcionalnog programiranja je potpuno dostupno unutar standardnog ekosistema u kome programer provodi vreme. Na raspolaganju je skoro sve, od razvojnih okruženja do Web servera, biblioteka za programiranje dronova, okruženja za rad sa neuronskim mrežama, *rule engine-a*, sistema za ispitivanje distribuiranih sistema, okruženja za statističko računanje. Pored svega toga

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Igor Dejanović.

Clojure ima i probleme koji su najvećim delom vezani za nedostatak nekih manje-više klasičnih koncepata na koje su programeri danas navikli. To sve utiče na brzinu započinjanja projekata u praksi. Vreme koje je potrebno za uspostavljanje jedne standardne npr. *Web* aplikacije je duže nego u *Ruby* ili *Python* ekosistemima. Ako se programer odluči da koristi neki već gotov skup biblioteka za rad sa *Web-om* onda su problemi poput postavljanja *Web* servera, deskripcije *REST* servisa ili komunikacija *backend-a* i *frontend-a* rešeni. Vreme potrebno za slične zadatke je slično kao u navedenim okruženjima. Međutim, ako ima potrebu za velikim brojem manje-više standardnih entiteta i ako želi da ih povezane čuva u nekoj od standardnih baza podataka, vreme potrebno da uspostavi svoj projekat može u startu da se produži.

Clojure, kao član *Lisp* familije, ima razvijenu kulturu pisanja internih jezika specifičnih za domen. U ovom radu će biti predstavljen jedan takav jezik kao predlog za rešavanje gorenavedenog problema.

2. TRENUTNO STANJE U OBLASTI

Kada govorimo o perzistenciji podataka na raspolaganju su nam različiti koncepti. Neki od njih su veoma niskog nivoa apstrakcije, drugi rešavaju problem na višim nivoima.

2.1. Java Database Connectivity — JDBC

JDBC je *Java API* koji definiše kako klijent komunicira sa bazom podataka. Deo je *Java Standard Edition* od verzije 1.1 iz 1997. godine. Klase koje spadaju u *JDBC* nalaze se u paketu „*java.sql*“. Namenjen je u glavnom za rad sa relacionim bazama podataka. Od verzije 3.1 razvoj *JDBC* je pod procesom razvoja u okviru *Java* zajednice (*Java Community Process*). *JDBC* dozvoljava da postoji nekoliko implementacija i da one budu korišćene od strane iste aplikacije. *API* ima mehanizam koji obezbeđuje da se dinamički učitavaju korektni *Java* paketi kroz *JDBC Driver Manager*. Taj *Manager* se koristi kao fabrika za stvaranje *JDBC* konekcija (ali se u praksi češće koriste *connection pools* nego direktno uzimanje konekcija iz *JDBC Driver Manager-a*). *JDBC* omogućuje davanje naredbi bazama podataka. Naredbe mogu biti *update*, kao što su u *SQL* *CREATE*, *INSERT*, *UPDATE* i *DELETE*, ili upitne kao što je *SQL* *SELECT*. Takođe je moguće izvršavati i uskladištene procedure. Upitne komande vraćaju *JDBC row result set*. Kroz ovu strukturu se može prolaziti i dobijati rezultate upita, pri tom se kolonama može pristupiti po indeksu ili po imenu. Struktura takođe ima i meta podatke koji govore o tome kog su tipa kolone.

Clojure ima dve *JDBC* implementacije.

2.2 *clojure.java.jdbc*

Namenjen je da bude *wrapper* niskog nivoa oko različitih *JDBC* drajvera za baze podataka. Generalna ideja je da se postavi izvor podataka — „*database spec*“ i da se potom on prosleđuje raznim *CRUD* (*create*, *read*, *update*, *delete*) operacijama koje pruža „*java.jdbc*“. Po automatizmu, svaka operacija će otvoriti konekciju ka bazi podataka i izvršiti se unutar transakcije. Moguće je raditi više

operacija unutar jedne konekcije, bilo kroz transakciju, bilo kroz *connection pooling* ili kroz deljenu konekciju.

```
(def db-spec
  {:classname "com.mysql.jdbc.Driver"
   :subprotocol "mysql"
   :subname    "//127.0.0.1:3306/moja_baza_podataka"
   :user       "nalog"
   :password   "lozinka"})
```

Slika 1. Kreiranje *database spec* *clojure.java.jdbc*

Ako želimo da izvršimo neki upit preko iste konekcije na bazu podataka možemo to da uradimo korišćenjem makroa „*with-db-connection*“ iz *clojure.java.jdbc* biblioteke. Primer za to je dat na slici 3 gde učitavamo sve redove čija kolona „*ime*“ sadrži slova „*ike*“, potom uzimamo prvu od njih i snimamo u bazu njen duplikat.

```
(ns db-primer
  (:require [clojure.java.jdbc :as jdbc]))

(with-db-connection [db-con db-spec]
  (let [redovi (jdbc/query db-con
    ["SELECT * FROM table WHERE ime like '%?%' " "ike"])]
    (jdbc/insert! db-con :table (dissoc (first redovi) :id))))
```

Slika 2. Ponovno korišćenje konekcije na bazu podataka

2.3 *clojure.jdbc*

Naprednija biblioteka od prethodne. Napravljena je sa idejom da reši neke od nedostataka prethodne:

- Upravljanje konekcijama u „*clojure.jdbc*“ je jednostavnije i eksplicitnije. „*java.jdbc*“ ne pravi jasnu razliku između specifikacije baze i same konekcije na nju. To može izgledati kao fleksibilnije rešenja ali ono za posledicu ima da svaka funkcija koja prima konekciju mora da ima svu kompleksnost obrade tog ulaznog parametra.
- „*clojure.jdbc*“ dolazi sa sređenim upravljanjem transakcijama koje u potpunosti podržava ugnježdene transakcije i zadavanje transakcionih strategija. Nasuprot nje je „*java.jdbc*“ koja dolazi samo sa jednom transakcionom strategijom i ne podržava ugnježdene transakcije („*poravnava*“ transakcije — čini da se sve transakcije pretvore u jednu što može dovesti do pogrešnih sigurnosnih ograničenja).
- Dolazi sa ugrađenom podrškom za *connection pooling*.
- Uglavnom ima bolje performanse zbog bolje implementiranog *JDBC*. „*clojure.jdbc*“ implementacija je čistija i ima manje nepotrebnog (*boilerplate*) koda.

2.4 *HoneySQL*

Ideja *HoneySQL-a* je predstavljanje *SQL-a* kao *Clojure* struktura podataka. Upiti se posledično mogu generisati i u vreme izvršavanja bez potrebe za spajanjem stringova. Svi upiti se definišu na nivou *Clojure* mapa. Tako na primer jedan prost upit izgleda kao na slici 4.

```
(def sql-mapa {:select [:ime :prezime :nadimak]
               :from   [:covek]
               :where  [:= :covek.mesto_stanovanja "Novi Sad"]})
```

Slika 3. Primer *HoneySQL* upita

2.5 SQLingvo

Interni *DSL* koji se može koristiti iz *Clojure* ili *ClojureScript*. Kompatibilan je sa „*clojure.jdbc*“, „*clojure.java.jdbc*“, „*postgres.async*“ i „*node-postgres*“ (kroz *ClojureScript*) drajverima. Osnovna namena mu je da se koristi sa *Postgres* bazom podataka, ali ako se ne koriste *Postgres* specifične komande onda se može koristiti i sa ostalim bazama podataka. Ako je potrebno napraviti neki parametrizovani upit to se radi uz pomoć funkcija i *Lisp* citiranja sintakse (*syntax quoting*). Na slici 5 je dat primer parametrizovanog upita.

```
(defn pretrazi-po-mestu-stanovanja
  [baza grad]
  (sql (select baza [:ime :prezime]
        (from :covek)
        (where ` (= :grad ~grad))))))

(pretrazi-po-mestu-stanovanja baza "Novi Sad")
;; => ["SELECT \"ime\", \"prezime\" FROM \"covek\"
      ; ; WHERE (\n\"grad\" = ?) \"Novi Sad\""]
```

Slika 4. Primer parametrizovanog upita u SQLingvo

3. IMPLEMENTACIJA

Larva je izvedena kao kod generator. Ulaz u taj generator je opis podataka izražen u internom *Clojure DSL*, a izlaz su funkcije koje rukuju opisanim podacima i *DDL* kod koji uspostavlja odgovarajuću šemu baze podataka. Za implementaciju *Larve* korišćena je između ostalog i *HugSQL* biblioteka.

3.1 Larva DSL

Larva DSL je prost *Clojure* interni *DSL*. Zapravo, on predstavlja jednu *Clojure* perzistentnu mapu u kojoj su ključevi i njihove vrednosti unapred opisane i pre generisanja koda se validiraju. *Clojure* ekosistem ima nekoliko biblioteka koje bi mogle biti od koristi pri validiranju pomenute mape. Najsveobuhvatnija među njima je „*plumatic/schema*“ koja će i biti upotrebljena. *Clojure* će od verzije 1.9 (koja je trenutno u alfa izdanju) predstaviti „*clojure.spec*“ sistem koji će potom, u okvirima *Larva* sistema, biti razmatran kao zamena za „*plumatic/schema*“. Sam *Larva DSL* se potom pretvara u usmereni graf koji služi kao osnova za donošenje zaključaka potrebnih za generisanje koda.

3.2 Meta model Larva DSL i *plumatic/schema*

Clojure je u osnovi dinamički tipiziran programski jezik. To u ovom praktičnom slučaju znači da nema ugrađeni sistem tipova koji bi nam mogao pomoći da eventualno validiramo neke delove *Larva DSL-a* u vreme kompajliranja. Ono što nam donosi „*plumatic.schema*“ je da opišemo kog su oblika podaci koji ulaze u neku funkciju i kog su oblika oni koji iz nje izlaze. Ako neki poziv ove funkcije ne zadovoljava osobine opisane šemama, „*plumatic.schema*“ će baciti grešku koja opisuje koji deo prosleđenih podataka ne odgovara definisanoj šemi za funkciju. Tako funkciju koja prevodi *Larva DSL* u graf možemo definisati kao na slici 6.

```
(s/defn ~:always-validate ->graph :- ubergraph.core.Ubergraph
  "Transforms standard application program (uni-model)
  to its graph representation."
  [program :- Program]
  ...)
```

Slika 5. Potpis funkcije koja pretvara *Larva DSL* u graf I time ćemo postići da ovoj funkciji ne može biti prosleđena ni jedna struktura koja ne zadovoljava pravila *Larva DSL-a* opisanog u šemi „*Program*“. Ta šema je opisana izrazom na slici 7.

```
(s/def Program
  "Schema for standard app program."
  {(s/optional-key :about) About
   (s/optional-key :meta) Meta
   :entities Entities})
```

Slika 6. Šema programa

Ona deklariše svoje podšeme od kojih je jedna „*About*“ koja predstavlja sekciju o meta podacima modelovanog sistema podataka. Njen opis je dat na slici 8.

```
(s/def About
  "Schema for about section of program."
  {:name s/Str
   (s/optional-key :author) s/Str
   (s/optional-key :comment) s/Str})
```

Slika 7. Šema meta podataka

Šema koja opisuje same modelovane podatke je „*Entities*“ – predstavlja skup jedne ili više pojava „*Entity*“ šeme koja je data na sledeći način (Slika 9).

```
(s/def Entity
  {:signature s/Str
   :properties Properties
   (s/optional-key :plural) s/Str})
```

Slika 8. Šema jednog entiteta

Dalje se svaka pojedinačna osobina entiteta opisuje jednom šemom „*Entities*“ koja je data na slici 10.

```
(s/def Property
  {:name s/Str
   (s/optional-key :type) PropertyDataType})
```

Slika 9. Šema jedne osobine entiteta

Dakle, svaka osobina može imati tip. Mogući tipovi se opisuju zasebnom šemom koja je data na slici 11.

```
(s/def PropertyDataType
  (s/conditional
    keyword? SimpleDataType
    string? CustomDataType
    #(and (map? %) (= 1 (count %))) Collection
    :else SomethingWithReference))
```

Slika 10. Šema tipa jedne osobine entiteta

Kada znam kako je definisan meta model *Larva DSL* u terminima „*plumatic/schema*“ šema, možemo dati jedan primer opisa entiteta u njemu (slika 12).


```

{:about
{:name "Pilot model"
:author "Pera Peric"
:comment "Model muzickog festivala."
:entities
[{:signature "Muzicar"
:properties [{:name "ime" :type :str}
{:name "prezime" :type :str}
{:name "nadimak" :type :str}
{:name "priznanja" :type {:coll :str}}
{:name "bend" :type {:one :reference
:to ["Bend"]}}]}

{:signature "Bend"
:properties [{:name "ime" :type :str}
{:name "zanr" :type :str}
{:name "velicina" :type :str}
{:name "clanovi" :type {:coll :reference
:to ["Muzicar" "bend"]}}]}

{:signature "Festival"
:properties [{:name "ime" :type :str}
{:name "lokacija" :type :geo}]}]}

```

Slika 11. Primer opisa podataka u Larva DSL

3. 3 Graf reprezentacija i Ubergraph

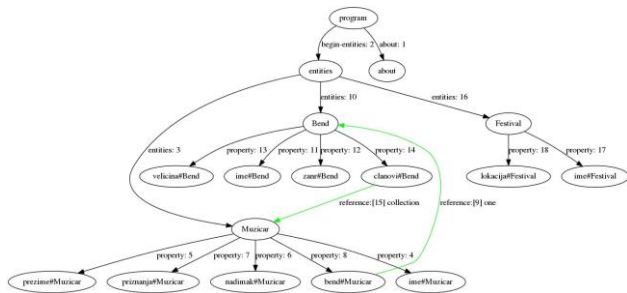
Videli smo da funkcija „>graph“ sa slike 6 vraća „Ubergraph“. *Ubergraph* je svestrana struktura tipa grafa za opštu upotrebu. Pružena je kroz istoimenu biblioteku. Predstavlja dopunu i proširenje *Loom* biblioteke koja predstavlja kolekciju graf protokola i algoritama. Pored toga što nam pruža mogućnost uspostavljanja graf struktura podataka daje nam i mogućnost grafičkog predstavljanja tih grafova kroz *Graphviz* biblioteku.

Ova mogućnost *Ubergraph-a* nam omogućuje da svoje entitete predstavljene kroz *Larva DSL* iscrtavamo kao grafove (stabla) i time dobijemo solidnu vizuelizaciju na kojoj možemo da uvidimo veze između entiteta.

Ako želimo da predstavimo naš primer sa slike 12 to možemo da uradimo koristeći funkciju „ubergraph.core/viz-graph“.

Dakle, prvo svoj model „festival“ prosledimo funkciji koja to pretvori u graf a potom graf vizualizujemo preko *Graphviz-a*.

Ovo se može postići izrazom „(u/viz-graph (->graph festival))“. Kao rezultat dobijamo graf sa slike 13.



Slika 12. Graf prikaz primera opisa podataka

3. 4 Generisanje početnih migracija

Na osnovu internog modela *Larva* sistem je u stanju da izgeneriše sve početne migracije baze podataka što umnogome ubrzava početak razvoja.

Sistem takođe vodi računa o integraciji sa programskim okvirima u kojima se nalazi. Shodno tome, migracije se vrše na standardan način za korišćeni programski okvir (*framework*). Na primer, za *Luminus*[5] se migracije pokreću sa „lein run migrate“ ili „lein run rollback <id_migracije_1> <id_migracije_2> ...“.

3. 5 Generisanje funkcija

Pored toga što daje migracije kojima se uspostavlja baza podataka *Larva* pruža i niz funkcija koje omogućavaju korisniku da na idiomatski način pravi i koristi podatke (poštujući šemu ako ona postoji). Ovo se postiže tako što se korisniku pruža niz funkcija za rukovanje tim podacima koje su generisane tako da oslikavaju oblik (ili šemu) podataka opisanu pomoću *Larva DSL-a*. Svi opisani entiteti imaju sledeće četiri osnovne funkcije: *create-ime_entiteta!*, *update-ime_entiteta!*, *get-ime_entiteta*, *delete-ime_entiteta!*. Pored ovih osnovnih funkcija biće napravljena i nekolicina dodatnih, za primer sa slike 12 dobićemo još: *get-muzicar-priznanja*, *assoc-muzicar-priznanja!*, *dissoc-muzicar-priznanja!*, *dissoc-all-muzicar-priznanja!*, *get-muzicar-bend*, *get-bend-clanovi*, *assoc-muzicar-bend!*, *assoc-bend-clanovi!*, *dissoc-muzicar-bend!*, *dissoc-bend-clanovi!* i *dissoc-all-bend-clanovi!*.

4. ZAKLJUČAK

Predmet ovog rada je bila implementacija sistema za perzistenciju podataka u *Clojure* okruženju. Ovaj rad je motivisan idiomatskim i praktičnim nedostacima uočenim pregledom i analizom stanja u oblasti. *Larva* predstavlja težnju da se u *Clojure* okruženju idiomatski podrži izražavanje i perzistencija podataka koncipiranih po ideji relacionog modela podataka. Prednosti *Larva* sistema pri rukovanju i perzistenciji podataka u odnosu na ostala rešenja su: izražavanje modela podataka u obliku perzistentne mape, podrška za migracije, *Larva* pruža mogućnost integracije sa okvirima, *Larva* funkcije za rukovanje podacima su napravljene sa osnovnom idejom da podrže idiomatske konstrukcije *Clojure* programskog jezika a ne relacionih baza podataka. U budućim verzijama razvoj *Larve* bi mogao biti koncentrisan na raščlanjivanje trenutne monolitne implementacije i generisanje *Larva* modela podataka iz već postojećih šema baza podataka (za relacione baze podataka).

5. LITERATURA

- [1] Voelter, Markus, et al. *DSL Engineering-Designing, Implementing and Using Domain-Specific Languages*. dslbook.org, 2013.
- [2] John McCarthy. Recursive functions of symbolic expressions and their computation by machine, part i. *Communications of the ACM*, 1960.
- [3] Alonzo Church. *The calculi of lambda-conversion*. Princeton University Press, 1941.
- [4] Paul Graham. *The roots of lisp*. <http://lib.store.yahoo.net/lib/paulgraham/jmc.ps>, Januar 2002. [Online; accessed 13-Avgust-2016].
- [5] Dmitri Sotnikov. *Luminus Web framework*. <http://www.luminusweb.net/> [Online; accessed 13-Avgust-2016].

Kratka biografija:

Novak Boškov rođen je u Novom Sadu 1992. godine. Master rad na Fakultetu tehničkih nauka, u Novom Sadu iz oblasti Elektrotehnike i računarstva – softversko inženjerstvo, odbranio je 2016. godine.

ANDROID APLIKACIJA ZA AKVIZICIJU SIGNALA VIBRACIJA I NAKNADNA OBRADA U SVRHU DETEKTOVANJA KVARA MOTORA**ANDROID APPLICATION FOR VIBRATION SIGNAL ACQUISITION AND SIGNAL ANALYSIS FOR ENGINE FAULT DETECTION**

Žolt Varga, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu je predstavljeno jedno od rešenja akvizicije signala vibracije i implementiran je sistem detekcije kvara motora na Android platformi. Analiza se vrši u vremenskom i frekventnom domenu uz pomoć softverskog paketa Matlab i njegovim alatima (Toolbox-ovima).

Abstract – This paper presents one of the solutions of vibration signal acquisition and implementation of engine fault detection system on Android platform. Signal Analysis was performed in time and also frequency domain with Matlab software and their toolboxes.

Ključne reči: Android platforma, Java, vremenska i frekventna analiza, numerika, optimizacija, Matlab, akvizicija signala vibracije, detekcija kvara asinhronog motora.

1. UVOD

Asinhroni motori zauzimaju važnu ulogu u industriji zbog njihove relativno male cene i širokih mogućnosti. Da bi se detektovao kvar pre potpunog otkaza i izvršen remont motora, prikazaćemo jedno od rešenja detekcije [4].

Android mobilni uređaji su veoma zastupljeni i tehnološki veoma napredovali zadnjih godina. Zbog njihovih zastupljenosti i hardverskih senzora pogodni su za akviziciju signala vibracije.

Za implementaciju Android aplikacije u radu je prikazan razvoljno okruženje Android Studio, sa korišćenim alatima.

2. ANDROID STUDIO I ALATI

Za Android postoje dva primarno integrisana developer-ska okruženja (IDE), a to su Android Studio i Eclipse. IDE je glavni program gde se piše kod i sklapaju aplikacije. Pomaže u organizaciji ili editovanju raznih fajlova u aplikaciji, upravljanju paketima i nudi podršku za biblioteke, te testiranju istih na stvarnim uređajima ili emulatorima [1]. Za rad na Android Studio-u potrebno je imati preinstaliran Java Development Kit (JDK).

Za testiranje aplikacije nije potrebno imati više uređaja, sve se može obaviti na emulatoru. Postoje male razlike na emulatoru, nemoguće je virtualizovati senzore, kao što je u našem slučaju.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Zoran Jeličić.

Pre upotrebe emulatora potrebno je instalirati SDK. Android SDK je razvio Google. Android SDK podržava razne operativne sisteme kao što su *Linux*, *Mac OS* i *Windows* [1].

Android Studio već dolazi sa u sebi implementiranim bild sistemom (Gradle).

ADB (Android Debug Bridge) - primarna svrha ovog alata je pokretanje softvera i izvršenje promena na uređajima ukoliko su povezani sa računarom [1].

Da bi se mogao koristiti emulator treba se izraditi AVD (Android Virtual Device). Za izradu novog AVD-a treba pokrenuti alat AVD Manager koji se nalazi na alatnoj traci [1].

Aplikacija je jedna datoteka aplikacijskog paketa. APK Datoteka (Android Application Package) je spakovana datoteka koja sadrži izvršni kod koji je pisan u Java programskom jeziku.

3. ANDROID OPERATIVNI SISTEM I ARHITEKTURA

Arhitektura Android operativnog sistema deli se na pet celina:

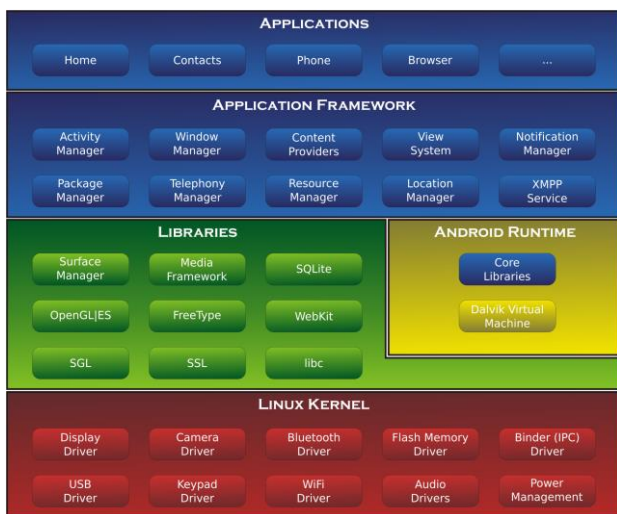
- Aplikacije (Applications): Contacts, Browser...
- Aplikacioni frejmwork (application framework): Activity Manager, Window Manager, Content Providers, View System, Location Manager...
- Biblioteka (Libraries): SQLite, WebKit, SSL, SGL, FreeType, OpenGL...
- Android runtime: Core Libraries, Dalvik Virtual Machine.
- Linux Jezgra (Linux Kernel): Display Driver, Camera Driver, Flash Memory Driver, Binder Driver, Wifi Driver, Audio Drivers...

Bazne biblioteke pisane su u programskom jeziku Java i predstavljaju sve esencijalne klase kao što su klase za manipulaciju kolekcijama, klase za komunikaciju sa okolinom i slično [3].

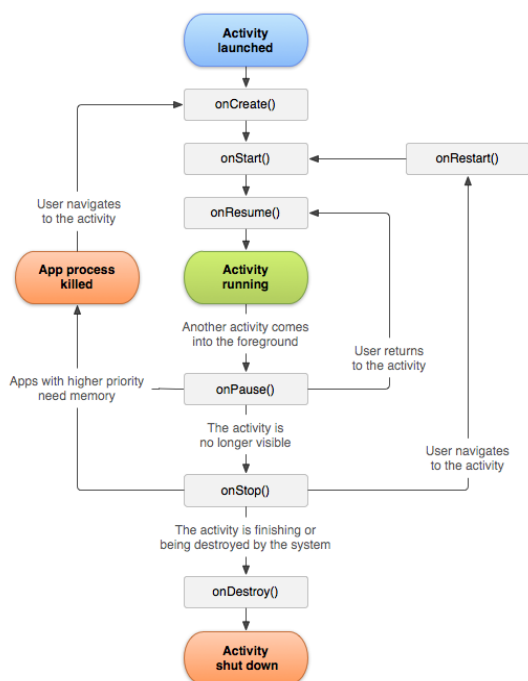
Sloj aplikativnih okvira napisan je u programskom jeziku Java i sadrži proširiv skup programskih komponenti kojeg koriste sve aplikacije uređaja. Ovaj sloj obezbeđuje različite karakteristike Android OS-a.

Aktivnost (Activity) je grafički korisnički interfejs koji korisnik u jednom trenutku može videti na uređaju. To mogu biti na primer postavke za aplikaciju ili neka lista gde korisnik mora nešto odabrati. Aplikacija se sastoji od

više aktivnosti između kojih korisnik može prelaziti. Aplikacija ima glavnu aktivnost koja se prikaže kada se pokrene aplikacija. Aplikacija pruža prelazak s jedne aktivnosti na drugu, može prekinuti jednu aktivnost i pokrenuti aktivnost u drugoj aplikaciji [3].



Slika 3.1 Arhitektura Android operativnog sistema



Slika 3.2. Životni vek aktivnosti

Activity klasa sadrži sledeće metode koja predstavljaju različita stanja aktivnosti: *onCreate()*, *onResume()*, *onPause()*, *onDestroy()*, *onStart()*, *onRestart()*, *onStop()*. Čitav životni ciklus aktivnosti se dešava između *onCreate()* – *onDestroy()* metoda.

Životni ciklus vizuelno prisutne aktivnosti je između *onStart()* – *onPause()*. Aktivnost je u prednjem planu (u fokusu) između *onResume()* – *onPause()* metoda.

Namere (Intents) su poruke koje se šalju između glavnih gradivnih blokova. Namere pokreću aktivnosti, govore uslugama da se pokrenu, zaustave ili šalju poruke svima i

ne moraju čekati da bude završene. [3]

Usluge (Service) se izvršavaju u pozadini i nemaju komponente korisničkog interfejsa. Usluge mogu obavljati iste poslove kao aktivnosti, ali bez korisničkog interfejsa [3].

Kontekst aplikacije odnosi se na okruženje aplikacije i procesa u kojem se izvršavaju sve njegove komponente. Kontekst aplikacije omogućava aplikacijama da dele podatke i resurse između različitih gradivnih blokova. Referencu na kontekst možemo jednostavno dobiti pozivanjem *Context*. *getApplicationContext()* ili *Activity.getApplication()* [3].

Svaka Android aplikacija mora imati korisnički interfejs. Dva načina izrade korisničkog interfejsa su deklarativno i programsko. Deklarativni korisnički interfejs koristi XML (Extensible Markup Language) za opisivanje izgleda. Programsko korisnički interfejs piše se Java kodom.

Najbolje je koristi oba interfejsa. XML pristup koristi se za deklarisanje svih statičkih delova interfejsa, na primer raspored elementa na ekranu, a programski pristup služi za zadavanje radnje za određene elemente koji se dinamično menjaju.

Android organizuje elemente korisničkog interfejsa u rasporede (layouts) i prikaze (views). Postoji nekoliko rasporeda. To su *LinearLayout*, *TableLayout*, *FrameLayout*, *RelativeLayout* i *AbsoluteLayout* [1].

Datoteka *AndroidManifest.xml* se nalazi u svakoj Android aplikaciji. Bez ove datoteke Android aplikacija se ne može pokrenuti. Ova datoteka sadrži ključne informacije o Android aplikaciji, a to su naziv paketa koji služi kao jedinstveni identifikator Android aplikacije, opis svih komponenti na primer aktivnosti i usluga, dozvole za pristup aplikacionim komponentama, popis klasa, popis biblioteka i minimalni nivo Android API-ja.

4. IMPLEMENTACIJA SISTEMA

Aplikacija se bazira na dva dizajn šablona (design pattern):

- MVC (Model View Controller) – veoma je zgodan šablon kada se interfejs i specifikacija često menja. Čini redizajn jednostavnijim, tako i izmene algoritama. Takođe omogućuje da se model prikaže na više načina (na različitim uređajima).
- Singleton – naglasak na bezbednosti za niti (thread safe)

Većina android uređaja poseduje senzor vibracije tj. akcelerometar, ali nemaju svi linearni akcelerometar koji eliminiše hardverski uticaj gravitacije. Da bi aplikaciju mogli da primenimo na što više različitih uređaja, softverski eliminišemo gravitaciju upotrebom *high-pass* filtera. Na ovaj način efikasno se eliminiše gravitacija uz minimalan gubitak preciznosti.

```

public class Singleton {
    private Singleton instance;

    private Singleton() {
        //lots of initialization code
    }

    public static synchronized Singleton getInstance() {
        if(instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}

```

Listing 4.1. Koncept bezbednog načina korišćenja niti

```

public class SensorActivity extends Activity implements SensorEventListener
{
    private final SensorManager mSensorManager;
    private final Sensor mAccelerometer;

    public SensorActivity() {
        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        mAccelerometer =
mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }
    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(this, mAccelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
    }
    protected void onPause() {
        super.onPause();
        mSensorManager.unregisterListener(this);
    }
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }
    public void onSensorChanged(SensorEvent event) {
    }
}

```

Listing 4.2. Implementacija servisa akvizicije senzora vibracije

```

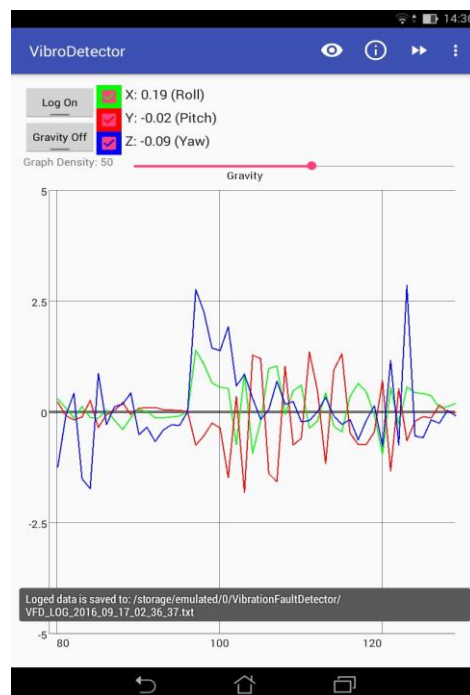
modeList.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> arg0, View view, int arg2,
long itemID) {
        progress.show(); // LOADING BAR show
        _ChosenFile = files[arg2];
        // Before dismiss dialog, trigger share instance
        levelDialog.dismiss();

        // Resolving loading in background Thread
        final Handler handler = new Handler();

        final Thread thread = new Thread() {
            @Override
            public void run() {
                try {
                    while(!_ProcessTrue) {
                        loadDataAndAnalyse(_ChosenFile);
                        progress.dismiss(); // LOADING BAR dismiss
                        sleep(100);
                        handler.post(this);
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        };
        thread.start();
    }
});
levelDialog.show();

```

Listing 4.3. Nit za učitavanje i obradu signala vibracije



Slika 4.1. Izgled aplikacije pri beleženju akcelerometra

Bitna napomena jeste da na Android operativnom sistemu beleženje podataka senzora nije *real time* (u realnom vremenu), nemoguće je dobiti industrijsku preciznost.

5. DETEKCIJA KVARA

Da bi detektovali kvar rada asinhronog motora koristili smo niz algoritama. Prvo je rađena analiza u vremenskom domenu korišćenjem RMS (Efektivna vrednost signala) I Crest faktora.

$$x_{rms} = \left(\frac{1}{N} \sum_{i=1}^N x^2(i) \right)^{1/2}$$

Listing 5.1. RMS algoritam

$$C = \frac{\max(x(i)) - \min(x(i))}{x_{rms}}$$

Listing 5.2. Crest faktor

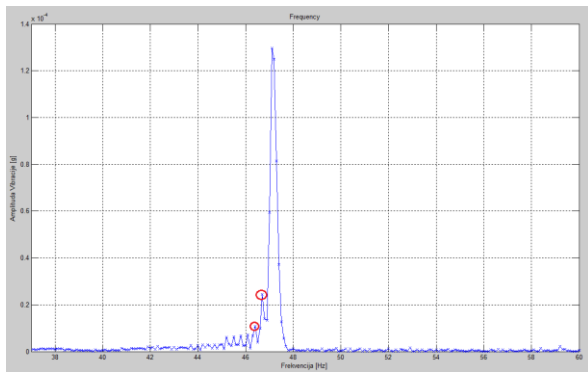
Ova tehnika analize se veoma brzo izvršava na uređajima.

Međutim u eksperimentu je grešila u skoro 40% slučajeva. Zbog nepreciznosti analize u vremenskom domenu, okrećemo se analizi u frekventnom.

Prosečna brzina učitavanje određenog kanala i detekcije za vremenski domen:

- HTC Desire X: 1minut 15s
- Asus MemoPad 7: 12s

Frekventna analiza se započinje sa FFT (brzom Furijeovom transformacijom). Oštećenje šipke rotora se u spektru signala vibracija ogleda u prisustvu povećane amplitude spektra na frekvencijama bočnih opsega.



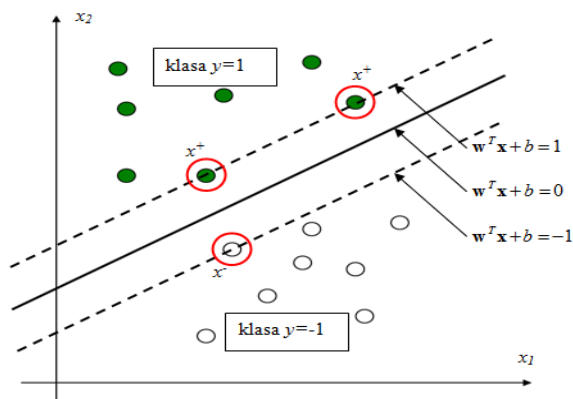
Slika 5.3. Bočni opsezi frekventne analize

Granica amplitude bočnih opsega određuje se standardnom devijacijom.

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

Listing 5.4. Standardna Devijacija

SVM (Super Vektore Machines) ili potporni vektori se koriste u ovoj fazi, radi detekcije kvara na osnovu skupa obeležja signala vibracija. Kao ulazni podatak u SVM klasifikator koristi se vektor podataka $\mathbf{x}_i$ ($i = 1, 2, \dots, M$), gde je M ukupan broj uzoraka. Svaki od ulaznih uzoraka ima N dimenzija i pripada jednoj od dve klase, pozitivnoj ili negativnoj. Osnovni problem SVM klasifikatora je određivanje optimalnog položaja hiperravni koja razdvaja uzorke koji pripadaju različitim klasama. [4]



Slika 5.5. Hiperravan koji razdvaja u dve klase upotrebom SVM

Zbog jednostavnijeg izračunavanja, ovaj problem se svodi na određivanje minimuma funkcije $\|\mathbf{w}\|^2$ po parametrima $\mathbf{w}$ i b , može se rešiti upotrebom Lagranževe funkcije [4]:

$$L(\mathbf{w}, b, \alpha_i) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^M \alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

Listing 5.6. Langranžev množitelj

6. ANALIZA I REZULTATI

Signali za analizu su sadržali različita opterećenja motora (100%, 50%, minimalno). Takođe su različitog kvaliteta. Iako rezultat nije na zadovoljavajućem nivou, uz dalji razvoj bi lako došli do poboljšanja. U Tabeli 6.1. su dati rezultati klasifikacije.

	Ukupan br. signala za obuku	Br. signala sa greškom motora	Br. signala bez greške	Br. signala za testiranje	Preciznost
Test 1	156	78	78	10	80%
Test 2	104	52	52	6	57%
Test 3	234	156	78	24	67%
Test 4	182	104	78	17	82%

Tabela 6.1. Finalni rezultati tesiranja

7. ZAKLJUČAK

Cilj projekta je bio da se pokuša napraviti prenosiva Android aplikacija za detekciju kvara motora, u čemu smo i delimično uspeali. Konstatovali smo da je usko grlo Java baratanje sa numeričkim algoritmima, za razliku od Matlaba. Aplikacija ima potencijala da se uz mnogo optimizacije koda koristi u stvarnim industrijskim problemima, uz ograničenje da se koristi na uređajima sa jačim performansama.

8. LITERATURA

- [1] Milena Jovanović - Master rad, Razvoj interaktivnog turističkog vodiča na android platformi primenom lokacijskih servisa (2012)
- [2] Android documentation, <https://developer.android.com/>
- [3] Android Arhitektura, <http://blog.vladoivankovic.from.hr/>
- [4] Željko Kanović - Doktorska Disertacija, Modifikacije algoritma optimizacije rojem čestica sa primenom u detekciji kvarova na objektima automatskog upravljanja sa kontinualnim dinamikom (2011)

Kratka biografija

Varga Žolt je rođen 04.02.1986. godine u Zrenjaninu. 2005. godine upisao se na Fakultet tehničkih nauka, na studijskom programu Računarstvo i Automatika. Osnovne bečelor studije je završio 2012 godine, iste godine se upisuje na master usmerenje Automatika i upravljanje sistemima.

SIMULACIJA SAOBRAĆAJA UPOTREBOM AGENTSKE TEHNOLOGIJE I SARL PROGRAMSKOG JEZIKA**TRAFFIC SIMULATION USING AGENT TECHNOLOGY AND SARL PROGRAMMING LANGUAGE**

Aleksandra Aleksić, Milan Vidaković, *Fakultet tehničkih nauka, Novi Sad*

Oblast - ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj: U ovom radu je opisana agentska tehnologija koja je bila korišćena prilikom realizacije simulacije saobraćaja u realnom vremenu. Opisan je algoritam po kojem se vrši pretraga sledeće moguće putanje vozila. Objašnjava se upotreba Web Socket tehnologija i Java Swing biblioteka prilikom iscrtaavanja putanje vozila u realnom vremenu. Implementacija je izvršena upotrebom Java i SARL programskog jezika u okruženju Eclipse (verzija Mars).

Abstract: This paper gives a detailed description about the agent technology that was used for the realization of the traffic simulation in real time. It describes the traversal algorithm (first in depth search) which is used to find the next possible route the vehicle should take. It gives details about the Web Socket technology and Java Swing library and how they were they used to give a visual result of this project. The environment in which the project is implemented is the Eclipse. Java and SARL were the main programming languages used for the realization of this project.

Cljučne reči: Agent technology, Web Sockets, Java Swing.

1. UVOD

Porastom učesnika u saobraćaju nastaju gužve na putevima u većim gradovima. Problem saobraćajne signalizacije (engl. traffic light problem) predstavlja pronalazak optimalne putanje tako da vozilo iz polazne tačke stigne do ciljne u najkraćem mogućem roku. Prilikom prolaska kroz ovu putanju, vozilo treba da što kraće čeka na semaforima. Za simulaciju mreže semafora koji menjaju stanje korišćen je SARL programski jezik. Agentska tehnologija daje mogućnost implementacije dinamičkog sistema semafora koji se nalaze na raskrsnicama. Kako bi se rezultati vizuelno prikazali, korišćena je Java Swing biblioteka za iscrtaavanje inicijalne mreže, kao i za prikaz kretanja vozila u realnom. Prilikom kretanja vozila kroz mrežu, postoji mogućnost da se put račva, tj. postoji više ruta na koje vozilo može da se pomeri. Podaci su organizovani u stablo i zbog toga je implementiran algoritam "prvi u dubinu". Ako se naiđe na više mogućnosti, svaka putanja, veza između određištog i ciljnog čvora, dobija različite težinske vrednosti kako bi se mogla izabrati sledeća povoljna putanja.

NAPOMENA:

Ovaj rad je proistekao iz master rada čiji mentor je bio prof. dr Milan Vidaković.

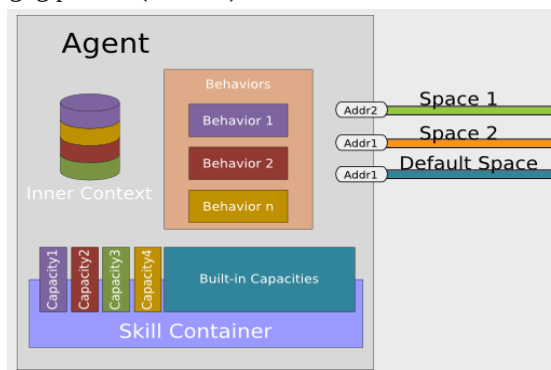
2. SARL PROGRAMSKI JEZIK

SARL [1] i Java programski jezici se ne razlikuju mnogo po sintaksi. SARL programski jezik je korišćen za delimičnu realizaciju ovog projekta zbog agentske tehnologije koju podržava. Omogućava kreiranje agenata koji mogu da funkcionišu u prostoru i da obavljaju poslove u paraleli, prilikom čega međusobno komuniciraju mehanizmom razmena poruka. Glavne komponente SARL programskog jezika koje će biti opisane su:

- agent,
- događaj (engl. Event) i
- prostor (engl. Space).

Agenti i njihovi događaji su napisani u posebnom fajlu cityagent.sarl na osnovu kog se pomoću biblioteke org.eclipse.xtext generiše java klasa koja realizuje opisani agente i događaje.

Agent je samostalni entitet koji ima skup: veština (engl. skills) preko koga ustanovljava sa kolikim kapacitetom raspolaže, ponašanja (engl. behaviors) kojima odgovara na događaje koji su ugrađeni ili definisani unutar paketa u kojem se agent nalazi ili uvedeni (engl. import) iz nekog drugog paketa. (Slika 1.)



Slika 1. Na slici je predstavljena arhitektura agenta [1]

Agent se deklarise upotrebom ključne reči **agent**. Agent može da se definiše kao prazan agent, bez atributa, veština, ponašanja. Prilikom pokretanja standardnog agenta sledeći koraci se izvršavaju:

- inicijalizacija: agent reaguje na **Initialize** događaj,
- životni vek: agent izvršava svoja ponašanja u dva slučaja: agenti reaguju kada dobiju događaj (engl. reaktive) ili agenti sami izvršavaju sopstveno ponašanje
- uništavanje: agenti reaguju na **Destroy** događaj.

Postoje dva tipa agenta. Master agent, kao glavni agent iz kojeg se inicijalizuju (engl. slave) agenti. Master agent je u ovom radu opisan kao grad (engl. city) dok su ostali (engl.

slave) agenti koji se kreiraju (engl. spawn) iz master agenta zapravo raskrsnice u mreži. Bitna polja u master agentu su:

- `hasLight` – vrednost koja označava da li postoji još aktivnih (engl. slave) agenata, jer ne može da se uništi master ako postoji još podagenata,
- `car` – instanca vozila koji se kreće u mreži i
- `listOfPossibilities` – lista ako ih u datom trenutku više postoji.

Agent raskrsnica (engl. slave) se kreira prilikom instanciranja master agenta. Dalje u tekstu biće indentifikovan kao `TrafficLightAgents`. Broj `TrafficLightAgents` zavisi od broja čvorova definisanih u `Networks.xml` fajlu. Bitna polja u `TrafficLightAgents` agentu su:

- `id` – jedinstvena vrednost preko koje se razlikuje svaka raskrsnica, jednaka je `id`-u čvora,
- `passed` – vrednost koja govori da li je vozilo prošlo kroz tu raskrsnicu i
- `greenLight` – vrednost koja govori o stanju u kojem se nalazi semafor (crveno, zeleno).

Događaj je jedan od fundamentalnih koncepata u SARL programskom jeziku. Predstavljen je kao struktura podataka koja je sačinjena od atributa. Svaki atribut ima svoj naziv, tip i vrednost. Događaji pokreću agent. Događaji se razmenjuju između agenata unutar zadatog prostora. Definisanje događaja podrazumeva navođenje ključne reči **event** potom naziv događaja. Događaj može isto kao i agent da bude prazan. Takođe, može da sadrži informacije koje su opisane atributima.

Događaj se šalje primaocima (engl. emit) u prostoru naredbom:

```
emit (e : Event)
```

na koju reaguju svi agenti koji se nalaze u tom prostoru i koji imaju definisano ponašanje na taj događaj:

```
on <EventName>
```

Neki od događaja koji su bili implementirani u svrhu simulacije semafora su:

- event *VechileStarted* – informisanje ostalih (engl. slave) agenata da je vozilo pokrenuto,
- event *Moving* – informisanje ostalih (engl. slave) agenata da se vozilo pomerilo u novi čvor,
- event *ChangeLight* – (engl. slave) agent sam sebe informiše da treba da promeni stanje svetla na semaforu.

Prostor u kojem se definiše agent predstavlja (engl. default) prostor u njemu se ostali (engl. slave) agenti kreiraju (engl. spawn) i imaju mogućnost da međusobno komuniciraju preko definisanih događaja.

Pokretanje agenta iz koda je omogućeno Janus platformom [2]. Janus platforma obezbeđuje **Boot** klasu koja obezbeđuje metodu `startJanus` koja pokreće Janus platformu. Pokretanje master agenta je omogućeno definisanjem objekta `Kernel` i pozivom metode `startJanus` gde se kao parametar navodi master agent `traffic.City.class` koji se želi pokrenuti:

```
Kernel janusKernel =  
Boot.startJanus((Class<? extends Module>)  
null, traffic.City.class, args);
```

U gornjem primeru, Janus pokreće `City` agenta.

3. ALGORITAM PRVI U DUBINU

Vozilo koje se kreće kroz mrežu u nekim trenucima može da ima više mogućih ruta na raspolaganju. Podaci o mogućim rutama su smešteni u stablo, gde su čvorovi identifikacioni brojevi raskrsnica, a povezani čvorovi označavaju povezane raskrsnice.

Ovi podaci se čitaju iz fajla `Network.java`. Implementiran je algoritam "prvi u dubinu" kako bi se mogla izvršiti efikasna pretraga podataka koji su organizovani kao stablo.

Dodate su takođe i težine (engl. cost) vezama. Ove težine se računaju u realnom vremenu, tj. prilikom izvršavanja programa. Kao što je bilo napomenuto, vozilo u trenutku pretrage može da naiđe na čvor, raskrsnicu, odakle ima više mogućih putanja.

Kako bi se odlučilo za najpovoljniju, sve rute će međusobno biti upoređene i u zavisnosti od osobina kao što su razdaljina između posmatranog čvora i ciljnog čvora, vreme koliko je potrebno da iz početnog stigne do ciljnog čvora i vreme trajanja crvenog svetla u ciljnom čvoru, cene će biti sabrane i po svakom kriterijumu rute će biti sortirane.

Ruta sa najnižom cenom, težinom, će se gledati kao ruta sa najvećim prioritetom i ako je zeleno svetlo na toj raskrsnici, trenutno vozilo prolazi. Ako je svetlo na semaforu crveno, onda se ide na sledeću putanju u nizu. Najveći prioritet u svakom slučaju ima raskrsnica koja nema semafora, npr. kružni tok.

```
procedure dfs(G, s)
```

```
  S := stack contains only s
```

```
  while S not empty
```

```
    v := S.pop()
```

```
    if v not visited:
```

```
      mark v as visited
```

```
      for w G.neighbors(v) : s.push(w)
```

Pseudo-kod 1. Algoritam "prvi u dubinu" baziran na steku [3]

4. VIZUALIZACIJA

Za prikaz rezultata prethodno opisane implementacije semafora i kretanja vozila, izabrana je Java Swing biblioteka za iscrtavanje inicijalne mreže semafora u čvorovima koji su povezani putanjima.

Da bi se omogućilo ažuriranje mreže u zavisnosti od novog položaja vozila, korišćena je Web Socket tehnologija za komunikaciju vozila koja omogućuje da se u realnom vremenu javlja novi položaj vozila prilikom pomeranja, pa Java Swing aplikacija može da pruži vizuelni prikaz mreže.

Java Swing

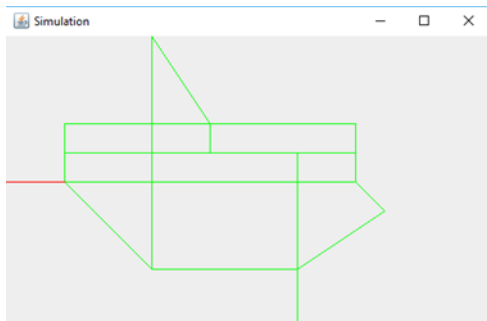
Korisnički interfejs (engl. User Interface) UI, predstavlja objekat koji ima ulogu posrednika između računara i korisnika.

Osnovna komponenta Swing biblioteke jeste `JComponent`. Predstavlja nadklasnu svake komponente i apstraktna je klasa. (Slika 2.)

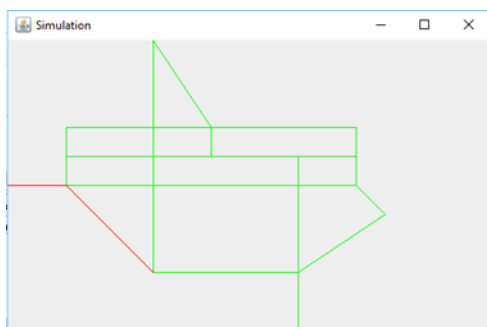
klijentu da javi trenutnu poziciju vozila koja se metodom `paintNewRoute` iscrtava na panel.

Svaka nova pozicija vozila šalje se preko `sendMessage` metode klase `Car`, koja se preko `WSManage` čvorišta šalje `NetworkPanel` klijentu i time se slika ažurira u realnom vremenu.

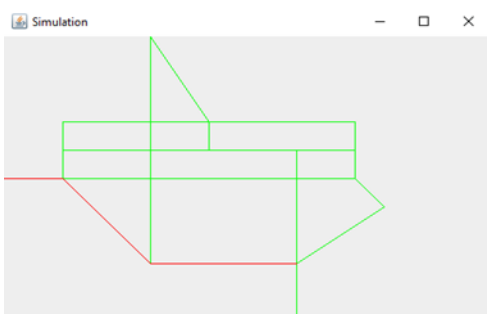
Slike 4, 5, 6 i 7 prikazuju kretanje vozila u vremenu.



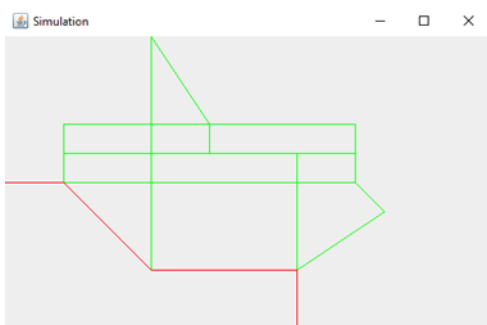
Slika 4 Početak prolaska kroz grad



Slika 5 Prolazak prve raskrsnice



Slika 6 Prolazak druge raskrsnice



Slika 7 Završetak prolaska kroz grad

5. ZAKLJUČAK

Cilj ovog rada je bio da se implementira aplikacija koja će simulirati saobraćaj u realnom vremenu. Prvi korak je bio odabir tehnologije preko koje će se simulacija semafora implementirati i za to je izabran SARL programski jezik. Realizacija simulacije semafora upotrebom agentske tehnologije se pokazalo kao uspešnim programskim rešenjem. Potom je implementirana pretraga prvi u dubinu za pronalazak sledećih čvorova u mreži. Inicijalizacija težine na vezama u realnom vremenu je bila neophoda kako bi vozilo u zavisnosti od položaja u kojem se u datom trenutku nalazi moglo da proceni sledeću optimalnu putanju. Na kraju je omogućena vizuelizacija ovog programskog rešenja u realnom vremenu preko Java Swing biblioteke i Web Socket.

6. LITERATURA

- [1] SARL Programming Language, <http://www.sarl.io/docs/suite/io/sarl/docs/SARLDocumentationSuite.html>
- [2] Janus Platform, <http://www.janusproject.io/>
- [3] Graph Traversal, <http://www.cs.cmu.edu/~ckingsf/class/02713-s13/lectures/lec07-dfsbfs.pdf>
- [4] Micheal Abernethy, Introduction to Swing, 2005., <http://www.ibm.com/developerworks/java/tutorials/j-intswing/j-intswing.html>
- [5] Web Sockets Introduction, http://www.tutorialspoint.com/websockets/websockets_tutorial.pdf
- [6] Glassfish Quick Start Guide, <https://glassfish.java.net/docs/4.0/quick-start-guide.pdf>

Kratka biografija:

Aleksandra Aleksić je rođena 18.06.1992. godine u Novom Sadu. Osnovnu školu „Jovan Popović“ završila je 2007. godine. Gimnaziju „Jovan Jovanović Zmaj“ u Novom Sadu završila je 2011. Iste godine upisala se na Fakultet tehničkih nauka, odsek *Računarstvo i automatika*. Smer *Primenjene računarske nauke i informatika*, *Inteligentni sistemi*. Položila je sve ispite u roku predviđene planom i programom.

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*.

SERVISNO-ORIJENTISANI DIZAJN U RUBY PROGRAMSKOM JEZIKU**SERVICE-ORIENTED DESIGN IN RUBY PROGRAMMING LANGUAGE**Jelena Keravica, Milan Vidaković, *Fakultet Tehničkih Nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj: U radu je opisan servisno-orijentisani dizajn softverskih sistema, konkretno u smislu korišćenja mikroservisne arhitekture. Dat je pregled faktora kvaliteta softvera i kako ovakav pristup dizajnu na njih utiče. Ponuđeno je rešenje u Ruby on Rails i Sinatra razvojnim okvirima, zasnovanim na Ruby programskom jeziku.

Abstract: This paper describes service-oriented design of software systems, particularly regarding microservice architecture. It presents software quality factors and how this design approach can affect them. A solution is offered using Ruby on Rails and Sinatra, both based on the Ruby programming language.

Ključne reči: servisno-orijentisani dizajn, mikroservisi, softverska arhitektura, web aplikacije, behaviour-driven development, continuous integration, web socket

1 UVOD

Ideja podele web aplikacije na više nezavisnih servisa orijentisanih oko specifičnih funkcionalnosti prisutna je u softverskoj industriji kao način na koji se može ubrzati razvojni proces. Mikroservisna arhitektura je našla primenu u mnogim velikim kompanijama, koje tvrde da im takav pristup modelovanja sistema omogućava bolju organizaciju posla, slobodnije eksperimentisanje sa novim tehnologijama i bržu implementaciju novih funkcionalnosti. U skladu s tim, u ovom radu akcenat je na internim servisima u odnosu na eksterne i kako oni mogu poboljšati saradnju čitavog razvojnog tima unutar neke organizacije.

Ovaj rad opisuje kako se servisno-orijentisani dizajn softvera može primeniti koristeći *Ruby on Rails* i *Sinatra* razvojne okvire, dat je uvid u to kako ovakva arhitektura utiče na faktore kvaliteta softvera i objašnjena je njena uloga u prevazilaženju kompleksnosti.

2 SERVISNO-ORIJENTISANI DIZAJN

U osnovi, arhitektura softverskih sistema je skup struktura potrebnih da bi se o sistemu moglo rezonovati, a one obuhvataju softverske elemente, odnose među njima i osobine tih elemenata i njihovih odnosa. Budući da arhitektura obuhvata softverske elemente i odnose među njima, ona takođe isključuje detalje elemenata koji nisu važni za razumevanje njihovih odnosa. Ovakva apstrakcija je ključna za savladavanje kompleksnosti sistema [1].

Servisno-orijentisani dizajn softverskih sistema je dizajnerska paradigma sastavljena od mnoštva dizajnerskih principa, čijom primenom nastaje servisno-orijentisana

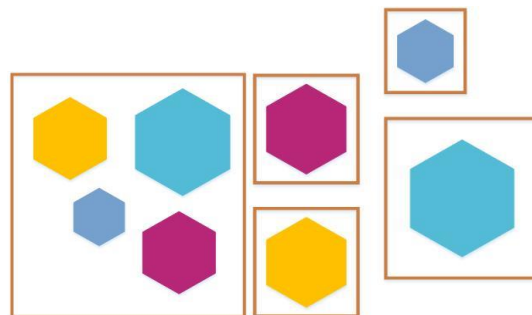
arhitektura (SOA). Postoji više definicija servisno-orijentisane arhitekture. Jedna od njih glasi: SOA je paradigma za organizaciju i korišćenje distribuiranih odlika sistema koje mogu biti u vlasništvu različitih domena. Ono u čemu se ogleda vrednost takve arhitekture je stvaranje okvira koji povezuje potrebe i rešenja koja odgovaraju tim potrebama. Centralni koncept definicije, servis, je mehanizam kojim se povezuju potrebe i rešenja u SOA [1].

2.1 Mikroservisi

Termin „mikroservis“ prvi put je iskorišćen 2011. godine. [2] Mikroservisna arhitektura se definiše kao pristup razvoju jedne složene aplikacije kao skupa malih nezavisnih procesa koji komuniciraju međusobno putem API-a nezavisnih od jezika implementacije. Ovi servisi su mali, nezavisni i fokusirani na obavljanje malog zadatka, što olakšava modularni pristup u razvoju sistema. Takođe se može reći da je mikroservisna arhitektura distribuirana aplikacija čiji svi moduli su mikroservisi. U sklopu te definicije, mikroservis je minimalni nezavisni proces koji komunicira putem poruka, gde „minimalni“ znači da sadrži samo one funkcionalnosti koje pripadaju njegovom domenu [2, 3].

Trenutno postoji velika dilema da li su ta dva pristupa povezana i na koji način, [4, 5, 6] ali upoređivanjem njihovih osnovnih principa, može se reći da je mikroservisna arhitektura u osnovi podvrsta servisno-orijentisane arhitekture, odnosno da je nastala evolucijom ideje SOA. U odnosu na servisno-orijentisanu, mikroservisna arhitektura prenosi neke od njenih ideja sa nivoa organizacije, odnosno preduzeća, na aplikaciju.

Mikroservis je takva komponenta koje se može nezavisno izvršavati. Suprotno njima, monolitne aplikacije dizajnirane su s ciljem kreiranja jedinstvene izvršne verzije aplikacije. Moduli unutar monolitne aplikacije su zasnovani na deljenju resursa jedne iste mašine (memorije, baze podataka, datoteka). Mikroservisna arhitektura premešta komunikaciju iz memorije na mrežu – koristeći protokole na kojima je zasnovan *world wide web* [6, 2].



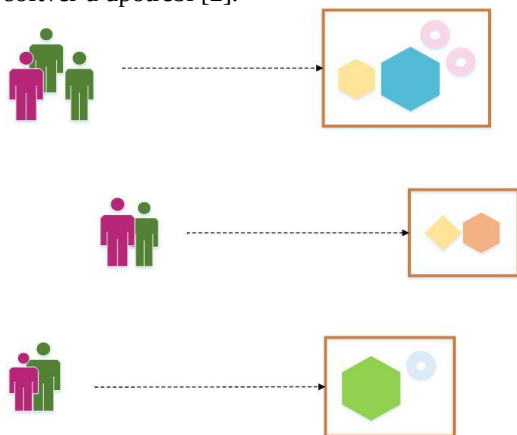
Slika 2.1 Levo: moduli unutar monolitne aplikacije, desno: moduli izdvojeni u zasebne mikroservise

Svaka promena u nekom od modula monolitne aplikacija zahteva ponovno kreiranje izvršne verzije čitave aplikacije. To je i jedan od glavnih razloga za korišćenje servisa kao komponenti [2].

Slika 2.1 prikazuje razliku između monolitne i mikroservisne arhitekture. Svaki od šestouglova predstavlja neku komponentu sistema. Na levoj slici, one su oivičene granicom jedne aplikacije, a na desnoj je svaka funkcionalnost smeštena u zaseban proces.

Raslojavanje je jedan od najčešćih pristupa u dizajnu softvera za organizaciju kompleksnih sistema. Ovakav obrazac prisutan je i u arhitekturi računara gde su slojevi organizovani od programskih jezika, preko sistemskih poziva ka uređajima do skupa instrukcija na računarskom procesoru. Sličan obrazac je primenjen i u dizajnu računarskih mreža [7]. To najčešće dovodi do formiranja timova zaduženih za korisnički interfejs, timova za razvoj logike sistema na serverskoj strani, timova za bazu podataka i slično.

Pristalice mikroservisne arhitekture su više naklonjene posmatranju mikroservisa kao proizvoda za koji je njegov tim odgovoran tokom čitavog njegovog životnog veka. Cilj je da razvojni tim preuzme potpunu odgovornost za svoj softver u upotrebi [2].



Slika 2.2 Organizacija multidisciplinarnih timova prema poslovnim mogućnostima

Modelovanje mikroservisa prema poslovnim mogućnostima se dovodi u vezu sa pojmom ograničenog konteksta *domain-driven* dizajna.

Za timove koji rade u takvom okruženju, jasno definisani kontekst im daje jednu važnu prednost: jasnoću. Oni donose odluke o dizajnu unutar svog konteksta, a ostali timovi dobijaju na slobodi da ne moraju da koriste jedan isti model [8].

Ovakav pristup uključuje široku upotrebu tehnika automatizacije infrastrukture – kontinualne integracije i isporuke. Kontinualna isporuka je disciplina razvoja softvera gde se softver pravi tako da može biti pušten u upotrebu u bilo kojem trenutku. [9] Kontinualna isporuka se postiže kontinualnom integracijom, što podrazumeva praksu da članovi tima integrišu svoje izmene često (bar jednom dnevno). Svaka integracija se verifikuje automatskim kreiranjem izvršnog koda i testiranjem da bi se integracione greške otkrile što pre. Ovakav pristup primenjen na razvoj mikroservisa u odnosu na čitavu aplikaciju omogućava brži razvoj softvera i postepeni prelazak na novije verzije mikroservisa [10].

Jedna od posledica centralizovanog upravljanja je tendencija da se standardizuje jedna tehnologija. Iskustvo je pokazalo da takav izbor ograničava i da je ugodnije koristiti pravi alat za određeni problem. Primenom mikroservisne arhitekture se u teoriji za svaki servis može koristiti druga tehnologija. Razvojni timovi imaju visok stepen slobode izbora konfiguracije razvojnog okruženja koje najbolje odgovara njihovim potrebama (u smislu troškova i kvaliteta mikroservisa) [2].

2.2 Uticaj na faktore kvaliteta softvera

Kvalitet softvera najbolje se opisuje kombinacijom nekoliko faktora. U ovom radu su razmotreni neki od osnovnih i najčešćih faktora: tačnost, robusnost, modularnost, efikasnost, pravovremenost i bezbednost [11].

Neki od principa mikroservisne arhitekture su direktno povezani sa faktorima kvaliteta softvera, kao, na primer, modularnost. Slabo spregnuti i nezavisni servisi su laki za održavanje, dalje evoluiranje i iskorišćavanje.

Na faktore tačnosti i robusnosti, odnosno pouzdanosti sistema, u velikoj meri utiče automatizacija infrastrukture. Najslabija tačka pouzdanosti mikroservisa je u njihovoj integraciji, a pretpostavka da je mreža pouzdana je prva od zablude distribuiranog računarstva. Zbog toga posebnu pažnju treba obratiti na pouzdanost mehanizama razmene poruka [6, 12].

Servisi moraju biti dizajnirani tako da budu otporni na otkaze ostalih servisa. Ovo je mana u odnosu na monolitnu arhitekturu, jer se uvodi dodatni sloj kompleksnosti s kojim se treba izboriti.

Jedan od argumenata za korišćenje mikroservisne arhitekture radi poboljšanja efikasnosti je način na koji se oni mogu horizontalno skalirati. Naime, broj komponenti u mikroservisnoj arhitekturi može se dinamički menjati, bez potrebe za replikacijom čitave aplikacije [13].

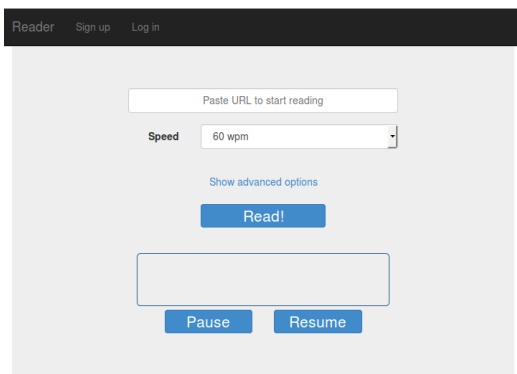
Sa aspekta bezbednosti, jedan od preporučenih načina za upravljanje mikroservisima, a ujedno i obezbeđivanje njihove sigurnosti je *API gateway* obrazac, čija ideja je zasnovana na *Facade* obrascu objektno-orijentisanog dizajna, ali ovakav pristup može stvoriti usko grlo u čitavom sistemu, kako u razvoju, tako i u komunikaciji [14].

3 IMPLEMENTACIJA REŠENJA

Kao primer servisno-orijentisane arhitekture, razvijena je web aplikacija za brzo čitanje. Ova aplikacija radi tako što korisnik unese web adresu članka ili neke druge stranice sa koje želi da čita tekst, odabere brzinu čitanja i klikne na *Read* (slika 3.1). Aplikacija će izdvojiti tekst članka sa stranice i prikazivati reči odabranom brzinom unutar sivog polja na stranici, jednu po jednu. Ideja je da se na taj način čitaocu drži pažnja, dok je fokusiran samo na jedan deo stranice.

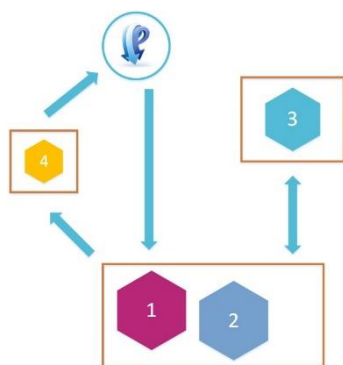
Aplikacija za brzo čitanje će poslužiti kao dobar primer u ovom radu, jer sadrži dovoljno elemenata da bi mogla biti dekomponovana na mikroservise.

Na dijagramu (slika 3.2) je grafički prikazan odnos softverskih elemenata ovog rešenja. Donji, najveći, pravougaonik predstavlja Rails aplikaciju koja unutar sebe obuhvata dva modula: registraciju i prijavljivanje korisnika (1) i čuvanje istorije čitanja (2). Parsiranje HTML stranice dobijene od strane korisnika je izdvojeno



Slika 3.1 Početna stranica aplikacije

u Sinatra aplikaciju (3). Dvosmerna strelica na slici označava da Rails aplikacija šalje ovom servisu URL, a kao odgovor dobija izdvojeni tekst i naslov koji čuva u svoju bazu podataka. Za prikaz reči na stranici korisnika korišćen je *WebSocket* protokol. U ovom radu je korišćen eksterni servis koji implementira ovaj protokol, *Pusher*, i na dijagramu je to njegov logo u kružnici u gornjem delu.



Slika 3.2 Arhitektura rešenja: (1) Registracija i prijavljivanje korisnika, (2) čuvanje istorije čitanja, (3) parsiranje HTML stranice, (4) slanje reči na *Pusher* HTTP API

Ruby on Rails, ili jednostavno *Rails*, je razvojni okvir za web aplikacije napisan u *Ruby* programskom jeziku. Prvi put je objavljen u julu 2004. godine i zasnovan je na *model-view-controller* (MVC) šablonu.

Feature: Reading History

As a user

I want to see a list of all the pages I have read

Background:

Given I am on the home page

And I am signed in

Scenario: Listing all of user's previous pages

Given I have read pages in the past

When I go to the listing page

Then I should see all the pages I have read

Listing 3.1 Scenario koji opisuje čuvanje istorije čitanja

Sinatra je, kao i *Rails*, zasnovan na *Ruby* programskom jeziku. Nije neuobičajeno da cela Sinatra aplikacija bude smeštena u jednu datoteku i zbog toga je pogodan za kreiranje mikroservisa koncentrisanih na jednu jednostavnu funkcionalnost [15].

Prilikom razvoja rešenja korišćeni su BDD (*behaviour-driven development*) alati, koji pružaju mogućnost da se korisnički zahtevi pretvore u automatizovane testove koji u isto vreme usmeravaju inženjere, verifikuju da li je funkcionalnost pravilno implementirana i dokumentuju ono što aplikacija zaista radi. Preciznije, ovi zahtevi se nazivaju još i „izvršne specifikacije“. Za opis ponašanja aplikacije korišćeni su *Cucumber* scenariji, čiji primer se može videti u listingu 3.1.

BDD je praksa razvoja softvera koja treba da pruži tehnike za prevazilaženje nesigurnosti u poznavanju poslovnog domena i rizika koji s tim dolazi. Cilj ovog pristupa je da se razvoj fokusira na isporuku funkcionalnosti koje su od značaja za poslovni domen tako što će ponuditi zajednički jezik koji spaja tehnologiju i poslovni domen [16].

Zajedno sa *Cucumber*-om, korišćen je i *RSpec*, koji učestvuje u sklopu unutrašnjeg ciklusa BDD-a. *Cucumber* se koristi za opis ponašanja aplikacije, a *RSpec* za opis ponašanja objekata u aplikaciji. Deo jedne takve specifikacije (bez implementacije) dat je u listingu 3.2.

RSpec.describe Page do

```
describe "#words" do
  it "splits text into words"
end

describe "#text" do
  context "text attribute exists" do
    it "reads attribute from database"
  end

  context "text attribute doesn't exist" do
    it "sends post request to fetch the text"
  end
end
```

end

Listing 3.2 Izvršna specifikacija Page modela

Za testiranje komunikacije među servisima, potrebno je simulirati slanje poruka. Za ovo je korišćen *Ruby* gem pod nazivom VCR, koji čuva HTTP zahteve i odgovore u datoteku. Cilj je da se na ovaj način napravi skup testova koji će biti brzi i deterministički, odnosno da se eliminiše faktor spoljnih uticaja na rezultat testa.

Podrazumevano podešavanje za VCR je da se stvarni zahtevi upućuju servisima samo prvi put, nakon toga se interakcija čuva u datoteci i svako naredno izvršavanje će se oslanjati na prethodno izvršene zahteve. Nakon inicijalnog podešavanja, VCR se koristi dodavanjem oznaka na *Cucumber* scenario koji šalje zahtev ka nekom servisu [17].

HTML kôd korisničkog interfejsa sa slike 3.1 generiše se od strane Rails aplikacije. Da bi se na toj stranici prikazivale reči, potrebno je ostvariti konekciju sa *Pusher* servisom. Ovo se postiže instanciranjem *JavaScript* klijenta na stranici. Svaki klijent prati poruke na određenom kanalu i reaguje na određene događaje.

4 ZAKLJUČAK

Oblast mikroservisa je još uvek mlada. Interesovanje za mikroservise je u značajnom porastu tek od početka 2015. godine.

Dostupna literatura takođe nije velikog obima, a postojeće tehnologije su tek u začetku. Dokumentacija je često neformalnog karaktera i retko zalazi u srž implementacije ovakve arhitekture. To samo znači da postoji veoma mnogo prostora za istraživanje i unapređenje tehnologija i obrazaca arhitekture.

Sumirano, u primeni nekog od oblika mikroservisne arhitekture najznačajnija prednost je fleksibilnost u razvoju koju ovakav pristup pruža. Svaki servis se može postepeno zameniti nekim drugim ili samo reimplemen- tirati koristeći neke potpuno drugačije tehnologije, bez velikog uticaja na ostatak sistema. To je ogromna prednost za savremene web aplikacije, pogotovo za kompanije koje brzo rastu i menjaju se.

S druge strane, iako ova arhitektura ima cilj da smanji kompleksnost sistema, odnosno da razvojnom timu pomogne u boljem razumevanju domena, posledica je da se kompleksnost prebacuje ne druge aspekte sistema, kao što su testiranje, bezbednost i orkestracija saradnje među mikroservisima. Izvori koji navode prednosti mikroservisne arhitekture se često ne fokusiraju dovoljno na njene bezbednosne aspekte. Takođe, mnoge pozitivne osobine mikroservisne arhitekture (efikasnost, bolja organizacija timova) polaze od pretpostavke da su granice mikroservisa dobro definisane i da su oni nezavisni, što se u praksi teško postiže.

Ovakav oblik arhitekture softverskih sistema će sigurno naći primenu u mnogim kompleksnim web rešenjima u budućnosti. To će uticati na dalji razvoj tehnologija koje služe implementaciji, diskusije o potencijalnim problemima i poboljšanjima i edukacije inženjera kako bi iz ideje mikroservisne arhitekture iskoristili ono što im može pomoći da prevaziđu svoje sadašnje probleme. Dosadašnji izazovi i način razmišljanja su doveli do zaključka da je automatizacija infrastrukture, odnosno eliminacija ljudskog faktora i ručnog obavljanja rutinskih zadataka, jedna od ključnih stavki mikroservisne arhitekture. Prema tome, pretpostavlja se da će sličan princip u budućnosti biti primenjen na testiranje komunikacije među servisima, njihovu orkestraciju i registraciju radi lakšeg pronalaženja i održavanje bezbednosti takvih dinamičkih sistema.

5 LITERATURA

- [1] K. L. F. M. P. F. B. R. M. C. Matthew MacKenzie, „Reference Model for Service Oriented Architecture 1.0,“ October 2006. [Na mreži]. Available: <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>.
- [2] M. Fowler, „Microservices,“ [Na mreži]. Available: <http://martinfowler.com/articles/microservices.html>.
- [3] R. M. M. M. A. Irakli Nadareishvili, „Microservice Architecture: Aligning Principles, Practices and Culture,“ O'Reilly, 2016, pp. 97-98.
- [4] B. Rhubart, „Microservices and SOA,“ *ORACLE Magazine*, March/April 2015.
- [5] M. Little, „SOA versus Microservices?,“ 13 December 2015. [Na mreži]. Available: <https://www.infoq.com/news/2015/12/soa-v-microservices>.
- [6] S. G. A. L. L. M. M. F. M. R. M. L. S. Nicola Dragoni, „Microservices: yesterday, today, and tomorrow,“ 13 June 2016. [Na mreži]. Available: <http://128.84.21.199/abs/1606.04036?context=cs>.
- [7] M. Fowler, „Patterns of Enterprise Application Architecture,“ Addison-Wesley Professional, 2002, p. 23.
- [8] E. Evans, „Domain Driven Design: Tackling Complexity In The Heart Of Software,“ PBL, 2008, pp. 378-385.
- [9] M. Fowler, „Continuous Delivery,“ 30 May 2013. [Na mreži]. Available: <http://martinfowler.com/bliki/ContinuousDelivery.html>.
- [10] M. Fowler, „Continuous Integration,“ 1 May 2006. [Na mreži]. Available: <http://martinfowler.com/articles/continuousIntegration.html>.
- [11] B. Meyer, „Object-Oriented Software Construction,“ Prentice Hall, 1997, pp. 3-16.
- [12] A. Rotem-Gal-Oz, „Fallacies of Distributed Computing,“ [Na mreži]. Available: <http://rgoarchitects.com/Files/fallacies.pdf>.
- [13] M. B. E. B. J. Erika Ábrahám, „Self-Reconfiguring Microservices,“ *Theory and Practice of Formal Methods (Springer)*, pp. 194-197, 2016.
- [14] C. Richardson, „Building Microservices: Using an API Gateway,“ 15 June 2015. [Na mreži]. Available: <https://www.nginx.com/blog/building-microservices-using-an-api-gateway/>.
- [15] K. H. Alan Harris, „Sinatra: Up and Running,“ O'Reilly Media, 2011, pp. 2-4.
- [16] D. North, „Behaviour-driven Development,“ 29 September 2014. [Na mreži]. Available: <http://behaviourdriven.org/>.
- [17] RelishApp, „Project: VCR,“ 2016. [Na mreži]. Available: <https://relishapp.com/vcr/vcr/docs>.

Kratka biografija:

Jelena Keravica je rođena 14.06.1991. u Novom Sadu, Republika Srbija. Školske 2010/2011. je upisala Fakultet tehničkih nauka u Novom Sadu, studentski program računarstvo i automatika. Po završetku studija 2014. godine je upisala master akademske studije na istom fakultetu, smer – računarstvo i automatika, modul – primenjene računarske nauke i informatika.

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*.

PRIMENA MONTE KARLO METODE ZA ODREĐIVANJE MERNE NESIGURNOSTI IMPLEMENTATION OF MONTE KARLO METHOD FOR THE DETERMINATION OF MEASUREMENT UNCERTAINTY

Marina Bulat, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu je dato poređenje dva načina određivanja merne nesigurnosti: po preporuci Međunarodne organizacije za standardizaciju – ISO i primenom Monte Karlo metode. Prvi koncept je opšte prihvaćen i zahtevan u savremenoj metrologiji, ali postoje specijalni slučajevi kada ne daje dobre rezultate. U tom slučaju koristi se Monte Karlo metoda.

Abstract – This paper analyzes the comparison of two methods for determining measurement uncertainty: as recommended by ISO - "International Organization for Standardization" and using the Monte Carlo method. The first concept is accepted and required in modern metrology, but there are special cases when it doesn't give good results. In this case, a Monte Carlo method should be used.

Ključne reči: merna nesigurnost, Monte Karlo metoda, reaktivna snaga

1. UVOD

Pri svakom merenju nastaje greška, tako da rezultat merenja ne predstavlja broj, nego interval u kom se nalazi stvarna vrednost merene veličine. Metrološki gledano, kvalitetnijem merenju, u pogledu tačnosti i preciznosti, odgovara uži interval u kojem se nalazi prava vrednost. Gledano kroz istoriju, korišćeni su različiti načini iskazivanja granica intervala. Pored poznatih koncepata, kao što su sigurne i statističke granice greške merenja, postojali su mnogi drugi načini. Zbog ovakve neusaglašenosti, često su nastajali problemi prilikom međulaboratorijskih i međudržavnih poređenja. U cilju standardizacije postupaka definisanja merne nesigurnosti, Međunarodna organizacija za standardizaciju je 1995. godine objavila dokument pod nazivom "Uputstvo za izražavanje merne nesigurnosti" (engleski: The Guide to the Expression of Uncertainty in Measurement, u daljem tekstu GUM).

GUM je fundamentalni referentni dokument za procenu i izražavanje merne nesigurnosti. Svrha postojanja GUM-a jeste da se uspostave jedinstvena pravila za izražavanje merne nesigurnosti za potrebe metrologije, standardizacije, etaloniranja i sl, da se obezbedi potpuna informacija o načinu na koji je dobijena merna nesigurnost i da se obezbede osnove za međunarodno upoređivanje rezultata merenja. Koncept na kom se GUM bazira:

1. sve komponente merne nesigurnosti su iskazane standardnom devijacijom,
2. sve sistematske greške su korigovane,
3. intervali svih komponenti nesigurnosti su simetrični,
4. komponente su grupisane u dve kategorije, u zavisnosti od izvora podataka, merna nesigurnost tipa A - komponente su procenjene statističkim metodama, i merna nesigurnost tipa B - komponente su procenjene metodama koje nisu statističke.

Procedura za određivanje merne nesigurnosti po GUM-u:

1. identifikovati sve bitne komponente merne nesigurnosti,
2. izračunati standardnu mernu nesigurnost svake komponente,
3. izračunati kombinovanu standardnu mernu nesigurnost,
4. izračunati proširenu mernu nesigurnost,
5. izraziti rezultat merenja koji se sastoji od najbolje procene vrednosti merene veličine i kombinovane ili proširene merne nesigurnosti.

Iako je poštovanje preporuke GUM-a danas praktično obaveza svih metroloških institucija, postoje situacije u kojima GUM nije primenljiv: a) ukoliko raspodela funkcija verovatnoće merene veličine nije normalna, b) kod asimetričnih i nelinearnih problema i c) kada se očekuje da je merna nesigurnost veličine koju procenjujemo reda veličine njene vrednosti, odnosno kada je relativno iskazana merna nesigurnost veća od nekoliko desetina procenata.

Ovaj rad se bavi specijalnim slučajem kada nije opravdano mernu nesigurnost iskazivati po preporuci GUM-a.

Radi ilustracije primenjivosti preporuka GUM-a u specijalnom slučaju, uzet je jednostavan primer određivanja reaktivne snage na osnovu merenja napona, struje i aktivne snage potrošača. Merna nesigurnost je određivana na dva načina: po preporukama GUM-a i korišćenjem Monte Karlo metode. Na osnovu merenja napona, struje i aktivne snage na potrošaču, reaktivna snaga se određuje prema formuli.

$$Q = \sqrt{(U \cdot I)^2 - P^2}$$

Kada je faktor snage blizak jedinici, odnosno kada je reč o pretežno termogenoj impedansi, pokazuje se da se primenom ovog načina dobijaju velike granice greške, odnosno velika merna nesigurnost. Ovo se dešava zbog

NAPOMENA:

Ovaj rad je proistekao iz master rada čiji mentor je bio dr Dragan Pejić.

oduzimanja dveju bliskih vrednosti: prividne snage (definisane proizvodom napona i struje) i aktivne snage. U ovakvoj situaciji se očekuje da preporuke GUM-a ne daju dobru procenu merne nesigurnosti. Ispravnost preporuka GUM-a u ovom primeru je utvrđena primenom Monte Karlo metode.

Zadatak rada je određivanje procene reaktivne snage Q , njoj pridružene standardne i proširene merne nesigurnosti $u(\Delta Q)$ i najužeg 95%-tnog intervala proširenja, korišćenjem preporuka GUM-a i primenom Monte Karlo metode.

2. ODREĐIVANJE MERNE NESIGURNOSTI PO GUM-U

Reaktivna snaga impedanse iskazana je pomoću formule:

$$Q = \sqrt{(U \cdot I)^2 - P^2} = 3.59 \text{ VAr}$$

gde je U očitani napon sa voltmetra u vrednosti od 230 V, I je intenzitet struje od 50 mA očitani sa miliampermetra i P je očitana aktivna snaga sa vatmetra u vrednosti od 10.92 W.

Neka su klase tačnosti svih instrumenata iste i neka iznose 0.5.

Preporuke GUM-a za procenu granica merne nesigurnosti podrazumeva sledeće korake:

1. definisanje merne veličine (reaktivna snaga Q) i razvoj matematičkog modela koji opisuje merenje,
2. identifikacija bitnih komponenti merne nesigurnosti, U , I i P i dodeljivanje raspodele verovatnoća svakoj od njih,
3. određivanje apsolutne greške napona, struje i snage, ΔU , ΔI i ΔP ,
4. izračunavanje standardne merne nesigurnosti svake komponente $u(\Delta U)$, $u(\Delta I)$ i $u(\Delta P)$,
5. izračunavanje kombinovane merne nesigurnosti, $u_c(\Delta Q)$,
6. izračunavanje proširene merne nesigurnosti

$$u_{\text{prošireno}}(\Delta Q) = k \cdot u_c(\Delta Q)$$

gde je za k uzeta vrednost 2, da bi se dobila merna nesigurnost sa poverenjem od 95 %.

7. izražavanje rezultata merenja u obliku:

$$Q \pm u_{\text{prošireno}}(\Delta Q).$$

Za napon i struju je uzeto da je funkcija raspodele uvek uniformna, a za snagu su posmatrane tri situacije: uniformna, Gausova, a zatim trougaona.

Iz formule za klasu tačnosti:

$$kl_{X\%} = \frac{\Delta X}{X_{\max}} \cdot 100\%,$$

gde X predstavlja komponentu merne nesigurnosti, dobijene su apsolutne greške napona, struje i snage redom: 1.5 V, 0.25 mA i 0.075 W. U tabeli 1 su prikazane merne nesigurnosti za ove veličine za sva tri slučaja. Dometi instrumenata su redom: 300 V, 50 mA i 15 W.

Kombinovanoj mernoj nesigurnosti reaktivne snage dobijamo pomoću izraza:

$$u_c(\Delta Q) = \sqrt{\left(\frac{\partial Q}{\partial I}\right)^2 \cdot u^2(\Delta I) + \left(\frac{\partial Q}{\partial U}\right)^2 \cdot u^2(\Delta U) + \left(\frac{\partial Q}{\partial P}\right)^2 \cdot u^2(\Delta P)}$$

a dobijeni rezultati za sva tri slučaja dati su u tabeli 2.

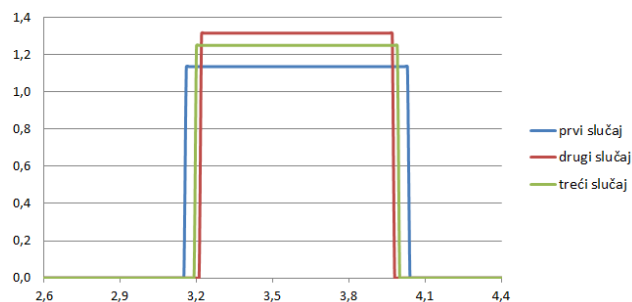
Tabela 1. Prikaz mernih nesigurnosti apsolutnih grešaka napona, struje i aktivne snage za sva tri slučaja

slučaj	$u(\Delta U)$	$u(\Delta I)$	$u(\Delta P)$
prvi	$\Delta U / \sqrt{3}$	$\Delta I / \sqrt{3}$	$\Delta P / \sqrt{3}$
drugi	$\Delta U / \sqrt{3}$	$\Delta I / \sqrt{3}$	$\Delta P / 3$
treći	$\Delta U / \sqrt{3}$	$\Delta I / \sqrt{3}$	$\Delta P / \sqrt{6}$

Tabela 2. Prikaz vrednosti proširene merne nesigurnosti, merne nesigurnosti iskazane u procentima i intervala koji predstavlja rezultat merenje za sva tri slučaja prema GUM-u

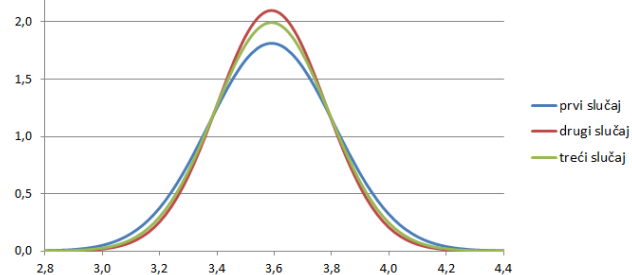
slučaj	$u_{\text{prošireno}}(Q)$ (VAr)	$u_c(Q)$ (%)	Interval (VAr)
prvi	0.43	12.19	(3.15 ; 4.03)
drugi	0.38	10.61	(3.21 ; 3.97)
treći	0.39	11.03	(3.19 ; 3.99)

Na slici 1 su prikazani intervali u kojima se nalazi tačna vrednost reaktivne snage računane prema GUM-u za sva tri slučaja. Veličine površina ispod krivih su vrednosti jedan (normalizovane), pa su iz tog razloga širi intervali viši i obrnuto.



Slika 1. Grafička predstava intervala dobijenih po preporuci GUM-a

Na slici 2 su prikazane tri normalne raspodele koje se po pretpostavci GUM-a dobijaju za raspodelu reaktivne snage za sva tri slučaja.

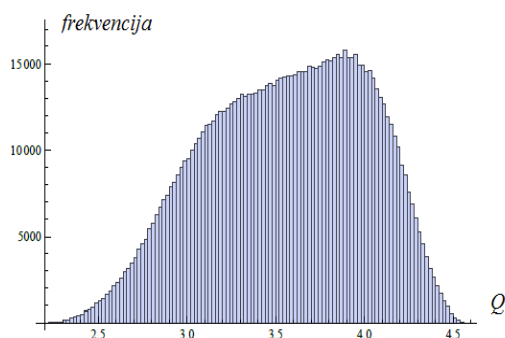


Slika 2. Raspodele reaktivne snage dobijene po preporukama GUM-a

3. ODREĐIVANJE MERNE NESIGURNOSTI PO MONTE KARLO METODI

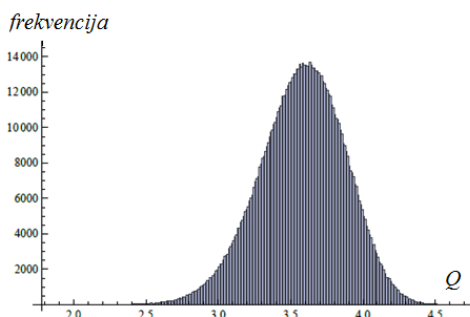
Monte Karlo metoda predstavlja postupak koji, na osnovu slučajnog uzorkovanja definisanih funkcija raspodela verovatnoća merenih veličina (napon, struja i aktivna snaga), generiše funkciju raspodele izlazne veličine (reaktivna snaga). Nasumično se vrši veliki broj uzastopnih merenja (ovde je posmatran uzorak od milion merenja), a u svakoj od situacija merene veličine (napon, struja i snaga) imaju različite vrednosti, ali takve da na ukupnom ansamblu bivaju zadovoljene pretpostavljene raspodele greške. Za celokupan ansambl se određuje funkcija gustine raspodele verovatnoće izlazne veličine - reaktivne snage. Uzimanjem opsega od 2.5 % do 97.5 % dobijaju se granice merne nesigurnosti za faktor proširenje $k=2$.

U prvom slučaju su dodeljene raspodele verovatnoća za sve tri ulazne veličine od interesa uniformne. Na osnovu Monte Karlo metode, interval u kom se nalazi tačna vrednost sa pouzdanošću od 95 % je (2.68 ; 4.31) VAR. Na slici 3 je prikazan histogram datih rezultata.



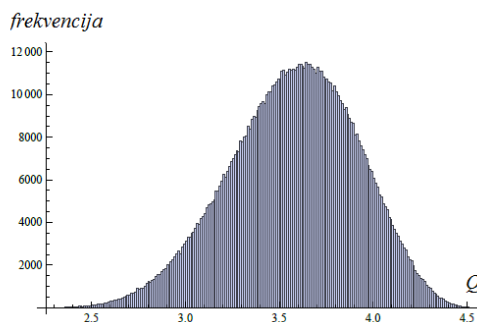
Slika 3. Raspodela rezultata po Monte Karlo metodi za uniformnu raspodelu napona, struje i aktivne snage

U drugom slučaju su dodeljene raspodele verovatnoća za napon i struju uniformne, a za aktivnu snagu Gausova. Na osnovu Monte Karlo metode, interval u kom se nalazi tačna vrednost je (2.97 ; 4.12) VAR. Na slici 4 je prikazan histogram datih rezultata.



Slika 4. Raspodela rezultata po Monte Karlo metodi za uniformnu raspodelu napona i struje, a Gausovu raspodelu aktivne snage

U trećem slučaju su dodeljene raspodele verovatnoća za napon i struju uniformne, a za aktivnu snagu trougaona. Na osnovu Monte Karlo metode, interval u kom se nalazi tačna vrednost je (2.87 ; 4.19) VAR. Na slici 5 je prikazan histogram datih rezultata.



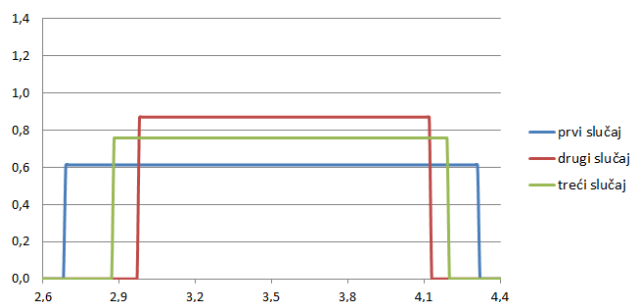
Slika 5. Raspodela rezultata po Monte Karlo metodi za uniformnu raspodelu napona i struje, a trougaonu raspodelu aktivne snage

U tabeli 3 dati su ukupni rezultati po Monte Karlo metodi.

Na slici 6 je grafička predstava rezultata dobijenih preko Monte Karlo metode.

Tabela 3. Rezultati po Monte Karlo metodi

slučaj	Interval (VAR)	u (%)
prvi	(2.68 ; 4.31)	23.32
drugi	(2.97 ; 4.12)	16.22
treći	(2.87 ; 4.19)	18.70



Slika 6. Grafička predstava intervala dobijenih pomoću Monte Karlo metode

4. POREĐENJE REZULTATA

U tabeli 4 su prikazani uporedo rezultati dobijeni po preporukama GUM-a i pomoću monte Karlo metode, prikazani u procentima.

Tabela 4. Uporedni rezultati procentualno iskazanih proširenih mernih nesigurnosti po Monte Karlo metodi i prema preporukama GUM-a

slučaj / metoda	MK	GUM	MK / GUM
prvi	23.32 %	12.19 %	1.91
drugi	16.22 %	10.61 %	1.53
treći	18.70 %	11.03 %	1.70

Prema GUM-u je dobijeno vrlo malo smanjenje merne nesigurnosti za tri posmatrana slučaja: 12.19 %, 10.61 % i 11.03 %. Dobija se ista procena za reaktivnu snagu bez obzira na variranje raspodela za U , I i P .

Po Monte Karlo metodi se dobijaju veće razlike kada se raspodele variraju za ova tri slučaja, i to 23.32 %, 16.22 % i 18.70 %.

16.22 % i 18.70 %. Dobijaju se različite procene vrednosti reaktivne snage: 3.49 VAR, 3.54 VAR i 3.53 VAR. Raspodele koje se dobijaju odstupaju od normalne i nisu simetrične.

Monte Karlo metoda daje veće merne nesigurnosti nego GUM i to redom: 1.91, 1.53, 1.70 puta za ova tri slučaja. Imajući u vidu da GUM i nije namenjen za ovako specijalan slučaj, kao i u način funkcionisanja Monte Karlo metode, logično je usvojiti merne nesigurnosti dobijene po Monte Karlo metodi.

Drugim rečima, merne nesigurnosti, koje bi neko dobio neopravdanom upotrebom preporuka GUM-a bi bile manje, odnosno interval u kojem se očekuje prava vrednost merene veličine bi bio uži.

5. ZAKLJUČAK

Na primeru određivanja reaktivne snage (pri faktoru snage bliskom jedinici, zbog čega se dobija vrlo velika merna nesigurnost, reda 10 %) je procenjena merna nesigurnost na dva načina: po preporuci Međunarodne organizacije za standardizaciju – ISO, i primenom Monte Karlo metode.

Po preporukama GUM-a dobijeni su rezultati manji u odnosu na rezultate Monte Karlo metode.

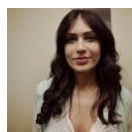
Ako bi neko, slepo sledeći preporuke GUM-a odredio mernu nesigurnost, ona bi mogla biti skoro duplo manja, a time i procenjeni interval u kojem se nalazi tačna vrednost duplo uža, od onog koji daje Monte Karlo metoda. U ovakvim graničnim situacijama, nije korektno koristiti GUM za određivanje merne nesigurnosti. Mnogo realniji rezultati se dobijaju primenom Monte Karlo metode.

Namena ovog rada nije nikako dovođenje u pitanje ispravnosti GUM-a, nego ukazivanje na jedan specijalan slučaj kada preporuka GUM-a ne daje dobar rezultat.

6. LITERATURA

- [1] „Evaluation of measurment data – Supplement 1 to “The guide to the Expression of Uncertainty in Measurement” - Propagation of distributions using a Monte Carlo method”, JCGM 101:2008
- [2] Stojaković M., “Uvod u teoriju verovatnoće i matematičke statistike”, Novi Sad, Stylos, 1995
- [3] Stojaković M., “Verovatnoća, statistika i slučajni procesi”, Novi Sad, Symbol, 2007.
- [4] Gentle, James E., “Random Number Generation and Monte Carlo Methods”, New York, , Springer - Verlag 2005.
- [5] Jevremović, V. i J. Malixić, “Statističke metode u meteorologiji i inženjerstvu”, Beograd, Savezni hidrometeorološki zavod, 2002.
- [6] Mladenović P., “Elementaran uvod u verovatnoću i statistiku”, Beograd, Društvo matematičara Srbije, 1998.
- [7] Stojaković M., “Matematička analiza 2”, Novi Sad, Symbol, 2007.
- [8] Mališić J., “Verovatnoća i matematička statistika”, Beograd, Krug, 1999.
- [9] Simonović V., Uvod u teoriju verovatnoće i matematičku statistiku”, Bor, Građevinska knjiga, 1995.
- [10] Brezinščak M., “Procenjivanje mjerne nesigurnosti”, Beograd, Savezni zavod za mjere i dragocene kovine, 1976.

Kratka biografija:



Marina Bulat rođena je u Apatinu 1988. god. Bečelov rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Metrologija električnih veličina odbranila je 2013.god.

**PODELA GRAFOVA ALGORITMOM SUPERKORENA
PARTITIONING GRAPHS WITH SUPER ROOT ALGORITHM**Marko Milić, *Fakultet tehničkih nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj – U ovom radu je predstavljen algoritam superkorena koji se bavi inicijalnom podelom modela podataka elektroenergetske mreže.

Abstract – This thesis presents super root algorithm for initial dispersion of data model for electrical distribution networks.

Ključne reči: graf, čvor, grana, distributivna mreža, algoritam superkorena – SR, distributivna mreža

1. UVOD

Sa razvojem distributivnih elektroenergetskih sistema, model podataka postaje sve složeniji. Podaci o celokupnom modelu podataka distributivnog elektroenergetskog sistema mogu biti smešteni na jedan računar (centralizovano rešenje) ili na više računara u okviru distribuiranog računarskog sistema (decentralizovano rešenje). U slučaju centralizovanog rešenja, podaci se obrađuju od strane jednog ili više procesora na jednom računaru. Kod decentralizovanog rešenja, podaci smešteni na više računara se obrađuju od strane lokalnih procesora.

Jedna od glavnih mana centralizovanog rešenja jeste što zahteva snažan (skup) hardver. Takođe, mogućnost proširenja distributivne mreže stvara problem u slučaju centralizovanog rešenja, dok je distribuirani računarski sistem skalabilan, nema ograničenja na proširenje mreže. Njegova snaga je određena zbirom snaga pojedinačnih računara.

To dovodi do potrebe za razvijanjem distribuiranih računarskih sistema za upravljanje distributivnom elektroenergetskom mrežom (*engl. Distribution Management System - DMS*). Jedan od glavnih problema kod distribuiranog sistema jeste način podele modela podataka distributivne mreže. Cilj je napraviti podelu modela podataka u kojoj su podaci nezavisni, odnosno da obrada podataka iz jedne grupe ne zahteva informacije dobijene u obradi neke od drugih grupa podataka. Najčešće je to nemoguće, te je potrebno izvršiti podelu sa minimalnom zavisnošću. Nezavisnost podataka je važna jer komunikacija u distribuiranim sistemima može biti „skupa“, odnosno iziskivati duže vreme izvršavanja, čime se stvara kašnjenje u sistemu. Pored toga, potrebno je podatke podeliti tako da opterećenje u svim delovima sistema bude podjednako. Model je potrebno optimalno podeliti pri pokretanju sistema, kao i odrediti postupak dinamičke preraspodele podataka u slučaju promena u sistemu.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Darko Čapko, docent.

**2. MODEL PODATAKA I POSTUPAK PODELE
MODELA PODATAKA**

Kao što je već ranije navedeno, podela modela podataka distributivne mreže se svodi na podelu grafa. U ovom poglavlju opisane su različite vrste grafova kao i način predstavljanja grafova. Takođe, opisan je problem podele podataka kao i algoritmi koji se koriste za rešavanje problema podele grafa.

2.1. Graf i vrste grafa

Grafovi su prirodan i pogodan način za modelovanje nelinearnih relacija. Graf G je uređen par (V, E) gde je V konačan neprazan skup, a E predstavlja binarne relacije elemenata skupa V .

Elementi skupa V se nazivaju **čvorovi**, a elementi skupa E **grane**. U elektroenergetskim sistemima, čvorovi predstavljaju napajane regione (korene), dok grane grafa predstavljaju veze između odgovarajućih korena.

Dva susedna čvora su krajnje tačke svake grane. Grana koja spaja čvor sa samim sobom naziva se petljom. Graf koji nema nijednu petlju naziva se **prostim** grafom. Graf koji ima konačan broj čvorova zove se **konačan** graf. Analogno, graf sa beskonačnim brojem čvorova zove se **beskonačan** graf.

Multigraf je graf kod kog između dva čvora postoji više od jedne grane.

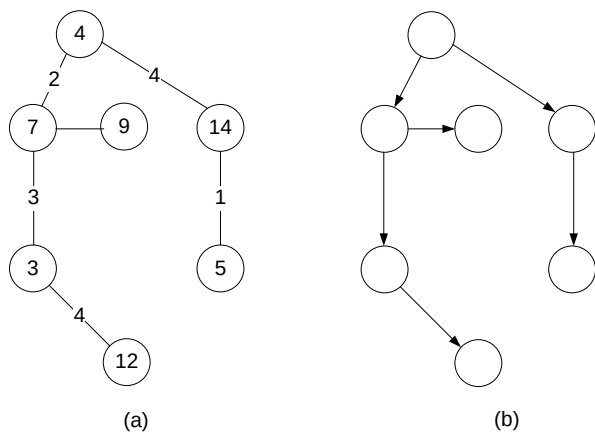
Kompletan graf je onaj graf kod kog su svaka dva čvora povezana granom. Kada je broj grana u grafu relativno velik, u pitanju je **gust** graf, u suprotnom graf je **redak**.

Ne postoji precizno definisana granica za ovo razvrstavanje.

Graf može biti **usmeren** ili **neusmeren**. Ukoliko graf sadrži i usmerene i neusmerene grane, tada se radi o **mešovitom** grafu. Grane u usmerenom grafu su orjentisane, te u odnosu na dati čvor mogu biti ulazne ili izlazne. Čvorovi u i v koji odgovaraju jednoj grani u neusmerenom grafu su međusobno susedni, dok je u usmerenom grafu samo čvor v susedan čvoru u , jer relacija susednosti nije simetrična. Broj grana nekog čvora u koje ga spajaju sa drugim čvorovima grafa je stepen čvora u .

Čvorovima grafa mogu biti pridružene težine i tada je graf **težinski**. U našim istaživanjima težina grane $w(u,v)$ predstavlja broj potencijalnih konekcija između dve oblasti. Težina čvora $w(u)$ zavisi od kompleksnosti proračuna za odgovarajući region.

Na slici 2.1 mogu se videti primeri grafova. Na primeru pod (a) prikazan je usmereni graf, dok je na primeru pod (b) prikazan težinski neusmeren graf.



Slika 2.1 Primeri usmerenog i neusmerenog težinskog grafa

2.2. Predstavljanje grafova

U programskim jezicima, graf se može predstaviti na različite načine. Postoje dva uobičajena načina predstavljanja grafova:

1. Matrica susedstva
2. Lista susedstva

U matricnoj reprezentaciji, graf se predstavlja kao kvadratna matrica čiji članovi predstavljaju grane. Vrednosti člana matrice susednosti pokazuju da li ima grana između dva čvora, čiji brojevi odgovaraju indeksu reda i kolone. Ukoliko je graf težinski, uobičajeno je da vrednost člana matrice predstavlja težinu odgovarajuće grane. Ako je graf neusmeren, matrica je simetrična. Matrica susednosti pogodnija je za guste grafove.

Sa druge strane, lista susednosti pogodna je za retke grafove, jer ona čuva informacije samo o postojećim granama. Obično se organizuju tako da se pamti lista grana za svaki čvor.

Ukoliko je čvor težinski, element liste nosi i informacije o težini odgovarajuće grane. Liste se dodatno mogu uređivati po određenom kriterijumu što dovodi do lakše pretrage. Svaki od pristupa ima svoje prednosti i nedostatke, a njihov zbir se vrši na osnovu potreba sistema. [1]

2.3. Definisanje optimizacionog kriterijuma

U multiprocesorskom okruženju analitičke elektroenergetske funkcije se mogu izvršavati nad različitim regionima (korenima). U slučaju da je broj korena veći od broja procesora, tada se koreni grupišu u p particija i raspoređuju na p procesora. Zadatak podele ukupnog grafa jeste da grupiše čvorove grafa tako da su težine particija približno jednake, i da nijedna particija nema težinu veću od maksimalno dozvoljene težine particije.

Podela podataka može se vršiti inicijalno, pre proračuna, i dinamički, gde se nad podeljenim modelom vrše preraspodele u zavisnosti od dinamike sistema i ulaznih podataka. U ovom radu predstavljeno je rešenje za inicijalnu podelu podataka elektroenergetske mreže.

Da bi definisali optimizacioni kriterijum, za particiju π_k potrebno je definisati težinu particije W_k . Težina particije jednaka je sumi težina svih pripadajućih regiona (korena):

$$W_k = \sum_{R_i \in \pi_k} W_{R_i} \quad (1)$$

Broj veza između particija EC, računa se kao suma broja potencijalnih veza između korena iz različitih particija:

$$EC = \sum_{R_i \in \pi_i, R_j \in \pi_j} Pot_{R_i R_j} \quad (2)$$

EC je potrebno minimizovati podelom.

Težina particija predstavlja meru kompleksnosti proračuna nad podacima unutar particije M:

$$M = (1 + \varepsilon) \frac{1}{p} \sum_{i=1}^p W_i \quad (3)$$

gde je p broj particija, a ε tolerancija. Ako je tolerancija jednaka nuli, tada sve particije moraju biti jednake, što je izuzetno restriktivan zahtev.

Optimizacioni kriterijum može se definisati kao maksimalna povezanost oblasti unutar particije:

$$F = \min(EC) \quad (4)$$

pod uslovom da sve particije ispunjavaju uslov balansiranosti:

$$W_i < M, i \in \{1, 2, \dots, p\} \quad (5)$$

3. ALGORITMI ZA PODELU GRAFA

Neki algoritmi za podelu težinskih grafova:

1. Algoritmi zasnovani na prebacivanju čvorova iz jedne particije u drugu:
 - KL (*Kernighan-Lin*) algoritam
 - FM (*Fiduccia-Mattheyses*) algoritam
2. Evolutivni algoritmi:
 - Genetski algoritam
 - PSO algoritam
3. Višefazni algoritam (*eng. multilevel*)
 - Višefazni algoritam za podelu grafa
 - Algoritam superkorena
4. Hibridni algoritmi
 - Evolutivni PSO (EPSO) algoritam
 - Hibridni genetski algoritam

U ovom radu, na osnovu istraživanja [2], kao algoritam podele izabran je algoritam superkorena koji spada u višefazne algoritme.

3.1. Algoritam Superkorena

Algoritam superkorena (*eng. Super-Roots - RS*) se uglavnom koristi za podelu retkih grafova. Redak graf se odlikuje malim brojem grana (broj grana je manji ili približno jednak broju čvorova).

Ukoliko je moguće podeliti čvorove grafa u dva neprazna skupa R i S, tako da nijedan čvor iz skupa R nije susedan čvoru iz S, takav graf nazivamo nepovezani graf. Povezane čvorove u nepovezanom grafu nazivamo **superčvorovi** ili **superkoreni** (sn) i u tom slučaju možemo nepovezani graf predstaviti kao skup „izolovanih“ superčvorova.

Težina superčvora jednaka je sumi težina pojedinačnih čvorova od kojih se sastoji superčvor. Ako je težina superčvora veća od maksimalno dozvoljene težine particije M (3), tada se taj superčvor proglašava za preveliki superčvor SN_X . Sa SN_X biće označen skup prevelikih superčvorova $SN_X = \{sn | w(sn) > M\}$. [3]

Superčvor čija je težina veća od 50% maksimalne dozvoljene težine particije M naziva se veliki superčvor sn_L . Sa SN_L je označen skup velikih superčvorova $SN_L = \{sn | w(sn) > 0.5M\}$, ($SN_X \subset SN_L$). [3]

Ako je potrebno izvršiti podelu povezanog grafa, dobre osobine SR algoritma ne mogu doći do izražaja i tada se SR algoritam svodi na **METIS** algoritam.

SR algoritam se sastoji iz tri faze: [3]

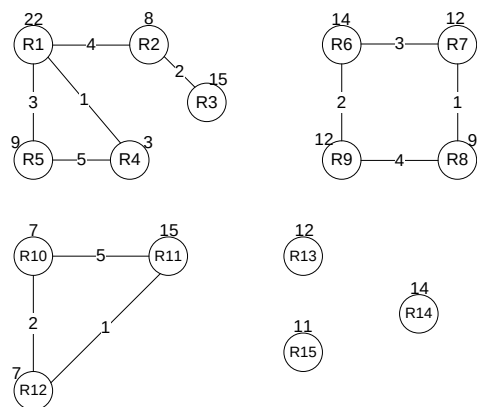
1. Uklupnjavanje – spajaju se svi povezani čvorovi (koreni) u jedan superkoren
2. Podela prevelikih superkorena
3. Raspodela – podela superkorena na zadati broj particija (npr. korišćenjem *Greedy* algoritma za podelu brojeva) [4], [5]

FAZA 1 – Uklupnjavanje

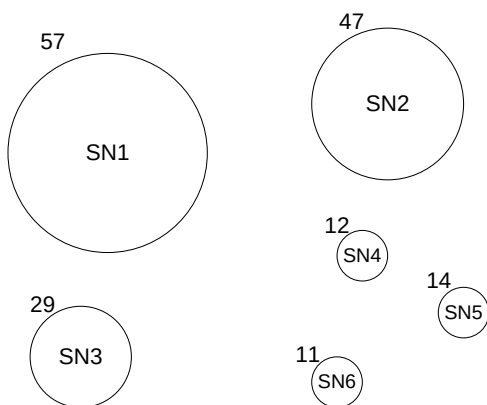
Na osnovu modela podataka distributivne mreže možemo da formiramo graf. Taj graf ćemo koristiti kao ulaz u algoritam. Radi se o težinskom grafu, te su potrebne informacije o težinama korena. Težinu grane određuje broj otvorenih prekidača između dva potencijalno povezana korena. Svaki koren se predstavlja jednim čvorom grafa, a težina čvora predstavlja broj elemenata provodne opreme i transformatora u korenu. Smatra se da taj broj odgovara kompleksnosti proračuna.

U prvoj fazi primene SR algoritma, nad određenim grafom (npr. Slika 3.1) se vrši ukupnjavanje korena. To znači da se sjedinjavaju međusobno povezani koreni koje nazivamo superkoreni.

Dobijamo graf koji je nepovezan (Slika 3.2). Uklupnjeni graf se sastoji od superkorena (SN1...SN6) i takav graf nam je ulaz u fazu 2.



Slika 3.1 Početni graf



Slika 3.2 Faza 1 - Uklupnjavanje početnog grafa

FAZA 2 – Podela prevelikih superkorena

Nakon faze ukupnjavanja, vrši se provera da li je težina nekog od superkorena veća od dozvoljene težine particije M (3). Ukoliko nemamo nijedan superkoren čija je težina veća od dozvoljene težine particije M , prelazi se na fazu 3. Takav je slučaj kada je potrebno podeliti model sa slike

na dve particije. U tom slučaju, težina svakog pojedinačnog superkorena je manja od M .

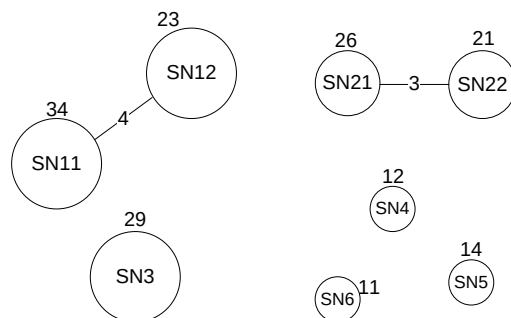
Ukoliko postoje preveliki superkoreni, onda se prelazi na fazu 2 – podela prevelikih superkorena. Superkoreni čija veličina nije veća od maksimalne veličine particije smeštaju se u skup SN_R , dok se superkoreni čija težina prekoračuje dozvoljenu veličinu particije smeštaju u skup SN_{XL} . Minimalan broj delova ($nMin$) na koji se deli preveliki superkoren se određuje po formuli:

$$nMin = \left\lceil \frac{W_{SN}}{M} \right\rceil,$$

a maksimalan broj delova na koji delimo preveliki superkoren je jednak broju particija p na koje treba da se smeste podaci ($nMax = p$). Iz tog razloga, optimalan broj podpodela nije unapred poznat, te se vrši više različitih podpodela. Počinje se sa $nMin$ i povećava se do ukupnog broja particija $nMax = p$. [5]

Nakon toga, potrebno je primeniti neki algoritam za podelu prevelikih superkorena. U ovom radu za te svrhe korišćen je METIS algoritam. METIS algoritam vrši pojedinačnu podelu svakog prevelikog superkorena na više manjih delova koje nazivamo fragmenti superkorena. Broj fragmenata superkorena zavisi od funkcije $F(4)$ koja predstavlja maksimalnu povezanost oblasti (korena) unutar particije. Funkciju F računamo za svaku podelu, i na osnovu toga formiramo niz dobijenih rešenja u opadajućem redosledu. Uvek biramo raspodelu sa najvećom funkcijom F koja zadovoljava kriterijum optimalnosti, odnosno podelu koja neće dovesti do prekoračenja veličine neke od particija u fazi 3.

Podela prevelikih superkorena će biti objašnjena kroz primer podele modela sa slike 4.2 na četiri particije sa tolerancijom $\varepsilon = 0.1$ i dozvoljenom težinom particije $M = 45.4$. U tom slučaju dobijamo dva prevelika superkorena SN1 i SN2 čije su težine 57 odnosno 47. Težina ostalih superkorena je manja od maksimalne težine particije M . Primenom METIS algoritma na prevelike superkorene SN1 i SN2, vrši se podela pomenutih superkorena (slika 3.3):



Slika 3.1 Faza 2 - Podela prevelikih superkorena

Odabrana je podela superkorena na dva dela, jer je funkcija F bila najveća za podelu superkorena na dva dela u oba slučaja, odnosno pri podeli oba prevelika superkorena.

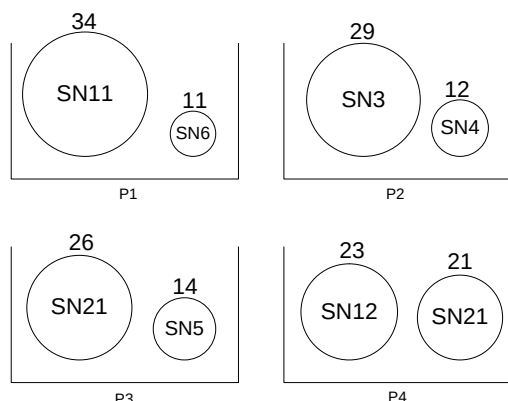
Kada nemamo ni jedan superkoren čija je težina veća od dozvoljene težine particije, možemo da pređemo na fazu 3, odnosno raspodelu.

FAZA 3 – Raspodela

Na početku faze raspodele formira se skup S koji se sastoji od nevelikih superkorena iz skupa SN_R i usitnjenih fragmenata prevelikih superkorena. U prvoj iteraciji

uzimamo usitnjene fragmente sa najvećom funkcijom F . Zatim se vrši podela brojeva na zadati broj particija. U ovom radu je u te svrhe korišćen Greedy algoritam za podelu brojeva. U ovom slučaju, brojevi označavaju veličine superkorena. Ukoliko algoritam za podelu brojeva (Greedy algoritam) nije dao dobar rezultat, odnosno ako skup određenih superkorena prekoračuje dozvoljenu težinu particije, proces se ponavlja, s tim da se uzima sledeća podela fragmenata. Bira se ona raspodela prevelikih superkorena na fragmente, koja će imati najmanje pogoršanje funkcije F , a zadovoljiti kriterijum optimalnosti pri raspodeli brojeva. Ukoliko nakon svega nije moguće dobiti balansirane particije, algoritam vrši podelu velikih superkorena iz skupa SN_L i postupak raspodele se ponavlja.

Primer raspodele brojeva pomoću Greedy algoritma dat je na slici 3.4



Slika 3.4 Faza – Podela nevelikih superkorena na zadati broj particija

4. EKSPERIMENTALNI REZULTATI

Grafovi korišćeni u ovom radu generisani su na slučajan način i to u nekoliko veličina. Pod veličnom grafa se smatra broj čvorova i grana od kojih se sastoji graf. U ovom radu su generisani grafovi čiji je broj čvorova 50, 100, 500, 1000, 2000, 5000 i 10000. Takođe svaki od grafova je generisan sa 3 različite gustine: retki, srednje gusti i gusti grafovi (ovo klasifikacija je napravljena u odnosu na realne gustine grafova koji opisuju distributivne elektroenergetske mreže). Retkim grafovima smatramo one koji imaju oko 50% manje grana u odnosu na broj čvorova. Srednje gusti grafovi imaju približan broj čvorova i grana. Gusti grafovi imaju oko 2 puta veći broj grana u odnosu na broj čvorova.

Prilikom testiranja performansi SR algoritma, vršilo se poređenje sa rezultatima dobijenim primenom METIS algoritma. METIS algoritam je jedan od najčešće korišćenih algoritama za podelu grafa. Poređenje je vršeno u odnosu na maksimalnu povezanost između particija.

Za retke grafove, odnosno grafove koji imaju za oko 50% manje grana u odnosu na broj čvorova, znatno bolji rezultati su dobijeni primenom SR algoritma. Za grafove sa malim brojem čvorova (grafovi sa 50, 100, 500 čvorova) rezultati dobijeni SR algoritmom su bolji ili jednaki u odnosu na rezultate dobijene METIS algoritmom. Za grafove sa većim brojem čvorova (1000,

2000, 5000, 10000) SR algoritam je u svim slučajevima dao bolje rezultate od METIS-a.

Za srednje-guste i guste grafove rezultati su promjenljivi, u nekim slučajevima su bolji rezultati dobijeni SR algoritmom, dok su u nekim slučajevima bolji rezultati dobijeni primenom METIS algoritma.

5. ZAKLJUČAK

Trendovi u razvoju sistema za upravljanje distributivnim mrežama, ali i elektroenergetskim mrežama uopšte, dovode do potrebe razvoja distribuiranih rešenja. U takvim distribuiranom računarskom sistemu potrebno je podeliti podatke, što doprinosi efikasnijem radu distribuiranog sistema. Cilj je da skupovi podataka budu što manje zavisni jedni od drugih i da opterećenje procesora bude približno jednako. Analizom sistema za upravljanje distributivnim mrežama uočeno je da se analitičke elektroenergetske funkcije izvršavaju nad korenima, odnosno grupama elemenata napajanih iz istog izvora. Za takvu reprezentaciju koristimo grafove.

U distribuiranom sistemu možemo paralelno izvršavati analitičke elektroenergetske funkcije. Zbog toga je potrebna procedura za raspodelu korena između procesora/računara. Ovaj problem se svodi na podelu grafa, a rešava se nekim od algoritama za podelu grafa. U ovom radu je implementirano i testirano rešenje za inicijalnu podelu grafa pomoću SR algoritma, odnosno algoritma superkorena.

Na osnovu rezultata testiranja, odnosno poređenjem rezultata dobijenih algoritmom superkorena sa rezultatima dobijenim METIS algoritmom, može se zaključiti da je za podelu retkih grafova bolje koristiti SR algoritam. To se naročito odnosi na grafove sa 1000 i više čvorova.

6. LITERATURA

- [1] Tomašević, M.: Strukture podataka, Elektrotehnički fakultet, Beograd, 2000
- [2] Čapko, D: Optimalna podela velikih modela podataka u okviru nadzorno-upravljačkih elektroenergetskih sistema, Doktorska disertacija, Novi Sad, 2012
- [3] Čapko, D., Erdeljan, A., Popović, M., Svenda, G.: An optimal initial partitioning of large data model in utility management system, advances in electrical and computer engineering, vol 11, no. 4, pp. 41-46, 2011
- [4] Korf, R.E.: A complete anytime algorithm for number partitioning, artificial intelligence, vol. 106, no.2, pp. 181-203, 1988
- [5] Korf, R.E.: Multu-way number partitioning, international joint conference on artifical intelligence, Pasadena, California, US, 2009

Kratka biografija:



Marko Milić je rođen 1988. godine u Zenici. Završio je tehničku školu Mihalo Pupin u Bijeljini 2007. godine. Osnovne studije je završio na Fakultetu Tehničkih nauka u Novom Sadu 2013. godine. Ispunio je sve obaveze i položio sve ispite u predviđenom studijskom programu..

**RAČUNARSKI MERNI SISTEM ZASNOVAN NA PIC MIKROKONTROLERU I
ETHERNET KOMUNIKACIJI****COMPUTER-BASED MEASUREMENT SYSTEM BASED ON PIC
MICROCONTROLLER AND ETHERNET COMMUNICATION**

Miloš Nedić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – MERNI SISTEMI

Kratak sadržaj – U radu je dat opis implementacije rešenja računarskog mernog sistema koji koristi potencijale PIC mikrokontrolera i ethernet komunikacije. Daje se opis hardvera, firmvera i softvera sistema, sa naglaskom na softversko rešenje, koje čini najveći deo sistema. Razlog za to je očigledna prednost u hardverskom kapacitetu računara današnjice nad hardverskim kapacitetom PIC mikrokontrolera u upotrebi.

Abstract – This paper contains the description for computer-based measurement system solution that implements PIC microcontroller and ethernet communication technologies. The description for hardware, firmware and software of the system is presented. The software comprises the largest part of the system, and the reason for that can be found in obvious advantage of PC hardware characteristics over PIC microcontroller in use.

Ključne reči : PIC, mikrokontroler, ethernet, računarski merni sistemi, računarska aplikacija

1. UVOD

Ethernet komunikacija danas zauzima najveći udeo u računarskim komunikacijama. Širom sveta, u svakoj oblasti privrede gde je prisutna potreba za računarskim mrežama i prenosom podataka između računara, prisutna je implementacija ethernet standarda, bilo da je u pitanju žični vid komunikacije, bilo bežični. Jedan od važnijih razloga za popularnost ethernet-a jeste upotreba dve vrste protokola koji daju značaj dvema različitim karakteristikama – garanciji tačnosti prenosa podataka i garanciji brzine prenosa podataka – što predstavlja potpunu pokrivenost zahteva tržišta.

Pored obezbeđivanja neikvarenosti podataka prilikom prenosa, brzine prenosa koje ethernet podržava daju mu prednost nad serijskom UART komunikacijom, često upotrebljavanom u računarskim mernim sistemima.

PIC mikrokontroleri su raznovrsni i pristupačni za mnoge korisnike, iz razloga što postoji opsežna dokumentacija o karakteristikama, ali i primeni PIC mikrokontrolera. Širok dijapazon ovih Microchip-ovih uređaja može biti u upotrebi kao samostalni sistemi ili kao delovi drugih sistema, pri čemu mogu biti u upotrebi kao tajmeri (brojači) analognodigitalni pretvarači, portovi serijske komunikacije i dr.

Upotreba računara u današnjim mernim sistemima je veoma značajna, istakavši pri tome njihovu primenu u akviziciji i obradi mernih podataka. Činjenica da postoje računari malih ukupnih dimenzija, a velikih performansi, koji tako postaju pouzdani prenosivi računari, samo im daje veći značaj i veću primenu u mernim sistemima.

2. CILJ I PREDMET ISTRAŽIVANJA

Primarni predmet istraživanja se odnosi na razvoj računarskog mernog sistema koji koristi PIC mikrokontroler za akviziciju podataka, a ethernet komunikaciju za prenos tih podataka do računara. Računar bi služio kao medijum za obradu i pohranjivanje svih mernih podataka. Sekundarni predmet istraživanja se odnosi na ispitivanje potencijala ovakvog sistema i njegovih delova, kao i utvrđivanje potencijalnog usavršavanja sistema.

Razvijen sistem trebalo bi da u najvećoj mogućoj meri ističe univerzalnost merenja kao svoju najbitniju karakteristiku, što treba da znači da korisniku bude na raspolaganju ekstenzivno podešavanje parametara merenja, poput: opsega mernih veličina, frekvencije odabiranja mernih podataka, broja kanala za uporedno merenje, itd. Potrebno bi bilo razviti firmver sistema, tj. program PIC mikrokontrolera, koji bi unutar sistema služio kao merni pretvarač analogne senzorske informacije u digitalnu informaciju za prenos do računara. Korišćeno bi bilo više kanala analognodigitalnog pretvarača PIC mikrokontrolera za istovremeno uporedno merenje. Razvoj softvera sistema podrazumevao bi razvoj aplikacije za Microsoft Windows platformu, koja bi uključivala: korisnička podešavanja karakteristika merenja, odabiranje mernih podataka slanjem zahteva preko ethernet-a do PIC-a, obradu primljenih mernih podataka, njihov tabelarni i grafički prikaz, kao i automatsko čuvanje u datoteku nekog od popularnijih formata (npr. CSV).

3. RAZVOJ RAČUNARSKOG MERNOG SISTEMA**3.1. Hardver sistema**

Za razvoj hardvera sistema korišćen je razvojni sistem za PIC mikrokontrolere, kompanije Mikroelektronika, dostupan na Katedri za električna merenja na Fakultetu tehničkih nauka. Sistem se sastoji iz razvojne ploče BigPIC 5, kojoj je pridružen PIC 18F8520 mikrokontroler i ethernet kartica Serial Ethernet Board sa ENC28J60 mrežnim kontrolerom.

BigPIC 5 se sastoji iz različitih modula pogodnih za ispitivanje mogućnosti pridruženih PIC mikrokontrolera (LED, Button, RS232, MMC/SD, LCD, GLCD, i dr.), od

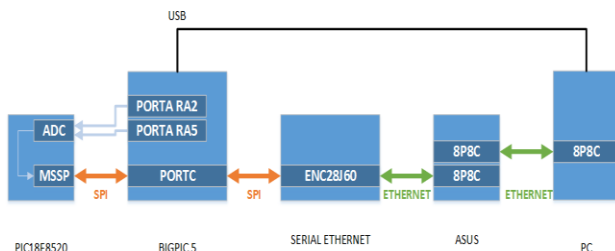
NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Platon Sovilj.

kojih su u upotrebi bili isključivo najpotrebniji – A/D Converter Input i Direct Port Access. Prvi služi za simulaciju analognog signala preko potencijometra i povezan je sa ulazom 2 kanala A/D pretvarača mikrokontrolera. Drugi predstavlja jednostavno elektronsko kolo sa izvodima ulazno-izlaznih portova mikrokontrolera i pull-up/down otpornicima [1].

PIC 18F8520 sadrži A/D pretvarački modul rezolucije od 10 bita i ukupno 16 kanala [2] (u upotrebi su svi kanali, a na korisniku je ostavljeno da odabere sa kojih kanala će softver sistema da čuva podatke). Za komunikaciju sa eksternim elektronskim modulima, poput ethernet kartice, koristi se MSSP (Master Synchronous Serial Port) modul sa serijskom (SPI) komunikacijom. Maksimalan radni takt mikrokontrolera od 40 MHz omogućava SPI brzinu od 10 MHz, što je sasvim dovoljno za široku upotrebu merenja.

Serial Ethernet Board predstavlja SPI-ethernet adapter, pa je samim tim povezan sa SPI komunikacijskim portom mikrokontrolera, a preko RJ45 (8P8C) konektora i UTP kabla sa računarom.



Slika 1. Blok dijagram hardvera sistema.

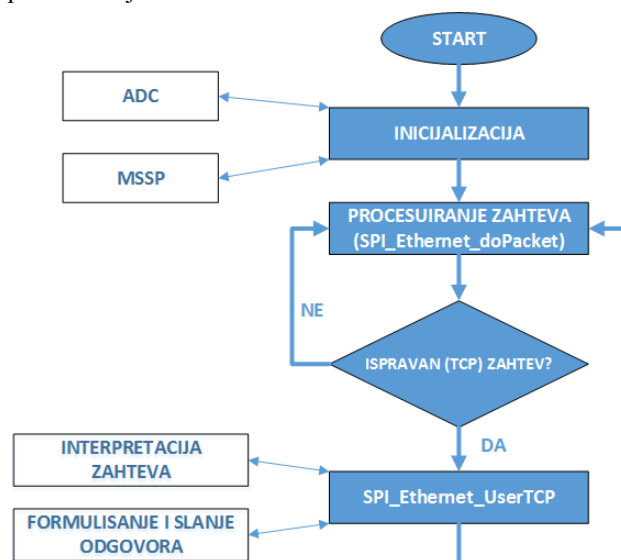
Kada pričamo o povezivanju hardverskih komponenti sistema i toku informacija između tih komponenti (slika 1), trebalo bi da spomenemo i potrebu za tzv. crossover kablom ili svič (eng. switch) uređajem, koji bi omogućili dvosmernu, full duplex komunikaciju između dva ethernet porta (mikrokontrolera i računara). U upotrebi je bio svič ASUS GigaX1005.

3.2. Firmver sistema

Pod razvojem firmvera sistema direktno se smatra razvoj programa mikrokontrolera u okruženju MikroC PRO for PIC kompanije Mikroelektronika. Ovo okruženje ima integrisan C kompajler prilagođen razvoju C koda za arhitekturu PIC mikrokontrolera. Pored kompajlera, koji C kod pretvara u HEX datoteku, u pomenuto okruženje integrisan je i programatorski softver, koji služi za fizičko programiranje mikrokontrolera, tj. prebacivanje HEX datoteke u mikrokontrolersku programsku memoriju. Glavni princip kojim je vođen razvoj firmvera jeste minimizacija funkcionalnosti firmvera na najneophodnije, tako da mikrokontroler bude efektivan posrednik u prenosu mernih podataka sa pridruženih (simuliranih) analognih senzorskih kola ka računaru. S tim u vezi, blok dijagram firmvera je prilično jednostavan, kao što je prikazano na slici 2.

Po učitavanju memorije mikrokontrolera (uključivanju razvojne ploče) izvršava se inicijalizacija A/D pretvaračkog modula i MSSP komunikacijskog modula. U tu svrhu se u znatnoj meri koriste inicijalizacijske funkcije koje su dostupne u respektivnim programskim bibliotekama u MikroC PRO for PIC okruženju. Posle

inicijalizacije, program ulazi u tzv. beskonačnu petlju, gde se proverava, interpretira i šalje odgovor na eventualno pristigli TCP zahtev. Razlog zašto se ovaj deo koda stavlja u beskonačnu petlju, a ne unutar nekog interrupt događaja vezanog za PIC tajmer, je u tome da ovo omogućuje najbržu proveru da li je pristigao zahtev za obradu (što radi SPI_Ethernet_doPacket funkcija u SPI_Ethernet biblioteci [3]), tako da se najbolje osigura da se neki zahtev ne obradi. Obrada TCP zahteva podrazumeva čitanje digitalne vrednosti (odnosno 10-bitnog broja) 16 kanala A/D pretvarača, kao i formiranje tekstualnog predstavljanja tih vrednosti, kako bi se poslali natrag ka računaru i mogli prikazati u računarskoj aplikaciji, ili na pr., u bilo kom programu za internet pretraživanje.



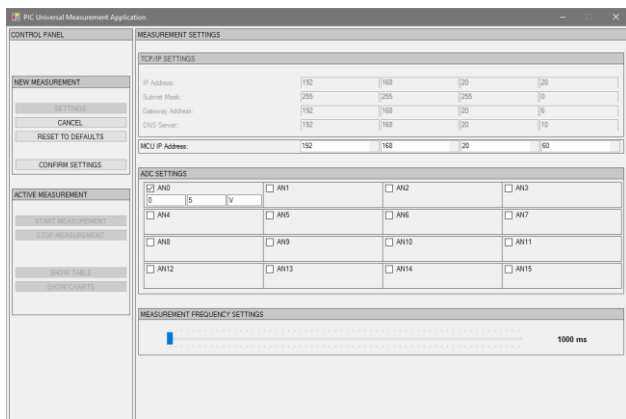
Slika 2. Blok dijagram firmvera sistema.

3.3. Softver sistema

Za razvoj softvera sistema, odnosno računarske aplikacije koja će praktično da kontroliše merenje u okviru računarskog mernog sistema, odabrano je okruženje Microsoft Visual Studio, dok je izvorni kod razvijan u programskom jeziku C# („si šarp“). Pomenuto okruženje i programski jezik su odabrani prvenstveno zbog dostupnosti dokumentacije za kompletan interfejs za programiranje aplikacija (API), tj. za sve upotrebljene programske biblioteke i pridružene klase i funkcije [4]. Razvijena računarska aplikacija je Windows Forms aplikacija i kao takva, oslanja se na .NET Framework (programski okvir) verzije 2.0. Ova verzija, iako nema najmodernije funkcionalnosti i klase, kao što je klasa Chart (za iscrtavanje grafika), ipak ima prednost u primeni na velikom broju uređaja, jer može da se pokreće na starijim Windows operativnim sistemima, zaključno sa Windows XP-om. Nedostatak odabira ove verzije .NET-a je u tome što je jednostavno potrebno više programskog koda razviti.

Važna karakteristika računarske aplikacije predstavlja mogućnost podešavanja merenja, pre pokretanja istog, i to po pitanju podešavanja aktivnih kanala A/D pretvarača mikrokontrolera, opsega i mernih jedinica mernih veličina za aktivne kanale, podešavanja frekvencije odabiranja, kao i IP adrese mikrokontrolera koji treba da bude priključen na istu računarsku mrežu sa računarom. Ovim podešavanjima

se pristupa klikom na dugme *Settings* na kontrolnom panelu (slika 3).



Slika 3. Prikaz računarske aplikacije sa aktivnim panelom za podešavanje parametara merenja.

Po željenom izboru, korisniku je omogućeno da odbaci podešavanja (dugme *Cancel*), da podesi sva podešavanja na njihove početne, podrazumevane vrednosti (dugme *Reset To Defaults*), kao i da potvrdi trenutna podešavanja (dugme *Confirm Settings*). Odabirom poslednje opcije, podešavanja se čuvaju u tekstualnu datoteku, kako bi bili vidljivi za druge komponente aplikacije, tj. za komponentu aktivnog praćenja merenja, koja se takođe aktivira nakon potvrde podešavanja.

Korisniku je tada omogućeno da otpočne merenje (dugme *Start Measurement*), čime se automatski pokreću tajmeri računarske aplikacije, vezani za frekvenciju odabiranja i frekvenciju osvežavanja grafika. Kada tajmer vezan za frekvenciju odabiranja „otkuca“, poziva se specijalna funkcija (tzv. *EventHandler* [5]), unutar koje se vrši slanje TCP/IP zahteva za merenjem ka mikrokontroleru.

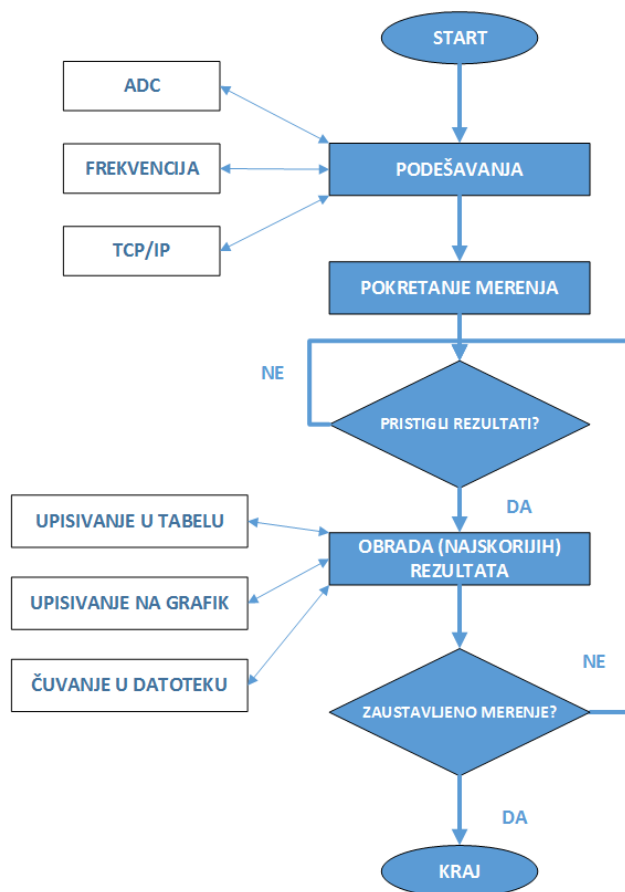
Kada pristigne odgovor poziva se druga funkcija (takođe *EventHandler*), unutar koje se vrši obrada pristiglih rezultata i njihov prikaz u tabeli, a potom i na grafiku. Ovde je takođe omogućeno automatsko čuvanje mernih podataka u CSV datoteku.

Ta datoteka se kreira prilikom svakog novog pokretanja merenja i to pod nazivom koji sadrži trenutni datum i vreme, kako ne bi došlo do neželjenog prepisivanja datoteke.

U toku rada tajmera za odabiranje merenja, ali i po završetku merenja, korisniku je omogućeno da kontinualno menja prikaz rezultata merenja između tabelarnog (dugme *Show Table*) i grafičkog (dugme *Show Charts*). Tabela prikaz sadrži informacije o merenju sa svih aktivnih mernih kanala u odnosu na vreme odabiranja (preciznije, tačno vreme odabiranja).

Grafički prikaz sadrži identične informacije, tj. promenu intenziteta određene merne veličine u vremenu. Grafički prikaz se ažurira odvojeno od ažuriranja mernih podataka, frekvencijom nešto većom od dve sekunde. Takođe, grafički prikaz za pojedinačni kanal može da se uveća (dugme *Zoom*) radi vidljivijeg pregleda.

Ukoliko se želi prekinuti merenje, to se može uraditi klikom na dugme *Stop Measurement*, nakon čega je, ukoliko se želi pokrenuti novo merenje, potrebno ponovo proći kroz proceduru podešavanja merenja. Na slici 4 nalazi se uprošćen blok dijagram rada računarske aplikacije.



Slika 4. Blok dijagram softvera sistema.

4. ISPITIVANJE SISTEMA

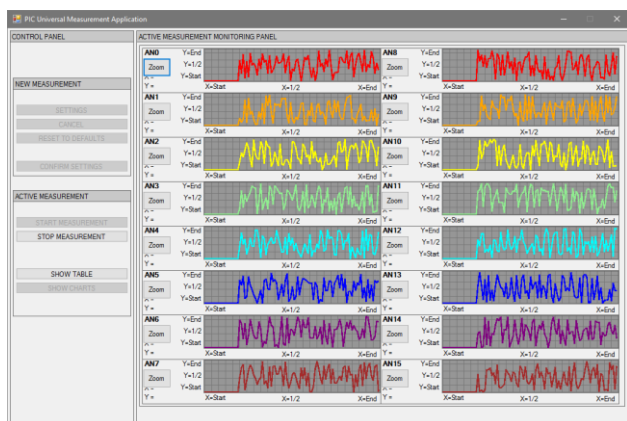
Ispitivanje računarskog mernog sistema izvršeno je na dva načina. U oba slučaja hardver sistema nije sadržao senzorska elektronska kola na ulazu A/D pretvarača mikrokontrolera, već su bila u upotrebi potencijometrijska kola na razvojnoj ploči BigPIC 5.

Takođe, u oba slučaja postoji preduslov podešavanja računarskog *ethernet* adaptera, odnosno, podešavanja TCP/IPv4 protokola (IP adrese, *Subnet* maske, *Gateway* i DNS adrese). Na taj način praktično se dodeljuju statičke adrese dvema članicama lokalne računarske mreže (računaru i mikrokontroleru).

U prvom ispitnom slučaju, koji se može smatrati isključivo simulacijom sistema, veza između firmvera i softvera sistema nije postojala, pa je akviziciju mernih podataka podržala upotreba instance *Random* klase *.NET Framework*-a 2.0.

Drugim rečima, korišćeno je slučajno odabiranje brojeva u granicama definisanim u firmveru sistema, od 0 do 1023 (što predstavlja maksimalnu vrednost 10-bitskog računarskog zapisa). Simulacijski režim rada računarske aplikacije je bio u upotrebi ranije u toku njenog razvoja, odnosno, praktično do samog kraja razvoja, jer je funkcionalnost komunikacije firmvera i softvera poslednja dodata aplikaciji.

Karakteristike drugog slučaja ispitivanja su u najvećoj meri objašnjene u delu opisa softvera sistema, pa ne postoji potreba za dodatnim informacijama. Prikaz rada računarske aplikacije u simulacijskom režimu dat je na slici 5.



Slika 5. Prikaz rada softvera sistema koristeći slučajan odabir mernih podataka.

Biće navedena ukratko i metrika koja je ispraćena prilikom rada računarske aplikacije, a odnosi se na korišćenje hardverskih resursa računara (procenat korišćenja CPU-a i RAM memorije).

Pri pokretanju i neaktivnom radu, tj. van režima merenja, aplikacija je zauzimala približno 15% CPU resursa i približno 14 MB RAM-a.

Nakon približno 1200 odbiraka merenja (sa svih 16 kanala A/D pretvarača mikrokontrolera) primetno je uvećanje upotrebe RAM memorije za otprilike 6 MB, što ne predstavlja nedostatak za moderne računare koji imaju dostupne RAM kapacitete reda nekoliko (desetina) GB.

7. ZAKLJUČNA RAZMATRANJA

Razvijen računarski merni sistem zasnovan na PIC mikrokontroleru i *ethernet* komunikaciji predstavlja pionirski poduhvat da se funkcionalnost razvojnog hardverskog sistema BigPIC 5 sa PIC mikrokontrolerom inkorporiše sa modernim mogućnostima računara.

Iako je uloga mikrokontrolera u ovom sistemu znatno manja od uloge razvijene računarske aplikacije, to ne umanjuje potrebu da se on nalazi u sastavu sistema.

Naprosto, teško je zameniti paket funkcionalnosti koje mikrokontroler nudi ovom sistemu, počev od višekanalnog merenja i prenosa istog serijskom komunikacijom do računara.

Baš ta mogućnost višekanalnog merenja imala je važan uticaj na dalji razvoj sistema, pa se pri razvoju naglasak prebacio ka realizaciji aplikacije koja bi podržala univerzalno merenje i tako dala više mogućnosti korisniku da upotrebi aplikaciju na željeni način i u željenu svrhu.

Sistem uspešno upotrebljava sve neophodne komponente hardvera, firmvera i softvera, opisane u ovom radu, omogućavajući korisnicima 16-kanalno merenje sa rezolucijom od 10 bita, ali i mogućnost da se svojeručno podesi opseg merenja, menjajući tako i efektivnu rezoluciju merenja. Rezultati se automatski ažuriraju za tabelarni i grafički prikaz, a istovremeno se i pohranjuju u datoteku, takođe tabelarnog prikaza.

Prilike za unapređenje sistema svakako postoje. Iako sistem teži da omogući univerzalno merenje različitih fizičkih veličina Potencijalan povod za dodatno istraživanje može da bude u padu performansi na određenoj vrsti konfiguracije računara i to pod uslovima frekvencije odabiranja merenja koja je manja od 500 ms. Kao rezultat toga, aplikacija je u nekoj meri ograničena na frekvencije odabiranja veće od jedne sekunde.

U svom trenutnom razvojnem stadijumu, ovaj sistem može da se primeni za neka osnovna merenja više signala, njihovo praćenje u realnom vremenu i naknadnu analizu, ali isto tako može da posluži kao nastavno sredstvo za studente i entuzijaste iz oblasti merenja u realnom vremenu, sa verovanjem da će im pomenuti predlozi za unapređenje sistema predstavljati vredan, ali ne i nedostižan izazov.

8. LITERATURA

- [1] Mikroelektronika (2008), BigPIC 5 User's Manual, http://www.mikroe.com/pdf/bigpic5/bigpic5_manual.pdf
- [2] Microchip Technology Inc. (2004), PIC18F6520/8520/6620/8620/6720/8720 Data Sheet, <http://ww1.microchip.com/downloads/en/devicedoc/39609b.pdf>
- [3] SPI Ethernet Library, http://download.mikroe.com/documents/compilers/mikroc/pic/help/spi_ethernet_library.htm?TB_iframe=true&height=740&width=970 (16.09.2016.)
- [4] C# Programming Guide, <https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx> (16.09.2016.)
- [5] Handling and Raising Events, [https://msdn.microsoft.com/en-us/library/edzhd2t\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/edzhd2t(v=vs.110).aspx) (16.09.2016.)

Kratka biografija



Miloš Nedić, rođen je 15.09.1988. godine u Doboju, Bosna i Hercegovina, odbranio je master rad iz oblasti Mernih sistema, 2016. godine.

IMPLEMENTACIJA IMPORTA ŠEME RELACIONE BAZE PODATAKA U OKVIRU KROKI ALATA**IMPLEMENTATION OF RELATIONAL DATABASE SCHEMA IMPORT WITHIN KROKI TOOL**Filip Fesl, *Fakultet tehničkih nauka, Novi Sad***Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

Kratak sadržaj – U ovom radu prikazan je način kreiranja KROKI projekta od date metašeme baze podataka. U implementaciji korišten je programski jezik JAVA, JDBC API i postojeće metode KROKI projekta.

Abstract – This paper presents an example of creating a KROKI project out of a given database metaschema. For the implementation, programming language JAVA was used coupled with JDBC API and existing methods of KROKI project.

Ključne reči: Standardna Forma, JDBC, Metašema baze

1. UVOD

U okviru ovog rada će biti prikazane i objašnjene tehnike i tehnologije pomoću kojih je proširen KROKI alat za skiciranje poslovnih aplikacija sledećim funkcionalnostima: ostvarivanjem JDBC (Java Database Connectivity) konekcije ka proizvoljnoj bazi podataka, učitavanjem metašeme baze podataka na koju je alat povezan u memoriju, i mapiranjem učitane metašeme baze podataka na forme KROKI projekta.

Podešavanjem KROKI projekta da radi sa proizvoljnom bazom podataka daje projektantu više opcija koje tehnologije i DBM (Database Management) sisteme može da koristi. Učitavanjem metašeme postojeće baze podataka, i njeno prebacivanje u KROKI projekat daje projektantu mogućnost da iskoristi postojeći model baze podataka i da od njega na lak način kreira skicu poslovnog sistema, a samim tim i izgeneriše poslovnu aplikaciju. Pored ovoga, predviđena je i mogućnost konektovanja izgenerisane aplikacije na istu bazu iz koje je generisan projekat da bi se rad mogao nastaviti sa već postojećim skupom podataka, bez njegovog narušavanja.

2. OPIS DELOVA SISTEMA

JDBC (Java Database Connectivity) predstavlja API (Application Programming Interface) za programski jezik Java, koji definiše na koji način klijent može da pristupi bazi podataka. Deo je JavaSE (Java Standard Edition) platforme, pisan je za relacione baze podataka i pruža metode za upite nad podacima baze, kao i njihovo manipulisanje [1].

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević.

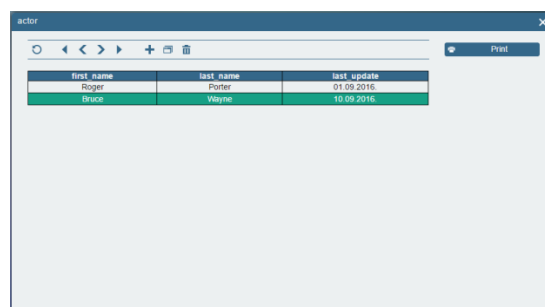
Pre ostvarivanja pristupa bazi podataka neophodno je uraditi dve stvari: učitati neophodni driver, tj njegovu klasu pomoću koje će ostvariti konekcija, i naravno ostvariti konekciju putem JDBC konekcionog stringa.

Skoro svi DBMS vendori pružaju .jar datoteke u kojima se nalaze drajveri za konektovanje na njihove baze, kao i opširna uputstva kako se oni grade. Pored .jar datoteka koje DBMS vendori daju na korišćenje postoji i mnoštvo javno dostupnih open source projekata koji pružaju slične drajvere za konekciju na mnoštvo različitih baza.

Metapodaci predstavljaju opisne podatke o nekim podacima. U smislu baza podataka, metapodaci predstavljaju opis same šeme baze, tj. kako su organizovane tabele, koji su njihovi nazivi, koje kolone tabele sadrže, kojeg su tipa itd. Svaka relaciona baza podataka ima neki svoj interni mehanizam tj. način na koji pohranjuje ove podatke.

Skoro svi JDBC drajveri pružaju mogućnost čitanja ovih podataka iz metašeme baze, što nam daje mogućnost analiziranja ovih podataka da bi dobili bolji uvid u strukturu, tj. organizaciju podataka u sklopu neke baze

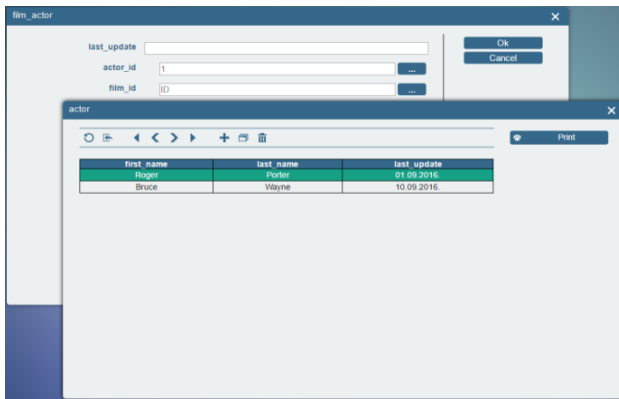
Analizom učitane metašeme podataka se kreiraju standardne forme u KROKI projektu, i to za svaku tabelu baze po jedna forma. Standardne forme su dizajnirane tako da na pregledan način prikazuju podatke i moguće operacije nad podacima. Zajedničke operacije nad entitetima se nalaze u toolbar-u, tj. paleti sa alatnama, a specifične operacije se nalaze na desnoj strani forme. Primer jedne standardne forme web aplikacije je dat na slici 2.1.



first_name	last_name	last_update
Roger	Porter	01.09.2016
Bruce	Wayne	10.09.2016

Slika 2.1. Izgled izgenerisane standardne forme

Povezane forme implementiraju Zoom mehanizam: ako su forme povezane asocijacijom, zoom mehanizam omogućava aktiviranje povezane forme tako da korisnik može sa drugog entiteta da prenese vrednost na trenutno aktivnu formu. Na slici 2.2 se može videti primer otvaranja forme „actor“ putem zoom mehanizma sa forme „film_actor“.



Slika 2.2 Primer izgleda Zoom mehanizma

3. PROJEKTOVANJE I IMPLEMENTACIJA

U postojećoj implementaciji novokreirani KROKI projekat je inicijalno podešen da ostvaruje vezu sa H2 bazom podataka uključenom u alat. U opcijama menija Projects postoji opcija podešavanja konekcije ka proizvoljnoj bazi - "Set database settings" koja prikazuje dijalog sa opcijama. U zavisnosti od odabranih opcija, projekat će biti iskonfigurisan da ostvari konekciju ka ponuđenim bazama podataka.

Unutar istog ovog menija je dodata nova opcija „Set database settings (JDBC)“ koja funkcioniše na sličan način, ali omogućava korisniku da podesi projekat da se konektuje na bilo koju proizvoljnu bazu podataka, ako ima neophodni drajver za konekciju.

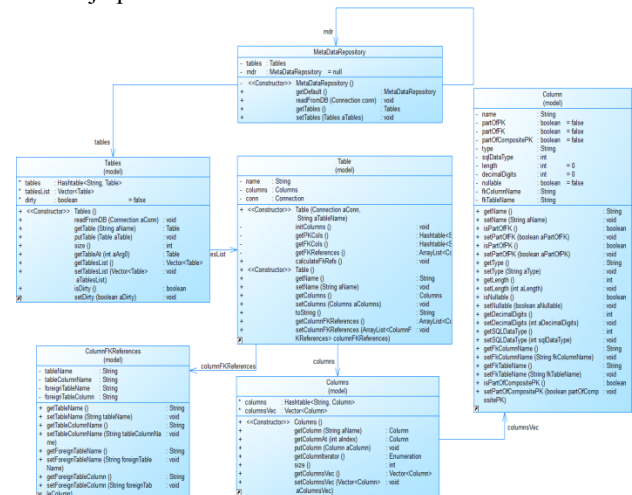
Prilikom testiranja konekcije, sistem prvo učitava .jar datoteku u memoriju putem klase JarClassLoader, pozivajući njenu metodu loadClass. Povratna vrednost ove metode će biti zatražena klasa drajvera koju onda koristimo da ostvarimo konekciju ka bazi.

Unutar klase DatabaseConfigGenerator se radi generisanje hibernate.cfg.xml i persistence.xml datoteka koristeći Freemarker [2] šablone. Prilikom generisanja ovih datoteka je potrebno postaviti parametre sa kojima će biti pozvan šablonski dokument .ftl kojim će Freemarker Java Template Engine izgenerisati date datoteke. Na osnovu vrednosti polja jdbcURL znamo da li generator treba da koristi stari način funkcionisanja (u kome gradi JDBC konekcioni string iz prosleđenih parametara) ili samo da preuzme već prosleđeni.

Izborom akcije menija File->Import->Import from database se pokreće dijalog za odabir parametara konekcije baze na koju se kačimo. Ovo predstavlja prvi korak u kreiranju KROKI projekta iz metašeme baze. Sledeći korak predstavlja podešavanje konfiguracije importa, a zatim analiza pročitanih podataka metašeme i kreiranje KROKI projekta. Klasa koja odgovara ovoj akciji je ImportFromDatabaseAction. U njoj se u metodi se instanciraju, prikazuju dijalozi za parametre konekcije i konfiguraciju import operacije, i pri njihovom zatvaranju preuzimaju podaci, koji se kasnije koriste pri pozivu klase koja će uraditi analizu učitanih podataka i kreirati projekat.

Pozivom singleton klase MetadataRepository sistem putem prosleđenog objekta konekcije učitava sve

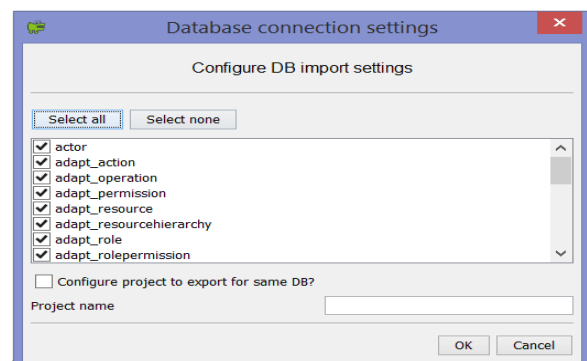
neophodne podatke o metašemi baze. Na slici 3.1 se može videti model ovog dela sistema koji je zaslužan za čitanje i čuvanje podataka o metašemi baze.



Slika 3.1 Model klasa za čuvanje podataka o metašemi

U prvom prolazu analize, kada dobavljamo podatke o kolonama, lako nam je da zaključimo da li je kolona strani ključ. Međutim metoda koja dobavlja te podatke ne nosi informacije o tome koji primarni ključ ovaj strani ključ referencira. Metoda getFKReferences na nivou celog modela metašeme dobavlja ove podatke koristeći metodu getExportedKeys. Kada se odradi prvi prolaz, za svaku tabelu se poziva calculateFKRefs metoda koja iterira po kolekciji tabela i po kolekciji datih referenci stranih ključeva i poredi da li je primarni ključ date tabele referenciran nekim stranim ključem i u slučaju da jeste, ovu informaciju smešta na nivou same kolone da bi nam kasnije pri prolasku kroz model kada radimo njegovo učitavanje u KROKI projekat bilo lakše da dobavimo i iskoristimo sve potrebne podatke.

Nakon završetka učitavanja metašeme baze u model, sistem koristi učitane podatke da iskonfiguriše DBImportSettingsDialog. Izgled ovog dijaloga je dat na slici broj 3.2



Slika 3.2 Izgled dijaloga za podešavanja parametara „Import from database“ operacije

Na ovom dijalogu se unosi željeno ime novog projekta, biraju se tabele koje će biti obrađene, i opciono se može odabrati da novogenerisani projekat bude iskonfigurisan da radi na istoj bazi iz koje je izgenerisan (opcija „Configure project to export for same DB?“). Ako je ova opcija odabrana, projekat se generiše tako da zadrži iste

nazive kolona i tabela baze u EJB (Entity Java Bean) klasama koje će se koristiti u projektu, da bi web aplikacija mogla, ako je tako podešeno, da radi i na inicijalnoj bazi.

Uneti parametri datog dijaloga se dalje prosleđuju konstruktoru klase `ImportDBMetaToProject`. Ova klasa nasleđuje klasu `ProgressWorker`, koja se u sistemu koristi za operacije čije izvršavanje obično traje dugo. Glavna metoda ove klase, koja započinje mapiranje metašeme na KROKI projekat se zove `createKroki`, njen kod se može videti u listingu 3.1. Osnovni dijagram aktivnosti mapiranja metašeme na KROKI projekat se može videti na slici 3.3.

```
private void createKroki() {
    project=new BussinesSubsystem(projectName,
    true, ComponentType.MENU, null);

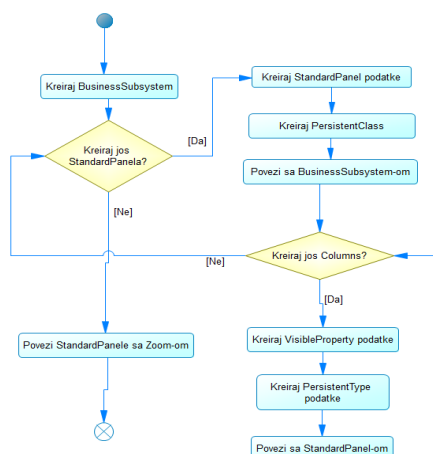
    project.setParseForOriginalDB(parseForOriginalDb
    );

    addIndentation();
    publishText("Project with name
    "+project.getLabel()+" created");
    addIndentation();
    publishText("Extracting project contents");

    for(String tableName : tablesToParse) {
        Table dbTable =
        tables.getTable(tableName);
        panels.put(tableName,
        createPanel(dbTable, project)); //create
        panel from table
    }

    for(String tableName : tablesToParse) {
        Table dbTable =
        tables.getTable(tableName);
        parseZoomForTable(dbTable,
        panels.get(tableName));
    }
}
```

Listing 3.1 Metoda `createKroki` klase `ImportDBMetaToProject`



Slika 3.3 Osnovni dijagram aktivnosti mapiranja metašeme

Metoda `createKroki` izvršava obradu u dve faze. U prvom koraku se kreira `BusinessSubsystem` klasa, iterira se po modelu metašeme i za svaku tabelu se kreira instanca standardnog panela, dok se u drugom koraku prolazi kroz isti model metašeme i kreirane standardne panele te se oni povezuju zoom komponentama.

Za kreiranje svih potrebnih elementa KROKI projekta su iskorišteni postojeći konstruktori i metode klase `KROKI` modela, kao i neke API metode. `BusinessSubsystem` klasa se može posmatrati kao korenski deo KROKI modela, u njoj se posle obrade podataka metašeme nalaze svi objekti jednog KROKI projekta.

Metoda `createPanel` odrađuje posao generisanja standardnog panela na osnovu prosleđenih informacija o tabeli baze podataka. Unutar ove metode se postavljaju osnovne informacije o samom panelu, kao što su ime panela, labela panela i osnovne informacije o izgledu panela. `StandardPanel` klasa pored ovih informacija, u svojoj `PersistentClass` klasi nosi i informacije na osnovu kojih će se generisati EJB klase web aplikacije – tj. podatke o tome koju tačno tabelu baze EJB klasa referencira. Svaki novokreirani standardni panel se smešta u `BusinessSubsystem` objekat, putem njegove metode `setUmlPackage`, čime taj panel postaje povezan sa projektom. Poslednji korak u kreiranju jednog standardnog panela jeste analiza podataka o kolonama prosleđene tabele baze, tj. kreiranje i postavljanje komponenti na panel koje odgovaraju kolonama baze, da bi sistem mogao da odrađuje CRUD (Create, update, delete) operacije nad datim entitetom.

JDBC API definiše set generičkih identifikatora SQL tipova, koji se mogu naći u `java.sql.Types` klasi. Prilikom učitavanja podataka o metašemi baze, u koraku obrade kolona, dobavljena je i informacija o SQL tipu podatka (`rst.getInt("DATA_TYPE")`) za svaku kolonu. Ove bročane vrednosti odgovaraju vrednostima enumeracije `java.sql.Types` klase, tako da nam je lako da odredimo koji java tip podatka i perzistentni tip odgovara koloni koju trenutno obrađujemo. U JavaSE dokumentaciji je opisan opšti slučaj mapiranja vrednosti `java.sql.Types` na Java tipove podataka, koji je iskorišten kao referenca pri mapiranju tipova podataka `VisibleProperty`-a u okviru rada. [3]

Nakon kreiranja svih standardnih panela, sistem prelazi na njihovo povezivanje, tj. kreiranje zoom komponenti za sve tabele baze koje imaju strani ključ. Ako je pronađen panel koji odgovara tabeli koju strani ključ referencira, za njega se kreira zoom komponenta od novog `VisibleProperty` objekta. Da bi ostvarili vezu između ova dva panela, potrebno je da zoom komponenti postavimo aktivacioni panel (odgovara panelu koji trenutno obrađujemo) i ciljni panel (panel koji odgovara referentnoj tabeli stranog ključa). Kada je završeno povezivanje standardnih panela, novi KROKI projekat je spreman za korišćenje, i iz njega se može izgenerisati web aplikacija.

Da bi izgenerisana web aplikacija mogla da radi sa bazom čija je konekcija postavljena kao JDBC konekcioni string, neophodno nam je da se pridruženi JAR ove konekcije nalazi na putanji (classpath) aplikacije. Generator sam po

U standardnom režimu rada generator svakoj EJB klasi koju kreira automatski dodaje polje za jedinstveni identifikator. Međutim, ako se projekat generiše tako da bude u stanju da radi sa originalnom bazom, ovo polje nam predstavlja problem, pošto nema garancije da će se u tabeli originalne baze naći polje opisano na sledeći način:

```
protected java.lang.Long id;
```

U slučaju kreiranja zoom polja EJB klase, pri standardnom generisanju uvek znamo da će na ciljnoj tabeli postojati polje ID koje će predstavljati identifikator entiteta. Međutim, ako generišemo aplikaciju za originalnu bazu, `referencedColumnName` unutar `JoinColumn` anotacije se mora poklapati sa originalnim nazivom identifikatora ciljne tabele, pa zbog toga taj podatak moramo dobiti i postaviti na anotaciju.

4. PRIKAZ GENERISANIH TABELA I FORMI

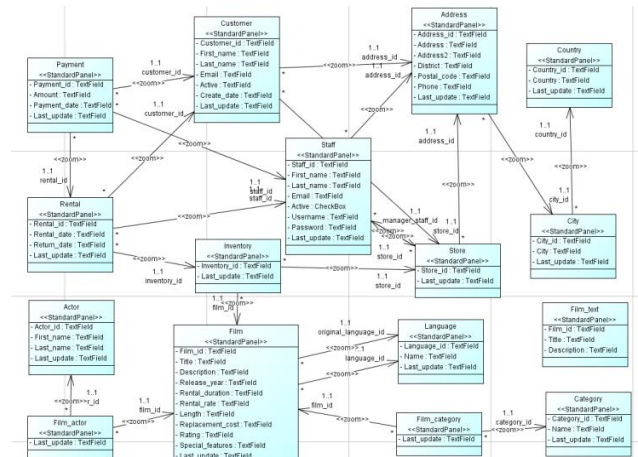
```

classDiagram
    class Payment {
        <<standard>>>
        Payment_id PK
        Amount
        Payment_date
        Last_update
    }
    class Customer {
        <<standard>>>
        Customer_id PK
        First_name
        Last_name
        Email
        Address
        Phone
        Create_date
        Last_update
    }
    class Address {
        <<standard>>>
        Address_id PK
        Address1
        Address2
        District
        Phone
        Create_date
        Last_update
    }
    class County {
        <<standard>>>
        County_id PK
        County1
        Last_update
    }
    class Supplier {
        <<standard>>>
        Supplier_id PK
        Supplier_name
        Contact_name
        Contact_title
        Address
        Phone
        Fax
        E_mail
        Create_date
        Last_update
    }
    class Ship {
        <<standard>>>
        Ship_id PK
        Ship_name
        Last_update
    }
    class Actor {
        <<standard>>>
        Actor_id PK
        First_name
        Last_name
        Email
        Password
        Last_update
    }
    class Inventory {
        <<standard>>>
        Inventory_id PK
        Last_update
    }
    class Film_actor {
        <<standard>>>
        Film_actor_id PK
        Last_update
    }
    class Film {
        <<standard>>>
        Film_id PK
        Title
        Description
        Release_year
        Rental_duration
        Rental_rate
        Length
        Replacement_cost
        Rating
        Special_features
        Last_update
    }
    class Language {
        <<standard>>>
        Language_id PK
        Name
        Last_update
    }
    class Film_language {
        <<standard>>>
        Film_language_id PK
        original_language_id
        Last_update
    }
    class Film_soul {
        <<standard>>>
        Film_soul_id PK
        Description
    }
    class Category {
        <<standard>>>
        Category_id PK
        Category_name
        Last_update
    }

    Payment "1" -- "1" Customer : payment_customer
    Customer "1" -- "1" Address : customer_address
    Address "1" -- "1" County : address_county
    Supplier "1" -- "1" Ship : supplier_ship
    Actor "1" -- "1" Inventory : actor_inventory
    Actor "1" -- "1" Film_actor : actor_film_actor
    Film "1" -- "1" Film_actor : film_film_actor
    Film "1" -- "1" Language : film_language
    Film "1" -- "1" Film_language : film_film_language
    Film "1" -- "1" Film_soul : film_film_soul
    Film "1" -- "1" Category : film_category

```

Kada se kreira projekat iz iste („sakila“) baze podataka, ali se odabere opcija generisanja sistema za originalnu bazu, i dozvoli se create drop šeme pri inicijalnom pokretanju, sistem generiše tabele istog naziva, sa istim nazivima kolona i poštuje postojeće primarne i strane ključeve originalne baze. Na slici 4.2 je dat UML dijagram ovako izgenerisanog sistema, da bi se videlo da su tabele, tj. entiteti ostali u istim međusobnim vezama.



5. ZAKLJUČAK

Opcija generisanja projekta na osnovu postojeće šeme baze podataka ubrzava proces reverznog inženjeringa u velikoj meri, jer sada projektant može u par lakih koraka doći do UML modela i potrebne skice postojećeg sistema.

6. LITERATURA

- [1] Java Database Connectivity
https://en.wikipedia.org/wiki/Java_Database_Connectivity
- [2] Freemarker Java Template Engine
<http://freemarker.org/>
- [3] Mapping SQL and Java Types
<http://docs.oracle.com/javase/1.5.0/docs/guide/jdbc/getstart/mapping.html>
- [4] Hibernate OGM
<http://hibernate.org/ogm/>

Filip Fesl je rođen 01.10.1990. godine u Odžacima. Školske 2011/2012. godine upisao se na smer Računarske nauke i informatika. Osnovne akademske studije je završio 2015, i iste godine upisao master studije

**IMPLEMENTACIJA ALGORITMA SINTEZE KORIŠĆENJEM PRISTUPA RAZVOJU
SOFTVERA ZASNOVANOM NA MODELIMA****AN IMPLEMENTATION OF THE SYNTHESIS ALGORITHM BASED ON THE MODEL
DRIVEN SOFTWARE**

Mirjana Laušev, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu razvijen je alat koji služi za projektovanje šeme baze podataka na osnovu šeme univerzalne relacije. Projektovanje baze podataka je vršeno algoritmom sinteze koji je implementiran korišćenjem pristupa razvoju softvera zasnovanom na modelima. Funkcionisanje alata, kao i krajnji rezultat prikazani su na primeru studentske službe. Rezultat na primeru studentske službe je implementaciona skripta za bazu podataka studentske službe, kao i prikaz koraka algoritma sinteze u vidu HTML stranice.

Abstract – In this paper we present the development of a software that is used for designing a database schema. Database design is done using synthesis algorithm and implemented using Model Driven Development. Usage of the software, as well as its results, are shown on the example of student services. The result of the example is an implementation script for creating a database for student service and also preview of the steps of the synthesis algorithm in the form of an HTML page.

Ključne reči: Namenski jezici, razvoj softvera zasnovanog na modelima, informacioni sistemi

1. UVOD

Informacioni sistemi (IS) bazirani na računarima služe za prikupljanje, obradu, skladištenje i prikazivanje podataka. Jedan od ključnih zadataka pri kreiranju IS predstavlja izbor pristupa razvoju IS. U ovom radu korišćen je pristup razvoju IS zasnovan na modelima.

Kako je količina podataka kod savremenih IS uglavnom velika, pri razvijanju IS treba obratiti pažnju na projektovanje šeme baze podataka. Baza podataka predstavlja strukturu međusobno povezanih podataka, organizovanu nad šemom baze podataka. Šema baze podataka predstavlja opis strukture baze podataka sa svim svojim ograničenjima. Od same šeme baze podataka zavisi dalji razvoj i implementacija sistema, a loše isprojektovana šema baze podataka može imati ozbiljne posledice po dalji razvoj sistema.

Projektovanje šeme baze podataka se vrši dovođenjem polaznog skupa šema relacija u željenu normalnu formu postupkom normalizacije. Normalizacija predstavlja postupak otklanjanja anomalija ažuriranja dovođenjem šeme baze podataka u određenu normalnu formu.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Ivan Luković, red. prof.

Do određene normalne forme se može doći korišćenjem jedne od dve metode, a to su metoda sinteze i metoda dekompozicije. U ovom radu biće predstavljena implementacija metode sinteze kao odabrane metode za projektovanje šeme baze podataka.

Cilj ovog rada predstavlja automatizacija procesa projektovanja šeme baze podataka. Taj cilj je ostvaren razvojem alata koji se koristi za prevođenje šeme univerzalne relacije u šemu baze podataka. Šema univerzalne relacije predstavlja skup svih obeležja i veza između tih obeležja koje se nazivaju funkcionalne zavisnosti (FZ). Ovaj alat će olakšati proces projektovanja šeme baze podataka i smanjiti rizik koji sa sobom nosi neispravno isprojektovana baza podataka.

2. OPIS KORIŠĆENIH TEHNOLOGIJA

Za razvoj softverskog rešenja izabran je pristup razvoju IS zasnovan na modelima. Kako je za ovaj pristup neophodan razvoj namenskog jezika, prvi korak predstavlja kreiranje meta-modela. Meta-model predstavlja apstraktnu sintaksu namenskog jezika i kreira se korišćenjem koncepata meta-meta-modela odnosno modela za specificiranje meta-modela. Eclipse Modeling Framework (EMF) predstavlja radni okvir (eng. framework) za modelovanje i generisanje koda koji se bazira na kreiranom modelu [1]. EMF sadrži meta-meta-model Ecore koji služi za kreiranje drugih meta-modela. Putem Ecore-a se mogu opisati entiteti putem njihovih atributa, veza između samih entiteta, kao i operacija koje mogu da se izvrše nad tim entitetima [2].

Nakon kreiranja meta-modela, odnosno apstraktnih sintakse, naredni korak predstavlja kreiranje konkretne sintakse namenskog jezika. Konkretna sintaksa predstavlja izgled namenskog jezika i može biti tekstualnog, grafičkog, tabelarnog ili nekog drugog tipa. Konkretna sintaksa u ovom radu je tekstualnog tipa i implementirana je u Xtext okruženju za razvoj tekstualnih jezika. Xtext predstavlja radni okvir koji služi za razvoj tekstualne sintakse namenskog jezika.

Razvoj IS zasnovan na modelima pored razvoja meta-modela i konkretne sintakse namenskog jezika podrazumeva i transformacije nad modelima. Transformacije nad modelima se definišu pomoću ATL Transformation Language (ATL) jezika za specifikaciju transformacija. ATL jezik za specifikaciju transformacija je hibridni jezik jer predstavlja kombinaciju deklarativnog i imperativnog jezika za programiranje. Deklarativni jezik je jezik koji definiše naredbe koje treba da se izvrše bez definisanja načina na koji se izvršavaju. Imperativno

programiranje predstavlja pisanje koda u kojem se korak po korak definiše način izvršavanja algoritma.

Nakon izgrađene apstraktne i konkretne sintakse i izvršenih transformacija nad modelima, sledi implementacija generatora koda. Generatori koda u ovom radu su implementirani korišćenjem Xtend jezika. Xtend je jezik opšte namene koji se koristi i za specifikaciju generatora koda.

3. PRISTUP RAZVOJU SOFTVERA ZASNOVAN NA MODELIMA

Razvoj IS zasnovan na modelima ima za cilj povišenje nivoa apstrakcije i poboljšanje automatizacije u procesu razvijanja softvera [3]. Model Driven Engineering (MDE) koristi modele na različitim nivoima (meta-meta-model, meta-model, model) kako bi postigao to povećanje nivoa apstrakcije. Veća automatizacija ostvaruje se putem transformacija iz jednog u drugi model i na kraju generisanjem koda iz modela na najnižem nivou apstrakcije. Razvoj IS zasnovan na modelima sastoji se iz namenskih jezika, transformacija i generatora.

Prednost pristupa zasnovanog na modelima je efikasnost koja se postiže generisanjem koda. Takođe u MDE validacije se definišu na višem nivou apstrakcije, a s obzirom da se koristi generator, manja je šansa da će doći do nekonzistentnosti, što je slučaj kod tradicionalnih pristupa. Jedna od najvećih prednosti MDE je to što su domenski eksperti uključeni u razvoj sistema i samim tim je manja šansa da dođe do nesporazuma. Takođe, s obzirom da se modelovanje radi na višem nivou apstrakcije, više napora ulaže se u modelovanje poslovne logike, umesto na njenu implementaciju.

Za implementaciju rešenja u ovom radu odabran je MDE zbog jednostavnosti transformisanja modela i mogućnosti generisanja više različitih izlaza koristeći isti ulaz.

4. METODA SINTEZE

Metoda sinteze predstavlja postupak projektovanja šeme baze podataka na osnovu datih obeležja i FZ. Algoritam sinteze služi za transformisanje šeme univerzalne relacije u skup šema relacija koji zadovoljava sledeće uslove:

1. nalazi se najmanje u trećoj normalnoj formi,
2. ima očuvan polazni skup FZ i
3. ima očuvanu spojivost bez gubitaka.

Koraci algoritma sinteze su sledeći [4]:

1. Formiranje kanoničkog pokrivača:
 - a. dekompozicija desnih strana skupa fz,
 - b. redukcija levih strana fz i
 - c. eliminacija redundantnih fz.
2. Transformacija kanoničkog pokrivača:
 - a. particioniranje kanoničkog pokrivača,
 - b. određivanje ekvivalentnih levih strana,
 - c. uklanjanje tranzitivnih zavisnosti i
 - d. rekonstrukcija particije kanoničkog pokrivača.
3. Formiranje relacione šeme baze podataka:
 - a. formiranje skupa šema relacija i
 - b. formiranje ograničenja stranog ključa.
4. Očuvanje spoja bez gubitaka.

Pored metode sinteze postoji i metoda dekompozicije kao jedan od načina za projektovanje šeme baze podataka.

Jedna od prednosti metode sinteze u odnosu na metodu dekompozicije je to što algoritam sinteze garantuje očuvanje početnog skupa FZ dok sa druge strane, metoda dekompozicije to ne garantuje [5]. Za ovaj rad odabrana je metoda sinteze s obzirom da na metodu dekompozicije veliki uticaj ima raspored FZ, a sinteza nema takvih ograničenja.

5. META-MODELI

Kreiranje meta-modela je prvi korak u procesu implementacije. U ovom radu kreirana su dva meta-modela. Jedan je meta-model šeme univerzalne relacije, a model koji odgovara ovom meta-modelu predstavlja ulaz za lanac ATL transformacija. Drugi je meta-model šeme baze podataka i model koji odgovara ovom meta-modelu predstavlja izlaz za lanac transformacija.

Meta-model šeme univerzalne relacije sadrži klasu Model koja predstavlja nadklasu svim ostalim klasama. Klasa NamedElement sadrži samo atribut ime i nju nasleđuju sve klase koje treba da imaju naziv. Klasa GeneralOptions se koristi za generisanje koda koje se vrši nakon transformacija. GeneralOptions sadrži dva atributa "html" i "sql" koji su tipa Boolean, i u koje se unosi vrednost da li se generiše HyperText Markup Language (HTML) odnosno Structured Query Language (SQL) kod. UniversalGroupOfAttributes je klasa koja odgovara pojmu univerzalne grupe atributa iz šeme univerzalne relacije. Attribute klasa predstavlja pojedinačni atribut iz skupa svih atributa šeme univerzalne relacije. Type predstavlja tip vrednosti koje određeni atribut ima. FunctionalDependency je klasa koja predstavlja funkcionalne zavisnosti kao odnose između dva skupa atributa iz univerzalne šeme relacije. GroupFD modeluje grupu FZ i šema univerzalne relacije ima samo jednu grupu FZ u kojoj se nalaze sve FZ. GroupOfGroups klasa predstavlja skup skupova FZ. GroupOfGroups u prvobitnoj šemi univerzalne relacije ne postoji, ali u kasnijim koracima se FZ grupišu, a grupa tih grupa je GroupOfGroups.

Meta-model šeme baze podataka započinje klasom Database odnosno bazom podataka koja, sadrži sve ostale klase. Kao i u prethodnom meta-modelu postoji klasa ModelElement koja sadrži samo atribut ime koju ostale klase mogu naslediti. Klasa Table predstavlja tabelu kao gradivni element svake baze podataka. Column klasa predstavlja kolonu u tabeli, pa samim tim ima vezu sa klasom Table. Ključ šeme relacije je minimalni podskup skupa obeležja šeme relacije, na osnovu kojeg se jedinstveno može identifikovati svaka torka relacije nad datom šemom, a primarni ključ je jedan izabrani ključ od svih ekvivalentnih ključeva [6]. PrimaryKeyConstraint klasa predstavlja koncept primarnog ključa tabele. UniqueKeyConstraint odgovara konceptu ograničenja jedinstvenosti, koje podrazumeva da vrednosti u toj koloni moraju biti jedinstvene. Klasa ForeignKey se mapira na koncept stranog ključa u bazi podataka, a označava da kolona koja je strani ključ u jednoj tabeli može da primi samo one vrednosti koje se nalaze u drugoj tabeli kao primarni ključ ili kolona sa ograničenjem jedinstvenosti.

6. KONKRETNA SINTAKSA

Konkretna sintaksa obezbeđuje način komunikacije korisnika sa aplikacijom. odnosno način da, u slučaju ove

aplikacije, korisnik unese neophodne informacije kako bi započeo proces projektovanja baze podataka. Konkretna sintaksa korišćena u ovom radu je tekstualna. Primer konkretne sintakse prikazan je na slici 1.

```
Model: StudentskaSluzba
U = {BRI:VARCHAR(7), IME:VARCHAR(50),
SEM:TINYINT, SPR:INT, NAZPRED:VARCHAR(50),
SSMER:INT, NAZSMER:VARCHAR(50),
IMERUK:VARCHAR(50), OCENA: TINYINT }
F = {BRI-> IME + SEM + SSMER + NAZSMER + IMERUK,
SSMER -> NAZSMER + IMERUK, BRI + SPR -> OCENA,
SPR -> NAZPRED + IMERUK }
#html #sql
```

Slika 1. Primer konkretne sintakse

Atributi koji se pojavljuju na slici imaju sledeća značenja:

- BRI - broj indeksa, tekstualnog tipa,
- IME - ime i prezime studenta, tekstualnog tipa,
- SEM - redni broj semestra, numeričkog tipa,
- SPR - šifra predmeta, numeričkog tipa,
- NAZPRED - naziv predmeta, tekstualnog tipa,
- SSMER - šifra smer, numeričkog tipa,
- NAZSMER - naziv smer, tekstualnog tipa,
- IMERUK - ime i prezime rukovodioca smer, tekstualnog tipa i
- OCENA - ocena iz predmeta, numeričkog tipa.

4. ATL TRANSFORMACIJE

Algoritam sinteze započinje formiranjem kanoničkog pokrivača. Kanonički pokrivač je skup FZ takav da je ekvivalentan početnom skupu FZ, da ima redukovane leve strane FZ, da se sa desne strane FZ nalazi samo po jedno obeležje i da ne sadrži redundantne FZ. Prvi korak u ovom delu algoritma je dekompozicija desnih strana FZ. Ovaj korak predstavlja u suštini samo rastavljanje FZ tako da sa desne strane ostaje po jedno obeležje. Nakon izvršenog prvog koraka, grupa FZ izgleda ovako:

- Group FD H
 - BRI → IME
 - BRI → SEM
 - BRI → SSMER
 - BRI → NAZSMER
 - BRI → IMERUK
 - SPR → NAZPRED
 - SPR → IMERUK
 - SSMER → NAZSMER
 - SSMER → IMERUK
 - BRI, SPR → OCENA

Drugi korak algoritma sinteze je redukcija levih strana FZ. Prvi korak je određivanje broja atributa sa leve strane FZ i ukoliko FZ ima samo jedno obeležje sa leve strane, ta FZ se odmah prosleđuje u izlazni model. Ukoliko postoji više od jednog obeležja, za svako obeležje sa leve strane vrši se provera da li je moguće redukovati ga. Prvo se izračunava pokrivač za najmanji skup atributa i ukoliko pokrivač nekog od podskupova sadrži i obeležje sa desne strane FZ, ta funkcionalna zavisnost će biti redukovana. Na primeru korišćenom u ovom radu ne postoji mogućnost redukcije levih strana.

Eliminacija redundantnih FZ predstavlja naredni korak algoritma sinteze. Ovaj korak zasniva se na eliminisanju

FZ koje su trivijalne ili tranzitivne. Trivijalne FZ su one FZ u kojima su leva i desna strana jednake ili je skup sa desne strane podskup skupa sa leve strane. Funkcionalna zavisnost $X \rightarrow A$ se naziva tranzitivnom, ako važi $X \rightarrow Y$, $Y \rightarrow A \in F+$ i $Y \rightarrow X \notin F+$ [5]. Uklanjanjem te FZ ne gube se nikakve informacije, stoga je ona redundantna. U ATL transformaciji se za svaku levu stranu FZ računa pokrivač, uzimajući u obzir sve FZ sem te. Nakon što se izračuna pokrivač proverava se da li on sadrži atribut sa desne strane FZ i ako sadrži ta FZ se redukuje.

Na primeru iz ovog rada dolazi do eliminacije dve FZ: $BRI \rightarrow IMERUK$ i $BRI \rightarrow NAZSMER$. Ukoliko posmatramo prvu izbačenu FZ, zatvarač BRI bez te FZ sadrži BRI, IME, SEM, SSMER, IMERUK, NAZSMER, odnosno sadrži atribut sa desne strane NAZSMER. Nakon što je ta FZ uklonjena, pokrivač leve strane za drugu FZ je BRI, IME, SEM, SSMER, IMERUK, NAZSMER, tj. pokrivač sadrži atribut sa desne strane IMERUK.

Nakon prva tri koraka, skup FZ predstavlja kanonički pokrivač. Transformacija kanoničkog pokrivača se vrši sa ciljem da se otkriju mogući ekvivalentni ključevi i da ti ekvivalentni ključevi postanu ključevi veće šeme relacije umesto više manjih.

Particionisanje kanoničkog pokrivača se vrši na način da jednu grupu funkcionalnih zavisnosti čine FZ koje imaju iste leve strane. Za razliku od prethodnih koraka gde se model sastojao iz jedne grupe FZ, u ovom koraku on sadrži jednu grupu grupa funkcionalnih zavisnosti koja se sastoji od grupa FZ, a svaka od tih grupa ima FZ sa jednakim levim stranama. Grupe FZ u primeru ovog rada, nakon particionisanja kanoničkog pokrivača su sledeće:

- Group Of Groups G
 - Group FD G(BRI)
 - BRI → IME
 - BRI → SEM
 - BRI → SSMER
 - Group FD G(SPR)
 - SPR → NAZPRED
 - SPR → IMERUK
 - Group FD G(SSMER)
 - SSMER → NAZSMER
 - SSMER → IMERUK
 - GroupFD G(BRI_SPR)
 - BRI, SPR → OCENA

Naredni korak u algoritmu sinteze predstavlja određivanje ekvivalentnih levih strana, a dve leve strane su ekvivalentne ako njihovi pokrivači sadrže isti skup atributa. U tom slučaju te dve šeme relacije spajaju se u jednu, a obe leve strane predstavljaju ključ šeme relacije.

Korak ukidanje tranzitivnih zavisnosti podrazumeva kreiranje novog skupa J koji sadrži FZ nastale kao kombinacije atributa levih strana. Nakon odvojene grupe J i izbačenih FZ iz njihovih originalnih grupa, vrši se redukcija suvišnih FZ u odnosu na uniju skupa J i preostalih grupa.

Poslednji korak u transformaciji kanoničkog pokrivača predstavlja rekonstrukcija kanoničkog pokrivača. FZ iz grupe funkcionalnih zavisnosti J se vraćaju nazad u odgovarajuće grupe FZ na osnovu levih strana.

Formiranje skupa šema relacija vrši se prevođenjem svake od grupa FZ u zasebnu šemu relacije. Šema relacije se predstavlja kao $N(R, K)$ pri čemu je R skup obeležja, a K skup ključeva. Prevođenje se vrši tako što se iz svake grupe FZ uzimaju sva obeležja i uključuju u R . Ključevi predstavljaju skupove atributa sa leve strane FZ, koji se uključuju u K . Šema baze podataka predstavlja skup svih šema relacija.

Formiranje ograničenja stranog ključa, kao poslednji korak u formiranju skupa šema relacija, implementiran je kroz dve ATL transformacije. Razlog za razdvajanje predstavlja dodavanje dodatne veze između atributa i primarnih i stranih ključeva. Atribut ima dve veze prema ključevima, a to je da ključ sadrži atribut, i da atribut pripada nekom ključu. Prvi korak kod određivanja primarnog i stranog ključa je kreiranje koncepta Primary and Unique Key i Foreign Key koji sadrže attribute od kojih se sastoje. Drugi korak predstavlja dodavanje druge veze između atributa i ključeva, a to je da atribut pripada nekom ključu.

7. GENERATORI KODA

U ovom radu kreiran je generator koji rezultira izvršivom SQL skriptom, kao i generator HTML stranice. Generator HTML-a prikazuje rezultate koraka algoritma sinteze. On daje uvid u postupak kojim se dolazi od šeme univerzalne relacije do šeme baze podataka. Takođe, ukoliko ima bilo kakve sumnje da postoji greška u algoritmu, tokom njegovog razvoja, na ovaj način se može lakše utvrditi da li postoji i u kojem koraku je ona nastala.

Drugi generator služi za prevođenje modela koji odgovara meta-modelu šeme baze podataka u izvršivu SQL skriptu. Generisanje se vrši imajući u vidu sintaksu jezika za upite, a imajući SQL Server kao osnovnu platformu za implementaciju. Ovim generatorom dolazi se do izvršive SQL skripte koja služi za kreiranje same baze podataka. SQL kod na primeru studentske službe nalazi se na slici 2.

```
IF OBJECT_ID('BRI_SPR', 'U') IS NOT NULL BEGIN
    DROP TABLE BRI_SPR;
END
IF OBJECT_ID('BRI', 'U') IS NOT NULL BEGIN
    DROP TABLE BRI;
END
IF OBJECT_ID('SSMER', 'U') IS NOT NULL BEGIN
    DROP TABLE SSMER;
END
IF OBJECT_ID('SPR', 'U') IS NOT NULL BEGIN
    DROP TABLE SPR;
END
CREATE TABLE SPR (SPR INT NOT NULL ,
    NAZPRED VARCHAR(50) NULL ,
    IMERUK VARCHAR(50) NULL ,
    CONSTRAINT SPR4301 PRIMARY KEY (SPR));
CREATE TABLE SSMER (SSMER INT NOT NULL ,
    NAZSMER VARCHAR(50) NULL ,
    IMERUK VARCHAR(50) NULL ,
    CONSTRAINT SSMER6979 PRIMARY KEY (SSMER));
CREATE TABLE BRI (BRI VARCHAR(7) NOT NULL ,
    IME VARCHAR(50) NULL ,
    SEM TINYINT NULL ,
    SSMER INT NULL ,
    CONSTRAINT BRI4880 PRIMARY KEY (BRI));
CREATE TABLE BRI_SPR (BRI VARCHAR(7) NOT NULL ,
    SPR INT NOT NULL ,
    OCENA TINYINT NULL ,
    CONSTRAINT BRI_SPR1828 PRIMARY KEY (BRI, SPR));
ALTER TABLE BRI ADD
    CONSTRAINT SSMER548 FOREIGN KEY (SSMER)
    REFERENCES SSMER (SSMER);
ALTER TABLE BRI_SPR ADD
    CONSTRAINT BRI2792 FOREIGN KEY (BRI)
    REFERENCES BRI (BRI),
    CONSTRAINT SPR8637 FOREIGN KEY (SPR)
    REFERENCES SPR (SPR);
```

Slika 2. Izgenerisana SQL skripta

8. ZAKLJUČAK

U ovom radu predstavljena je implementacija algoritma sinteze, kao metode projektovanja šeme baze podataka, korišćenjem pristupa razvoju softvera zasnovanog na modelima. Kreiran je meta-model kao osnova za kreiranje konkretne sintakse, kao i sama tekstualna konkretna sintaksa namenskog jezika za specifikaciju šeme univerzalne relacije. Takođe, kreiranjem generatora omogućeno je generisanje prikaza koraka algoritma sinteze, kao i kreiranje izvršive SQL skripte. Ovom implementacijom je omogućeno da se na lak način od korisničkog unosa specifikacije šeme univerzalne relacije dođe do isprojektovane baze podataka spremne za implementaciju putem SQL skripte.

Jedan od koraka daljeg razvoja je implementacija provere očuvanja spojivosti bez gubitaka. Provera očuvanja spoja bez gubitaka sastoji se iz određivanja ključa za šemu univerzalne relacije i provere da li krajnji skup šema relacija sadrži šemu relacije sa ključem šeme univerzalne relacije, i ukoliko je ne sadrži dodaje se ta šema relacije u šemu baze podataka. Još jedan od mogućih pravaca razvoja je kreiranje grafičke sintakse, pored postojeće tekstualne. Ta grafička sintaksa bi mogla imati formu UML dijagrama u kojima se entiteti povezuju sa drugim entitetima na taj način definišući funkcionalne zavisnosti. Još jedna od mogućih funkcionalnosti koje se mogu implementirati su neka od ograničenja koje trenutno ne mogu biti eksplicitno zadata. Četvrti pravac razvoja koji je planiran je specifikiranje ciljne platforme za implementaciju SQL skripte u konkretnoj sintaksi, tako da se SQL skripta generiše za platformu koja je navedena, umesto kao što se trenutno generiše samo za podrazumevanu SQL Server platformu.

9. LITERATURA

- [1] R. Gronback, "Eclipse modeling project", 2016. [Online]. Dostupno na: <https://goo.gl/p5Odha>.
- [2] Merks, E., Eliersick, R., Grose, T., Budinsky, F., Steinberg, D., "The Eclipse Modeling Framework", 2003
- [3] J. D. Haan, "MDE - model driven engineering - reference guide - by Johan den Haan", in Software Engineering, The Enterprise Architect, 2009. [Online]. Dostupno na: <http://goo.gl/KLxi8l>.
- [4] I. Luković, "Projektovanje šeme baze podataka metodom sinteze". [Online]. Dostupno na: <http://goo.gl/l6HqS4>
- [5] P. Mogin, I. Luković, M. Govedarica, „Principi projektovanja baza podataka“, 2004.
- [6] P. Mogin, I. Luković, „Principi baza podataka“, 1996.

Kratka biografija:



Mirjana Laušev rođena je u Novom Sadu 1991. god. Osnovne studije završila je 2010. god. na Ekonomskom Fakultetu u Novom Sadu, smer Poslovni informacijski sistemi. Master studije na Fakultetu tehničkih nauka je upisala 2014. god.

DSL I GENERATOR KODA ZA SPECIFIKACIJU KOMUNIKACIJE IZMEĐU SERVERA I HARDVERSKIH UREĐAJA**DSL AND CODE GENERATOR FOR THE SPECIFICATION OF THE COMMUNICATION BETWEEN SERVER AND HARDWARE DEVICES**

Viktor Šujić, Fakultet tehničkih nauka, Novi Sad

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu su opisane mogućnost T4 Template (Text Template Transformation Toolkit) alata u razvoju rešenja vođenog modelima.

Mogućnosti alata su prikazani preko implementacije generatora koda za podršku komunikacije između hardverskih uređaja.

Abstract – This theses presents the T4 Template (Text Template Transformation Toolkit) tool's possibilities in a model driven development solution.

The tool's possibilities are presented through the implementation of a code generator, for specification of communication between server and hardware devices.

Ključne reči: DSL, T4 template, MDE

1. UVOD

U trenutnom svetu sve je veća potreba za kvalitetnim i brzo implementiranim softverskim rešenjima, koja se mogu lako održavati i brzo unapređivati. Razvojni timovi softverske firmemoraju ovako razmišljati da ih konkurencija ne bi uništila. Za rešavanje ovog izazova su uvedene različite metode brzog razvoja softvera.

Jedan od ovih metoda je MDE – Model Driven Engineering [1]. MDE je predstavlja skup metoda i tehnika za razvoj softvera kod kojih je model na visokom nivou apstrakcije osnova za generisanje velikog procenta koda neke aplikacije. Domenski modeli su modeli specijalizovani za rešavanje jednog specifičnog problema, bazirani na jezicima specifičnim za domen.

MDE pristup je namenjen za povećanje produktivnosti putem povećanja kompatibilnosti sistema, pojednostavljenjem procesa dizajna i promovisanjem komunikacije između pojedinaca i timova koji rade narazvoju. Modelovanje je efektivno ako model ima smisla iz ugla korisnika koji poznaje domen i ako može poslužiti kao osnova za implementaciju sistema. Najbolje je ako modeli mogu da se kreiraju u saradnjimenedžera proizvoda, dizajnera, developera i poznavaoa domena.

Zadatak ovog rada jeste da se isproba korišćenje Text Template Transformation Toolkit (odnosno T4 template) alata u kontekstu razvoja jednog MDE rešenja. Rešenje koje će se razvijati jeste XML bazirani sistem za komunikaciju putem Request-Response poruka između

hardverskog uređaja i servera. Preko ovih poruka je omogućeno konstantno ažuriranje podataka uređaja i servera.

Rad je organizovan na sledeći način. U drugom poglavlju rada je ukratko prikazan T4 alat. U trećem poglavlju je prikazan način implementacije i generisanje koda za sistem za komunikaciju, korišćenjem T4 alata. Četvrto poglavlje sadrži zaključak i pravce daljeg razvoja.

2. T4 TEMPLATE

Microsoft-ov Text Template Transformation Toolkit (odnosno T4 template) je obrađivač šablona (template engine) koji je uključen je u Visual Studio. T4 šabloni u većini slučajeva imaju ekstenziju ".tt" [4].

Developeri T4 koriste kao deo aplikacije ili framework-a za automatizaciju kreiranja tekst fajlova. Ovi fajlovi mogu biti bilo kog tekstualnog formata, kao npr. C#, XML, HTML itd.

T4 ima svoj format za pisanje šablona koji u sebi sadrži .NET kod i string literale. Šabloni se parsiraju putem T4 alata iz komandne linije i prevode u .NET kod, posle čega se kompajliraju i izvršavaju. Izlaz je tekstualni fajl generisan putem šablona. Kreiran T4 jezik se može koristiti i unutar .NET aplikacije sa ubacivanjem TextTransformation klase koja eliminiše potrebu korisnika da ima instaliran Visual Studio. Tri glavna koraka u generisanju koda preko T4 template-a su:

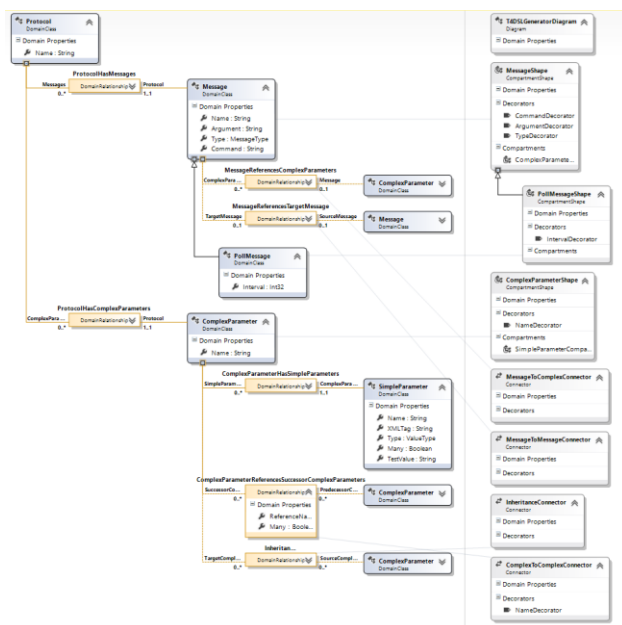
- DSL metamodel
- Šabloni (Template fajlovi)
- Eksperimentalna instanca

Šabloni (Listing 2.1) se sastoje od pet različitih elemenata:

- Direktive za T4 procesor (elementi između <#@ #> tagova). Vrednosti upisane u direktive neće biti vidljive u generisanom kodu, već služe za definisanje nasleđivanja i drugih veza između šablona.
- Kod za kontrolu toka procesa (elementi između <# #> tagova). Glavna funkcija koja se uvek izvršava. Tokom kodiranja se koristi C# jezik.
- Pomoćni kod (elementi između <#+ #> tagova). Služi za definisanje pomoćnih funkcija i izvršava se samo ako se pozove iz koda za kontrolu toka procesa.
- Ispis (elementi između <#= #> tagova). Služi za ispis vrednosti parametara.
- Direktni ispis (van tagova). Ovi elementi su statični ispisi.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević.



Slika 2.1. DSL dijagram – metamodel

Metamodel jezika koji se opisuje u okviru T4 alata se nalazi u `DSLDefinition.dsl` grafičkom fajlu (Slika 2.1).

```

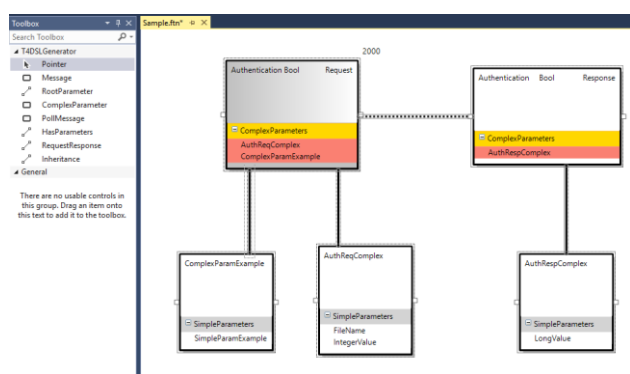
<# Template Inherits="Microsoft.VisualStudio.TextTemplating.VSHost.ModelingTextTransformation" hostspecific="true" #>
<# OutExtension="text" #>
<# T4DSLGenerator processor="T4DSLGeneratorDirectiveProcessor" requires="filename=Sample.ftn" #>

<#
// When you change the DSL Definition, some of the code below may not work.
for (int i = 0; i < MessageCommands.Count(); ++i)
{
    <# MessageCommands[i] #>
}

}
}
<#
List<string> MessageCommands
{
    get
    {
        List<string> result = (List<string>)CallContext.GetData("T4DSL.Report.MessageCommands");
        return result;
    }
}
#>

```

Listing 2.1. Primer šablona



Slika 2.2. Eksperimentalna instanca

Eksperimentalna instanca služi za kreiranje modela i generisanje koda na osnovu tog modela. Model se kreira na osnovu elemenata definisanih u metamodelu jezika, koji se pojavljuju na Toolbox-u. Kod se generiše snimanjem modela (naprimero sa slike 2.2 kod se generiše snimanjem Sample.ftn fajla). Generisani kod će se pojaviti u *Debugging Solution*-u. *Debugging Solution* sadrži fajlove koji se koriste tokom modelovanja i omogućuje pregled generisanih fajlova.

3. IMPLEMENTACIJA T4DSLGENERATOR PROJEKTA

Implementirani projekat služi za komunikaciju između servera i hardverskih uređaja XML porukama. Generisani kod je na C# jeziku.

3.1. Kreiranje DSL dijagrama

DSL dijagram je metamodel jezika (Slika 2.1.), čije će instance biti u okviru modela našeg rešenja. U ovom projektu je definisano pet domenskih metaklasa:

- **Protocol** – predstavlja sam projekat. Sadrži samo “Name” atribut. Tokom projektovanja metamodela izabrano ime će biti i ime generisanog projekta.
- **Message** – predstavlja jednu XML baziranu poruku. Jedan Protocol može imati više poruka. “Message“ ima 4 atributa:
 - Argument – argument poruke
 - Type – enumeracija koja može imati vrednosti “Request” ili “Response”.
 - Command – naziv komande
 - Name – naziv poruke, koja se generiše iz vrednosti prethodna 3 atributa (“Command”+“Argument”+“Type”)

3. **PollMessage** – predstavlja specijalnu vrstu poruke. Od ovog tipa poruke svaki Protocol mora sadržati jedan. PollMessage nasljeđuje attribute Message-a. Razlika između PollMessage-a i Message-a jeste atribut “Interval” koji definiše na koliko milisekundi će aplikacija tražiti od servera osvežavanje podataka.

4. **SimpleParameter** – predstavlja jednu prostu tag vrednost u XML poruci. SimpleParameter mora biti član jednog kompleksnog parametra. Sadrži 5 elemenata:

- Name – naziv promenjive
- Type – tip promenjive (Integer, Long, Boolean, String, UnsignedLong)
- TestValue – *default* vrednost
- Many – zbog specijalnih zahteva potrebno je uvesti ovu *flag* vrednost, koja definiše da li se može ovaj tag naziv ponavljati u poruci
- XMLTag – naziv taga. Zbog specijalnih prefiksa iz ove promenjive se automatski može izgenerisati tip i naziv.

5. **ComplexParameter** – kompleksni parametar je skup prostih parametara. Jedini atribut koji poseduje je “Name” kojim se definiše ime parametra. ComplexParametar mora postojati bar u jednoj poruci. Jedan kompleksni parametar može nasleđivati proste parametre drugog kompleksnog parametra. ComplexParameter može sadržati i druge ComplexParameter-e.

3.2. Definisanje provere validnosti

Pri kreiranju i snimanju DSL dijagrama sistem generiše odgovarajuće klase za definisane elemente. Klase se generišu na osnovu šablona koji se nalaze u `GeneratedCode` mapi DSL projekta. Proširenjem ovih

klasa moguće je dodati i validacione provere za definisane elemente. Listing 3.2.1 prikazuje validaciju nad kompleksnim parametrom.

Da bi validacija radila, potrebno je koristiti “Microsoft.VisualStudio.Modeling.Validation” biblioteku i uključiti je iznad klase komandom [ValidationState(ValidationState.Enabled)]. Provera validacije modela se može uključiti prilikom snimanja, otvaranja ili menjanja modela.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.VisualStudio.Modeling.Validation;

namespace FTN.T4DSLGenerator
{
    [ValidationState(ValidationState.Enabled)]
    partial class SimpleParameter
    {
        [ValidationMethod(ValidationCategories.Save)]
        private void ValidateSimpleParameterValues(ValidationContext context)
        {
            if (string.IsNullOrEmpty(Name) || string.IsNullOrWhiteSpace(XMLTag))
            {
                context.LogError(CustomCode.Validation.ValidationResource.EmptySimpleParameterError, "", this);
            }
            if (XMLTag.Contains(" ") || Name.Contains(" "))
            {
                context.LogError(CustomCode.Validation.ValidationResource.InvalidSimpleParameterError, "", this);
            }
        }
    }
}
```

Listing 3.2.1 SimpleParameter validacija

Poruke o greškama se mogu čuvati u resurs fajlu. Kada se desi neka greška tokom validacije, generisanje koda se neće izvršiti i prikazaće se odgovarajuća poruka o grešci. U slučaju primera SimpleParameter, greška se može desiti ako je polje “Name” ili “XMLTag” ostalo prazno, ili sadrži praznine (“whitespace”).

3.3. Implementacija šablona

Kao prvi korak pri kreiranju šablona mora se kreirati klasa koja služi za povezivanje šablona sa *TemplatedCodeGenerator*-om, koji će pokrenuti učitavanje šablona i generisanje koda. Radi bolje vidljivosti, ovaj zadatak je podeljen na dve klase.

Prva klasa je *CodeGenerator.cs* (Listing 3.3.1) koja se dodaje u *DSLPackage* projekat u *CustomCode/CodeGeneration* folder. Ova klasa služi za pokretanje procesa. Da bi se to postiglo, neophodno je uključiti biblioteku “Microsoft.VisualStudio.TextTemplating.VSHost” i podesiti nasleđivanje sistemske klase “TemplateCodeGenerator”. Pri pokretanju generisanja koda, iz modela će biti pozvana “GenerateCode” funkcijasa parametrima:

- **inputFileName** – ime modela iz kojeg se generiše kod
- **inputFileContent** – sadržaj modela

U sledećem koraku je potrebno dodati ”CodeGenerator.pkgdef” fajl (Listing 3.3.2). Ovim fajlom deklariramo da je ID “Custom Tool” T4DSLGenerator-a “FAE04EC1-301F-11d3-BF4B-00C04F79EFBC”. Tool povezujemo sa T4DSLGenerator-om preko prethodno kreiranog GUID-a i definišemo da će generator biti pozvan kada se snimi fajl sa “.ftn” ekstenzijom.

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Microsoft.VisualStudio.TextTemplating.VSHost;
using FTN.T4DSLGenerator.CustomCode.CodeGeneration;
using System.Runtime.Remoting.Messaging;

namespace FTN.T4DSLGenerator
{
    [System.Runtime.InteropServices.Guid("4FF3608D-91C7-428D-863B-F0B85CC09814")]
    public class CodeGenerator : TemplatedCodeGenerator
    {
        protected override byte[] GenerateCode(string inputFileName, string inputFileContent)
        {
            // Replace the supplied file contents with the template we want to run
            inputFileContent = ASCIIEncoding.UTF8.GetString(CodeGenerationResource.T4DSLGenerator);

            // Substitute the name of the current model file into the template.
            FileInfo fi = new FileInfo(inputFileName);
            inputFileContent = inputFileContent.Replace("Sample.ftn", fi.Name);
            inputFileContent = inputFileContent.Replace("namespace Debugging", "namespace " + FileNamespaces);

            // Now just delegate the rest of the work to the base class
            byte[] data;
            data = base.GenerateCode(inputFileName, inputFileContent);
            byte[] ascii = new byte[data.Length - 3];
            Array.Copy(data, 3, ascii, 0, data.Length - 3);

            return ascii;
        }
    }
}
```

Listing 3.3.1 Primer povezivanja kod generatora sa modelom

Povezujemo ga sa “FTN.T4DSL Generator.-CodeGenerator” klasom. Svaki novi fajl sa ekstenzijom “.ftn” će u svoj “Custom Tool” parametar automatski dobiti vrednost “T4DSLGenerator”.

```
[RootKey\Generators]
[RootKey\Generators\{FAE04EC1-301F-11d3-BF4B-00C04F79EFBC}]
[RootKey\Generators\{FAE04EC1-301F-11d3-BF4B-00C04F79EFBC}\T4DSLGenerator]
@="Custom tool that generates C# files for state machines described in .sm files"
"CLSID"="{4FF3608D-91C7-428D-863B-F0B85CC09814}"
"GeneratesDesignTimeSource"="dword:00000001"
[RootKey\CLSID]
[RootKey\CLSID\{4FF3608D-91C7-428D-863B-F0B85CC09814}]
@=" Custom tool that generates C# files for state machines described in .sm files "
"Class"=" FTN.T4DSLGenerator.CodeGenerator"
"InprocServer32"="%WinDir%\SYSTEM32\MSCOREE.DLL"
"ThreadingModel"="Both"
"CodeBase"="%PackageFolder%\FTN.T4DSLGenerator.DslPackage.dll"
[RootKey\Generators\{FAE04EC1-301F-11d3-BF4B-00C04F79EFBC}\.ftn]
@=" T4DSLGenerator "
```

Listing 3.3.2 Primer definisanja kod generatora

Drugi deo je T4DSLGenerator šablon, koji se kao i ostali šabloni ubacuje u “CodeGenerationResource” resurs fajl za lakše korišćenje.

Šabloni (Listing 3.3.3) zavise od kreiranog modela i zbog toga, pre njihovog procesiranja, moraju im se proslediti informacije o modelu. Pročitani podaci iz modela se mogu proslediti pomoću uključene “System.Runtime.Remoting.Messaging” blliblioteke i pozivanjem *CallContext.SetData("key", value)* i *GetData* funkcije.

3.4. Modeliranje u eksperimentalnoj instanci

Pokretanjem projekta otvara se eksperimentalna instanca, koja služi za modeliranje i generisanje koda na osnovu modela.

U primeru modela (Slika 2.2) se vidi jedan par *Request-Response Authentication* poruke.

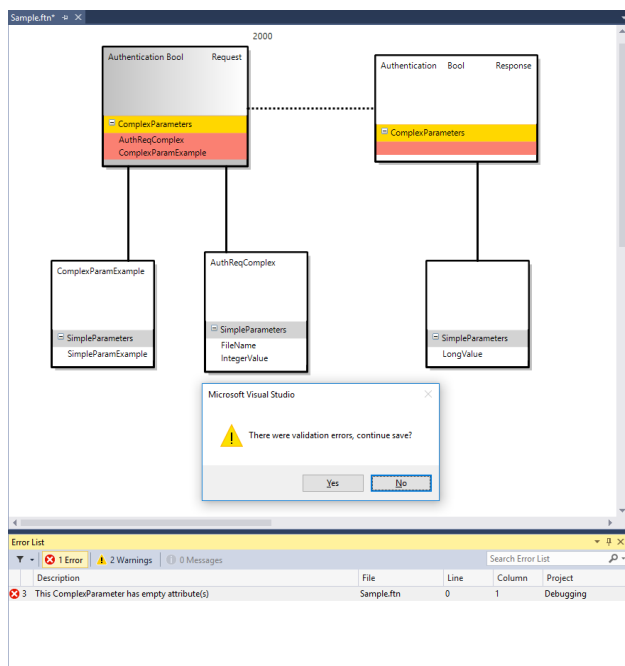
Pomoću prethodno definisanih validacionih provera se mogu izbeći greške u generisanom kodu. Validacijom proveravamo vrednosti parametara elemenata i ako je primećena greška šaljemo poruku greške (Slika 3.4.1). Ukoliko kliknemo na neku grešku, sistem će obeležiti objekat nad kojim je određena greška nađena.

```

1 Protocol protocol= this.Protocol;
2
3 //<# ProtocolName #>, FTN Novi Sad
4 // <copyright file="MessageTypes.cs" company="FTN">
5 // 0 <# DateTime.Now.Year.ToString() #> FTN
6 // Website: http://www.ftn.uns.ac.rs/
7 // </copyright>
8 using System;
9 using System.Collections.Generic;
10 using System.Linq;
11 using System.Text;
12
13 namespace FTN.DataSystem.Plugins.<# ProtocolName #>.Messages
14 {
15     internal static class MessageTypes
16     {
17         public const string <# ProtocolName #> = "<# ProtocolName #>";
18
19         List<string> mList = new List<string>();
20         for(int i = 0; i < MessageCommandVariables.Count(); i++)
21         {
22             string message = MessageCommands[i];
23             string command = MessageCommandVariables[i];
24             if(!mList.Contains(command))
25             {
26                 public const string <# command #> = "<# message #>";
27
28                 mList.Add(command);
29             }
30         }
31     }
32
33 List<string> MessageCommandVariables
34 {
35     get
36     {
37         List<string> result = (List<string>)CallContext.GetData("T4DSLReport.MessageCommandVariableList");
38         return result;
39     }
40 }
41 string ProtocolName
42 {
43     get
44     {
45         string result = (string)CallContext.GetData("T4DSLReport.ProtocolName");
46         if (result == null)
47             result = string.Empty;
48         return result;
49     }
50 }
51 List<string> MessageCommands
52 {
53     get
54     {
55         List<string> result = (List<string>)CallContext.GetData("T4DSLReport.MessageCommands");
56         return result;
57     }
58 }
59 }

```

Listing 3.3.3 Šablon MessageTypes.t4



Slika 3.4.1 Primer greške u validaciji

4. ZAKLJUČAK

Tokom implementacije nisam naišao na problem koji se nije mogao rešiti preko T4 *template*-a. Jedina prepreka na koju sam naišao je da Visual Studio nema punu podršku pri pisanju šablona. Iako posle instalacije dodatka (plugin) prepoznaje neke definicije i obeležava ih različitim bojama, tokom kodiranja ne prepoznaje definisana imena klasa i atributa i ne nudi njihove moguće vrednosti(CTRL + Space).

U okviru ovog rešenja omogućeno je generisanje osnovnog dela .NET aplikacije koja XML porukama komunicira sa serverom ili drugim uređajima. Osnovni deo aplikacije se generiše iz modela. Generisane klase definišu poruke sa njihovim događajima i prostim funkcijama. Ove generisane klase predstavljaju osnovne stubove aplikacije, i u kasnijim fazama ubrzaju implementaciju funkcija koji su specifične za dati uređaj.

Ono što u okviru ovog projekta nije urađeno jeste kreiranje *plugin* instalacije. Sa *plugin*-om bi posle instalacije developeri i projektni menadžeri odmah posle definisanja poruka mogli kreirati projekat bez promene izvornog koda DSL generatora. Takođe bi se moglo još dodati generisanje korisničkog interfejsa aplikacije i baze skripte. U kasnijim fazama bi trebalo dodati i mogućnost da se sa učitavanjem XML poruke kreira odgovarajuća poruka u modelu. Ovim korakom bi se moglo izbeći prepisivanje vrednosti u model i smanjila bi se mogućnost ljudske greške prilikom prepisivanja.

5. LITERATURA

- [1] MDE - https://en.wikipedia.org/wiki/Model-driven_engineering
- [2] Sitecore.CodeGenerator - <https://github.com/hermanussen/sitecore.codegenerator>
- [3] Glass Mapper - <https://github.com/mikeedwards83/Glass.Mapper>
- [4] T4 template - https://en.wikipedia.org/wiki/Text_Template_Transformation_Toolkit
- [5] ASP.NET_MVC - https://en.wikipedia.org/wiki/ASP.NET_MVC
- [6] Tutorial - <https://code.msdn.microsoft.com/windowsdesktop/Visualization-and-Modeling-313535db/sourcecode?fileId=19026&pathId=322740127>

Kratka biografija

Viktor Šujić rođen je 15.5.1990. godine u Subotici. 2009. godine upisao je Fakultet Tehničkih Nauka, odsek Računarstvo i automatika, smer Primenjene Računarske Nauke i Informatika u Novom Sadu. Osnovne akademske studije završio je 2014 godine.

ПРОШИРЕЊЕ КРОКИ АЛАТА НОВИМ ТИПОВИМА ФОРМИ**EXTENDING KROKI TOOL WITH NEW FORM TYPES**

Nemanja Blagojević, *Fakultet tehničkih nauka, Novi Sad*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај - У овом раду описано је проширење Кроки алата новим типовима форми, међу којима су: форма која обезбеђује унос података са везом “више према више”, форма за преузимање параметара извештаја од корисника као и форма за приказ извештаја подсистема.

Abstract - This paper describes an extension of Kroki tool with new types of forms, including: a form that provides data entry with "many to many" link, form for retrieving user entered report parameters as well as form for report subsystem display.

Кључне речи: Metodologije brzog razvoja softvera, Kroki alat, forme, proširenje, Many-to-many

1. УВОД

Употреба софтверских производа данас представља готово неопходан сегмент за ефикасно функционисање компанија. Јавила се потреба за развојем софтвера од стране лица које познају само функционисање система, а не морају у великој мери да буду квалификовани за развој софтверских производа. Дошло је до идеје за стварање интерактивног софтверског система који је у могућности да једноставним скицирањем корисничког интерфејса искреира готову пословну апликацију. Једно од таквих софтверских система је Кроки.

Тема овог рада је проширивање скупа подржаних форми Кроки алата новим типовима, међу којима су форма која обезбеђује унос података са везом “више према више” (Many-to-many), форма за преузимање параметара извештаја од корисника као и форма за приказ извештаја подсистема.

2. ОПИС ПОДРЖАНОГ СКУПА КРОКИ АЛАТА

У овом поглављу биће описан EUIS DSL (Enterprise User Interface Specification Domain Specific Language) коришћен у Крокију као и сам алат за скицирање форми, ради лакшег разумевања проширења скупа форми.

Стандардна форма је предвиђена за приказ, унос и измену података од стране корисника. У заглављу стандардне форме налази се toolbar panel који обухвата скуп дозвољених основних операција над подацима стандардне форме. У централном делу стандардне форме налази се standard panel, део резервисан за компоненте података.

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је била др Гордана Милосављевић.

Са десне стране налази се operational panel, панел са специјалним операцијама које се разликују у зависности од активног прегледа и представљају комплексне операције над податком.

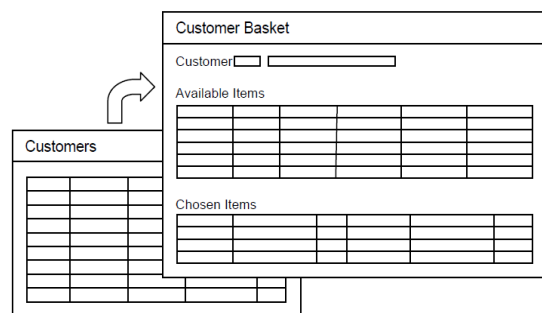
Parent-child форма, служи за приказ података које имају хијерархијску структуру (родитељ-дете). У формирању дате структуре фигурише хијерархија као основна компонента за приказ подређених и надређених форми. Сваки елемент хијерархије има индекс којим се дефинише његова позиција у хијерархији. Број хијерархија на једној parent-child форми нема ограничења. Хијерархија се смешта у централни део форме намењен за хијерархију. Уколико постоје, у доњем делу форме наводе се дугмићи са специфичним операцијама.

3. ПРОШИРЕЊЕ СКУПА ПОДРЖАНИХ ФОРМИ У КРОКИ АЛАТУ

У овом поглављу биће описан скуп форми којима је проширен Кроки алат описан у овом раду. Биће описана основна проблематика, проширења и ограничења, кроз примере у засебним потпоглављима. Такође, биће демонстрирана имплементација у оквиру Кроки алата као и приказ генерисаних форми.

3.1 MANY-TO-MANY ФОРМА

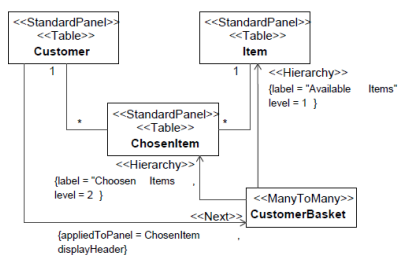
Many-to-many форма је тип форме предвиђен за интезиван унос података међусобно повезаних кардиналитетима „више према више“, са или без асоцијативних класа. Основни принцип функционисања ове форме заснива се на везној табели која се на неки начин попуњава и ажурира кључевима два ентитета. Дизајн и принцип функционисања који је замишљен за реализацију у овом раду приказан је на слици 3.1.



Слика 3.1 – Дизајн Many-to-many форме дефинисан HCI standardom

Стереотип коришћења који је предвиђен да се подаци из табеле понуђених ентитета (Item) комбиновањем са ентитетом надређене форме (Customer) пребаце у панел за преузимање и ажурирање изабраних

редова (CustomerItem) приказан је на слици 3.2.



Слика 3.2 – Стереотип Many-to-many форме

Структура Many-to-many форме је доста слична Parent-child структури, првенствено из разлога што се користи хијерархија као компонента. Из тог разлога, по узору на помоћне алате првенствено направљене за тип форме Parent-child, направљени су алати и за овај тип форме.

Many-to-many форма описана у овом раду се састоји само из панела за унос компоненти. Изглед форме (величина прозора, локација у односу на главну форму) као и панел за унос компоненти дефинисани су у оквиру класе ManyToMany модула Kroki-UIProfil. Many-to-many форма би требала да се састоји од две хијерархије, као и lookup компоненте која садржи податак о форми са које је креирана.

Ограничења која су везана за саму хијерархију су:

- да су хијерархије стриктно сачињене од стандардних форми
- да се могу додати само две хијерархије

Ограничење да су хијерархије стриктно сачињене од стандардних форми види се на листингу 3.1.

```
if (visitor != null) {
    visitor.visit(visibleElement);
    objectList = visitor.getObjectList();
    if(visibleElement instanceof Hierarchy){
        List<Object> listToRemove = new ArrayList<Object>();
        Hierarchy hierarchy = (Hierarchy) visibleElement;
        if(hierarchy.getActivationPanel() instanceof ManyToMany){
            ...

            for(Object o: objectList){
                if(!(o instanceof StandardPanel) || o
                    instanceof ReportPanel){
                    listToRemove.add(o);
                }
            }
            objectList.removeAll(listToRemove);
        }
    }
    ...
}
```

Листинг 3.1 – Ограничење дозвољених типова хијерархија

Друго од наведених ограничења подразумевало је да је потребно ограничити додавање на само две хијерархије, за разлику од Parent-child форме у којој је дозвољено додати произвољан тип хијерархија.

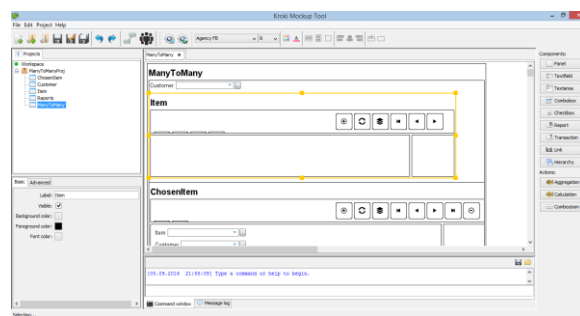
Ово ограничење је имплементирано у класи AddHierarchyAction модула Kroki-mockup-tool, описано у листингу 3.2.

```
public void actionPerformed(ActionEvent e) {
    TabbedPaneController tabbedPaneController =
        KrokiMockupToolApp.getInstance().getTabbedPaneController();
    Canvas currentCanvas =
        tabbedPaneController.getCurrentTabContent();
    tabbedPaneController.changeCursorImage(addEnabledIcon);
    tabbedPaneController.getContext().goNext(State.ADD_STATE);

    Hierarchy hierarchy = new Hierarchy();
    if(currentCanvas.getVisibleClass() instanceof ManyToMany){
        ManyToMany manyToMany = (ManyToMany)
            currentCanvas.getVisibleClass();
        int count = ManyToManyUtil.getHierarchyCount(manyToMany);
        if(count<=1){
            hierarchy.setActivationPanel(currentCanvas
                .getVisibleClass());
            ((AddState)tabbedPaneController.getContext()
                .getCurrentState()).setElement(hierarchy);
        } ...
    } else{
        hierarchy.setActivationPanel(currentCanvas
            .getVisibleClass());
        ((AddState)tabbedPaneController.getContext()
            .getCurrentState()).setElement(hierarchy);
    }
    ...
}
```

Листинг 3.2 – Ограничење броја хијерархија

Ограничење се манифестује појављивањем курсора забране при покушају додавања више од две хијерархије на форму. Убацавањем две хијерархије као и lookup-а који је везан за дату форму дефинише се њена структура. Пример правилно дефинисане форме може се видети на слици 3.3.



Слика 3.3 – Повезана Many-to-many форма

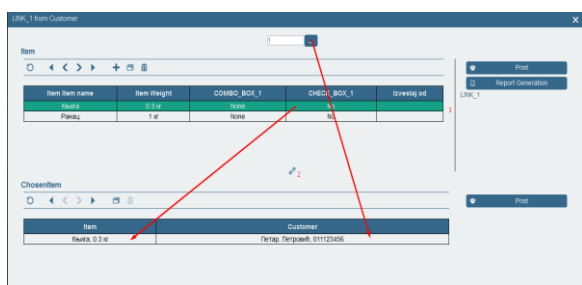
Убацавањем две хијерархије (Item – горе и ChosenItem – доле) као и lookup-а који је везан за дату форму (ChosenItem) дефинише се њена структура. Форма ChosenItem има дефинисане lookup компоненте ка ентитетима Customer и Item, док форма Customer има дефинисан Next механизам ка Many-to-many форми. На овај начин обезбеђује се потпуност Many-to-many механизма.

Након правилно дефинисане Many-to-many форме у Кроки алату, могућ је извоз и покретање пројеката као Web апликације позивом одговарајуће акције. Покретањем ове акције врши се генерисање кода за одговарајућу платформу. Many-to-many форма није захтевала генерисање EJB структуре из разлога што су се користиле већ постојеће структуре стандардних форми које представљају њене елементе. Оно што је

било неопходно је изгенерисати одговарајућу групу панела која ће садржати информације о компонентама које сачињавају Many-to-many форму. Генерисање ових панела врши се у класи PanelGenerator, а конфигурациони подаци се смештају у *panel.xml* у WebApp апликацији у којој је смештен WebFramework. Даље парсирање и тумачење на основу xml дефиниције ових елемената врши се у класи PanelReader из WebApp пројекта. Након генерисања и покретања веб апликације, први корак је отварање стандардне форме са које се позива Many-to-many форма. Како би се отворила Many-to-many forma, потребно је одабрати одговарајући податак из табеле и кликнути на линк, који уместо подразумеване акције Next за стандардну форму позива одговарајући Many-to-many форму.

Динамика између форми апликације у Web Framework-у је дефинисана у javascript фајлу page.js у складу са одговарајућим методама на серверу. Кликом на Next линк описан у претходном пасусу, javascript функција за креирање форме позива одговарајућу серверску методу за учитавање структуре Many-to-many форме. Листа свих мапираних метода налази се у класи AdaptApplication. Структура форме у зависности од потреба може се дефинисати коришћењем html или JSON образаца. За Many-to-many форму, слично имплементацији Parent-child, одлучено је да ће се користити JSON образац за добављање компоненти. Наведена JSON структура се, поред хијерархија и zoom компоненте, проширује и идентификатором надређене форме и прослеђује се назад на обраду. Сама структура форме креира се коришћењем JQuery објекта формираног на основу података прослеђеног JSON-а.

Резултат наведеног решења може се видети на слици 3.4. Одабиром одговарајућег реда из табеле Item (1) и кликом на одговарајуће дугме за пребацивање (2), комбинују се подаци ова два ентитета и попуњава се табела ChosenItem.



Слика 3.4 – Пребацивање података у табелу ChosenItem

У табели ChosenItem налазе се дефинисане дозвољене операције. Ради конзистентности брисања података из табеле, дугме за брисање је остављено на претходно дефинисаном месту.

3.2 ФОРМА ЗА ПРЕУЗИМАЊЕ ПАРАМЕТАРА ИЗВЕШТАЈА ОД КОРИСНИКА

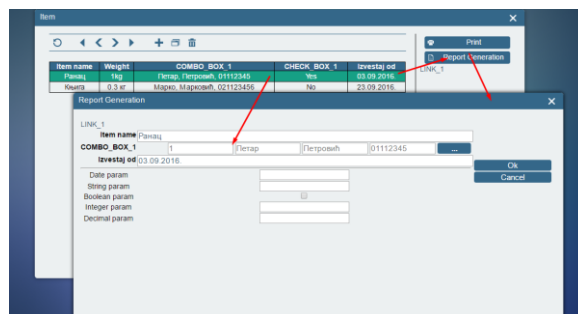
Форма за преузимање параметара извештаја од корисника предвиђена је да омогући кориснику унос и прослеђивање параметара који утичу на филтрирање података у извештају. Овај тип форме је

предвиђен да се генерише на основу иницијалне конфигурације од стране корисника, везујући је за специјалан тип операције *извештај*. Поља предвиђена за овај тип форме су: поља која одговарају ентитету за који је везан дати извештај, lookup поља, поља за преузимање додатних параметара (поља која нису везана за сам ентитет), дугмад за потврду и одустајање од акције као и дугмад са натписима за специфичне операције.

Међу компонентама са палете за додавање на стандардну форму налазе се и дугмад за специјалне операције међу којима се налази и дугме за покретање извештаја (акција Report) које је неопходно додати на стандардну форму. Неопходно је обезбедити унос параметара извештаја у напредним подешавањима наведеног дугмета. Подешавања која су убачена и чија је обрада измењена су обележена бројевима и то су:

1. Report name – параметар за име извештаја
2. Standard parameters – табела са уносом параметара стандардне форме
3. Additional parameters – табела са уносом додатних параметара извештаја.

Правилним дефинисањем параметара форме за преузимање извештаја у Кроки алату могућ је извоз и покретање пројекта као Web апликације. Први корак је отварање стандардне форме са које се позива форма за преузимање параметара извештаја. На слици 3.5 у позадини је приказана форма Item са дугметом за приказ извештаја у секцији специјалних операција.



Слика 3.5 – Отварање форме за подешавање параметара система

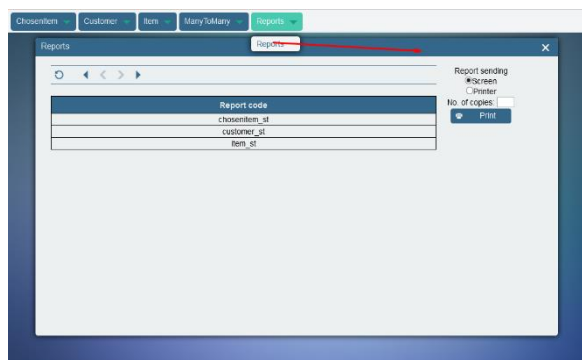
Кликом на дугме отвара се форма са параметрима за подешавање. Кликом на дугме за приказ извештаја описано сликом 3.5 из page.js фајла позива се подразумевана метода мапирање класе ParameterInfoResource у којој се врши припрема параметара за приказ на извештају. Након унетих параметара, притиском на дугме за потврду врши се приказ извештаја на екрану.

3.3. ФОРМА ЗА ПРИКАЗ ИЗВЕШТАЈА ПОДСИСТЕМА

Последњи тип форме којим је проширен подржан скуп је форма за приказ извештаја подсистема. У стандардним формама пројекта је већ омогућен приказ извештаја одабиром одговарајуће опције,

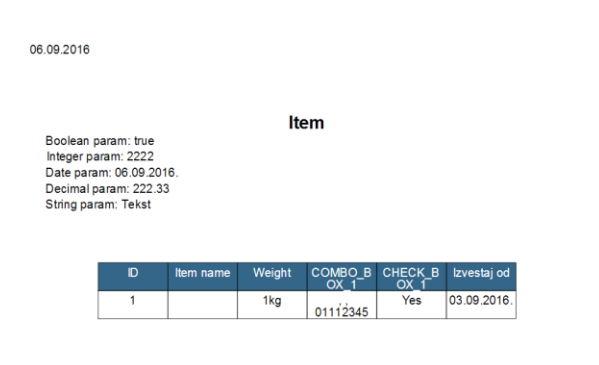
међутим овакав тип форме омогућио би већу прегледност. Форма за приказ извештаја подсистема је због своје потенцијалне комплексности реализована као посебан тип форме у оквиру Кроки алата. При креирању форме, креира се подразумевано поље за назив извештаја Report code. Ово поље представља јединствен идентификатор извештаја под којим је заведен у систему и на основу њега се зна за који извештај се врши приказ.

Правилним дефинисањем форме за приказ извештаја, а затим покретањем акције за генерисање и извоз апликације добија се одговарајућа форма у Web окружењу (слика 3.6).



Слика 3.6 – Отварање форме за приказ извештаја подсистема

Акција за приказ извештаја се разликује у зависности од тога да ли је за одређени тип стандардне форме за коју се дефинише извештај везана форма за подешавање параметара. Ако постоји, позива се подразумевана акција за приказ извештаја, у супротном позива се одговарајућа форма. Након уноса параметара и потврђивања акције добија се приказ извештаја на екрану (слика 3.7).



Слика 3.16 – Приказ извештаја у PDF форми

У заглављу извештаја су приказани додатни параметри који се могу користити у извештају, док су у табели приказани само филтрирани подаци по улазним критеријумима.

4. ЗАКЉУЧАК

У овом раду описано је проширење Кроки алата новим типовима форми, међу којима су: форма која обезбеђује унос података са везом “више према више”, форма за преузимање параметара извештаја од корисника као и форма за приказ извештаја подсистема.

Даљи развој ових типова форми подразумевао би проширивање новим корисним функцијама. Конкретно, за тип форме „више према више“ може се омогућити увоз података из више хијерархијских нивоа, при чему би се могли комбиновати подаци два или више ентитета за унос у везну табелу. Штампа извештаја у овој верзији подржава само приказ на екрану, у перспективи би требало омогућити штампање извештаја на другом излазном уређају. Такође требало би омогућити подршку за штампу коришћењем Jasper, Birt или неког другог система извештавања, који је свакако предвиђен у каснијој имплементацији додатних функционалности Кроки алата.

5. ЛИТЕРАТУРА

[1] М. Филиповић, В. Марсенић, Г.Милосављевић, И. Дејановић, Кроки: алат за

интерактивни развој пословних апликација базираних на скицама, YuInfo, Кораоник, 2012.

[2] Гордана Милосављевић, Прилог методама брзог развоја адаптивних пословних информационих система, докторска дисертација, Факултет техничких наука, Нови Сад, 2010.

[3] Рената Вадерна, Проширење Кроки алата за скицирање пословних апликација графичким UML едитором, мастер рад, Факултет техничких наука, Нови Сад, 2012.

[4] Гордана Милосављевић, Предавања са предмета Методологије брзог развоја софтвера

[5] Eclipse Modeling Framework - <http://www.eclipse.org/modeling/emf/>

[6] Free online encyclopedia - http://en.wikipedia.org/wiki/Main_Page

КРАТКА БИОГРАФИЈА

Немања Благојевић је рођен 4.7.1990. године у Новом Саду. 2009. године уписао Факултет техничких наука, одсек Рачунарство и аутоматика. Основне академске студије је завршио 2013. године. Након завршених основних студија 2013. године уписао је мастер студије на истом факултету смер Софтверско инжењерство и информационе технологије.

**GENAN – DSL ZA OPIS KORISNIČKOG INTERFEJSA WEB APLIKACIJA
UPOTREBOM TEXTX METAJEZIKA****GENAN - DSL FOR WEB APPLICATION USER INTERFACE SPECIFICATION BASED
ON TEXTX META LANGUAGE**

Bojana Zoranović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu je predstavljena implementacija jezika specifičnog za domen GenAn kao i realizovan generator koda za ovaj jezik. Jezik je kreiran upotrebom TextX metajezika. GenAn je razvijen sa ciljem da podrži jednostavno opisivanje izgleda korisničkog interfejsa dinamičkih web aplikacija. Generator koda je namenjen generisanju MEAN stek aplikacija (MongoDB, Express.js, AngularJS, Node.js).

Abstract – This paper presents implementation of domain-specific language GenAn along with code generator developed for this language. GenAn was developed using TextX meta-language. Main goal for GenAn language is to provide simple GUI description for dynamic web applications. Code generator generates MEAN stack applications (MongoDB, Express.js, AngularJS, Node.js).

Ključne reči: DSL, TextX, code generation, GUI

1. UVOD

Od početka 2000ih godina broj korisnika interneta vrtoglavo raste [1], a sa njime i očekivanja korisnika od web aplikacija. Ovaj trend nameće prebacivanje dela obrada, provera i logike u klijentske aplikacije u cilju rasterećenja serverskih aplikacija i same komunikacije između klijenta i servera.

Usloznavanjem klijentskih aplikacija dolazi do većeg opterećenja programera koji moraju da savladaju veliki broj formalizama za opis različitih segmenata klijentskih aplikacija. Programerima je potreban mehanizam za konzistentan opis aplikacija na visokom nivou apstrakcije.

Jezici specifični za domen [2] nude povećanje produktivnosti i poboljšanje kvaliteta softverskog proizvoda uz opise na visokom nivou apstrakcije usko vezane za jezik oblasti za koju se aplikacija kreira pa u određenom meri rešavaju predhodno opisan problem.

Generator koda [3] je automatski proces koja pristupa modelima, izvlači informacije iz njih i transformiše ih u kod spreman za interpretiranje ili kompajliranje u izvršni kod. Generisan kod je kompletan, izvršiv, produkcionog kvaliteta i ne traži prepravljavanja ni dodavanja. Procesom generisanja upravlja metamodel (jezik za modelovanje sa pripadajućim konceptima, semantikom i pravilima) i

ulaznom sintaksom potrebnu domenskom framework-u i ciljnom okruženju. Domenski framework [3][4] predstavlja osnovu za implementaciju softverskog rešenja i arhitekturu u koju se razvojni tim mora ukopiti. Između ostalog, sprečava dupliciranje generisanog koda i obezbeđuje integraciju sa postojećim kodom.

2. PREGLED KORIŠĆENIH TEHNOLOGIJA

Arpeggio parser [5] je rekurzivni, silazni parser sa vraćanjem i memoizacijom (packrat parser). Razvijen je na Fakultetu tehničkih nauka u Novom Sadu od strane tima predvođenim profesorom Igorom Dejanovićem. Dostupan je na GitHub-u pod MIT licencom. Implementiran je u programskom jeziku Python. Arpeggio ima mogućnost definisanja gramatika pomoću PEG notacije ili programskog jezika Python. Kreiran je sa ciljem da pruži podršku u procesu razvoja jezika specifičnih za domen – preciznije kao infrastruktura za parsiranje za TextX alat.

TextX [6] je metajezik za kreiranje jezika specifičnih za domen napisan u programskom jeziku Python baziran na Arpeggio parseru. Kao i Arpeggio, razvijen je na katedri za informatiku Fakulteta tehničkih nauka u Novom Sadu. Glavna namena TextX metajezika je da olakša i podrži razvoj tekstualnih jezika, pružajući mogućnost za razvoj sopstvenog jezika kao i izgradnje podrške za već postojeći jezik ili određeni format datoteke.

MEAN [7] je full-stack JavaScript rešenje za kreiranje brzih i robustnih web aplikacija korišćenjem MongoDB, Express.js, AngularJS i Node.js (od čijih početnih slova je i potekao naziv).

MongoDB [8] je jedna od vodećih NoSQL baza podataka. Podaci se u MongoDB čuvaju u obliku JSON dokumenata sa dinamičkim šemama.

Express.js [9] je projekat Node.js fondacije i predstavlja fleksibilan i minimalan radni okvir za razvoj Node.js web aplikacija.

AngularJS [10] predstavlja open-source JavaScript radni okvir (framework) za pisanje skalabilnih dinamičkih web aplikacija. Jedan od glavnih ciljeva radnog okvira je da prevaziđe neke od izazova prilikom kreiranja single-page aplikacija i olakša razvoj i testiranje pružajući osnovu za klijentske MVC (Model-View-Controller) i MVVM (Model-View-View-Model) arhitekture.

Node.js [11] je JavaScript izvršno okruženje otvorenog koda za razvoj različitih alata i aplikacija. Node.js ima arhitekturu orijentisanu ka događajima (event-driven

NAPOMENA:

Ovaj rad proistekao je iz diplomskog-master rada čiji mentor je bila dr Gordana Milosavljević.

architecture) sa neblokirajućim, asinhronim procesiranjem ulaza i izlaza.

3. IMPLEMENTACIJA SISTEMA

Referentna implementacija se sastoji od jezika specifičnog za domen i generatora web aplikacija. U ovom trenutku, podržano je generisanje MEAN steka aplikacija.

Kao metajezik je korišćen TextX alat koji je značajno olakšao razvoj ovog jezika.

3.1. GenAn DSL

Apstraktna sintaksa GenAn DSL-a je određena metamodelom – gramatikom.

GenAn DSL je dizajniran kombinovanjem koncepata vezanih za opis modela podataka i koncepata za opis korisničkog interfejsa web aplikacija.

Korenski pojam u gramatici GenAn DSL-a je modul (module). Modul se sastoji od definicije koncepata. Osnovni koncepti ovog jezika su Object, View, Page, Menubar i Sidebar. Navedeni koncepti se mogu podeliti na dve velike grupe:

1. Koncepti vezani za opis modela podataka (Object)
2. Koncepti vezani za opis korisničkog interfejsa web aplikacija (View, Page, Menubar i Sidebar).

Osnovni koncept vezan za model podataka je koncept objekta (Object) i predstavlja jedan entitet u sistemu. Objekat se deklarise navođenjem ključne reči object, zatim naziva objekta, metapodataka, svojstava (properties) i upita (queries) što je prikazano na slici 3.1.1.

```
Object:
  'object' name=ID ':'
  ('Meta' ':' meta += Meta ';')?
  properties *= Property
  queries *= Query
  ','
;
```

Slika 3.1.1. Koncept Object

U metapodatke objekta spadaju podaci o vezama datog entiteta sa drugim entitetima. U dizajnu se težilo pojednostavljenju ovih koncepata. Odabirom tipa stranog ključa (ForeignKeyType) korisnik može da specificira kardinalitet veze (single koja označava vezu jedan-prema-jedan ili list koja označava vezu jedan-prema-više). S obzirom da se svaka veza više-prema-više može svesti na veze jedan-prema-više, izostavljena je iz gramatike. Posle toga, korisnik navodi drugi entitet (objekat – Object) sa kojim se veza uspostavlja praćen ključnom rečju by i navođenjem kolekcije svojstava drugog entiteta koji se prikazuju. Na kraju se navodi naziv labela koja će se prikazati uz ovaj podatak

Svojstva objekta (properties) nose podatke vezane i za model podataka (poput naziva i tipa) i za način prikazivanja na korisničkom interfejsu (labela), slika 26. Postoji podrška i za navođenje dodatnih atributa poput HTML klasa sa kojim se svojstvo prikazuje, ponuđene

vrednosti (ukoliko su opcije u obliku enumeracije), link do slike i sl.

Tip svojstva objekta je jedan od ugrađenih tipova (TextX built-in object [12]). Ugrađeni tipovi se ne posmatraju samo u kontekstu modela već i u kontekstu prikaza.

U koncepte vezane za opis korisničkog interfejsa web aplikacija spadaju View, Page, Form, Menubar, Sidebar i Footer.

Pogled (view) predstavlja često korišćen deo korisničkog interfejsa izgrađen od drugih podpogleda (SubView). Ideja pogleda je da omogući korisniku da dizajnira tipične prikaze na jednom mestu i da ih posle samo poziva po potrebi. Važan pojam vezan za pogled je selektor (Selector). Selektor se može posmatrati kao referenciranje već postojećeg pogleda ili forme, pristup podacima o entitetu (sa navođenjem svojstava koja se žele prikazati), pristup entitetu na osnovu stranog ključa, link do stranice i slično. Prilikom specifikacije podpogleda unutar pogleda postoji mogućnost određivanja pozicije i veličine svakog podpogleda.

Stranica (Page) predstavlja nezavisnu web stranicu aplikacije. Stranica poseduje naziv, podatke o entitetima koji se prikazuju (uz opciono filtriranje), podatak o tome da li je u pitanju inicijalna stranica, naslov, podatke o korišćenim zajedničkim komponentama interfejsa, pogledima koje sadrži i njihovom rasporedu.

Forma (Form) je komponenta korisničkog unosa koja služi za korisničku manipulaciju podacima. U klijentskoj aplikaciji je predstavljena HTML formom uz potrebne dopune vezane za konkretnu tehnologiju. Forma se kreira navođenjem ključne reči form zatim naziva entiteta (objekta) nad kojim se vrši manipulacija i na kraju set akcija koje su dostupne korisniku

3.2. Generator

Pre samog početka generisanja vrši se provera sintakse mehanizmima koje obezbeđuje TextX – obrađivačima objekata (object processors) koji omogućuju implementiranje specifičnih provera za svaki od koncepata metamodela.

Generator koda GenAn jezika specifičnog za domen se sastoji od dva nezavisna dela – generatora klijentske i generatora serverske aplikacije. Klijentska i serverska aplikacija komuniciraju preko HTTP-a u skladu sa pravilima REST-a što ih čini nezavisnim i lako zamenljivim. Ove rute se kreiraju automatski, na osnovu opisa entiteta (object) u modelu. Za sada su podržane osnovne, CRUD (Create-Read-Update-Delete) operacije i postavljanje upita (filtriranje).

U prvobitnoj verziji ovog alata generiše se AngularJS klijentska aplikacija i Node.js serverska aplikacija sa MongoDB bazom podataka (MEAN stek aplikacija).

Generator koda GenAn jezika specifičnog za domen spada u generatore koda bazirane na visitor dizajn šablonu. Struktura Python klasa koju TextX alat generiše na osnovu modela je struktura tipa stabla. Svaki element

nosi podatke o načinu na koji se obrađuje i načinu na koji se prevodi u ciljni kod. Visitor šablon omogućuje sa jedne strane uniforman prolazak kroz stablo elemenata, a sa druge strane rasterećuje kod generatora prebacujući kontrolu nad načinom generisanja na klase pojedinačnih elemenata. Generisanje se obavlja na bazi Jinja2 template engine-a.

Prvi korak u procesu generisanja je instanciranje TextX interpretera. U toku instanciranja podešavaju se ugrađeni tipovi (built-in object) TextX metajezika. Određivanjem ugrađenih tipova se štedi vreme korisnika (korisnik ne mora da ih definiše u svakom modelu) i smanjuje se broj linija koda potrebnih za opis modela. U GenAn DSL-u, ugrađeni tipovi se odnose na predefinisane tipove pogleda.

Object. Generisanje objekta počinje od provere stranog ključa (proverava se da li u povezanom entitetu postoji svojstvo navedeno kao strani ključ entiteta) i jedinstvenosti naziva veze. Zatim se model menja dodavanjem novog (meta)svojstva koji reprezentuje vezu uz kreiranje pogleda (view) potrebnog za prikazivanje ovog svojstva tj. veze na korisničkom interfejsu. Proces generisanja se zatim nastavlja obradom navedenih upita (query) i prevođenjem u format koji se može poslati kao parametar GET metode HTTP zahteva. Zatim se kreira AngularJS factory koji predstavlja komponentu koja obezbeđuje osnovne CRUD operacije i navedene upite.

Page. Generisanje stranice kreće od dobavljanja referenci na AngularJS factory-je objekata vezanih za stranicu. Zatim se renderuje controller stranice na osnovu postojećeg šablona stranice. Sledeći korak je raspoređivanje pogleda na korisničkom interfejsu aplikacije na osnovu podrazumevanih ili eksplicitno navedenih vrednosti pozicije. U ovom koraku se izvršava i renderovanje i ubacivanje footer-a, sidebar-a i menubar-a čije se pozicije uzimaju u obzir prilikom raspoređivanja komponenti na ekranu. Poslednji korak u generisanju stranice predstavlja dodavanje stranice i njenih podpogleda u rute aplikacije.

Proces generisanja serverske aplikacije započinje čitanjem konfiguracionog fajla koji sadrži podatke o mapiranju GenAn tipova pogleda na Mongo tipove. Zatim se instaliraju svi potrebni node moduli kako bi generisana aplikacija bila odmah spremna za rad. Fokus serverske aplikacije je rad sa objektima. Za svaki od objekata se prvo kreira Mongoose šema na osnovu koje se podaci čuvaju u bazi. Mongoose predstavlja rešenje bazirano na šemi za modelovanje podataka u Node.js aplikacijama. Uključuje i ugrađeno kastovanje (type-cast) podataka, validaciju, kreiranje upita i sl.

Za svaki objekat se kreira posebna node route datoteka sa posebnim Express router-om pa zbog toga svaki objekat predstavlja modul, nezavistan od ostatka Node.js aplikacije. Tako formiran router modul se dodaje na rute glavne aplikacije. Svaka metoda router-a izvedena je iz

jedne od HTTP metoda (GET, POST, PUT, DELETE). Svaka od ovih metoda mapirana je na jednu od callback funkcija definisanih u direktorijumu route_callbacks. Za svaki od objekata formira se posebna datoteka sa callback funkcijama. Podržane funkcije su:

- **getAll.** Dobavlja više instanci traženog objekta. Kreirana je tako da podržava korisničke upite sa klijentske aplikacije od kojih formira upit za bazu podataka.
- **getById.** Na osnovu zadatog identifikatora pronalazi traženu instancu.
- **Post.** Kreira instancu sa prosleđenim podacima u bazi podataka.
- **deleteById.** Na osnovu zadatog identifikatora uklanja instancu iz baze podataka ukoliko postoji.
- **putById.** Na osnovu zadatog identifikatora menja sadržaj instance iz baze podataka prosleđenim.

4. STUDIJA IZVODLJIVOSTI

U okviru rada prikazana je studija izvodljivosti koja ima za cilj da dodatno pojasni mogućnosti opisanog softverskog rešenja.

Na osnovu GenAn opisa datom na listinzima 4.1. i 4.2. generisana je aplikacija.

```
menubar mainMenu 'GenAn' @home:
  menu User:
    'User Form' @user_form_page
  ;
;

sidebar mainSidebar:
  'Home' @home
  'GitHub' ref='https://github.com/theshammy/GenAn'
  ;

footer mainFooter 'Copyright MIT, organization GenAn ©' right color=black ;

object user:
  username text label='User name'
  mail_email label='Email'
  gender combobox label='Gender' parameters = { 'Male', 'Female' }
;

page user_form_page:
  title='User form'
  use menubar mainMenu
  show user_form_view position=center
;

view user_form_view:
  form for user actions=save update remove view at 3 size 6
;
```

Listing 4.1. GenAn opis objekta user i stranice user_form

U okviru aplikacije kreiran je jedan objekat korisnik (user) sa svojstvima korisničko ime (username) GenAn tipa text, adresa elektronske pošte (email) GenAn tipa email i pola (gender) GenAn tipa combobox. Zatim, opisana je stranica *user_form_page* koja sadrži formu za unos, izmenu, brisanje i pregled jednog korisnika.

```

page home:
  indexPage=True
  title='Home'
  use menubar mainMenu
  use sidebar mainSidebar
  use footer mainFooter
  jumbo: heading='Welcome'
    text='Lorem ipsum dolor sit amet,
    consectetur adipiscing elit. Donec interdum ipsum vitae
    odio congue, in placerat mauris congue. Sed imperdiet sodales finibus.'
    imagePath='http://www.planwallpaper.com/static/images/518079-background-hd.jpg'
    position=top
  paragraph 'Lorem ipsum dolor sit amet,
  consectetur adipiscing elit. Donec interdum ipsum vitae
  odio congue, in placerat mauris congue. Sed imperdiet sodales finibus.
  Lorem ipsum dolor sit amet,
  consectetur adipiscing elit. Donec interdum ipsum vitae
  odio congue, in placerat mauris congue. Sed imperdiet sodales finibus.'
  position=left
  link 'Read more' ref='#' position=left
  paragraph '&nbsp;' position=left
  paragraph '&nbsp;' position=left
  paragraph 'Lorem ipsum dolor sit amet,
  consectetur adipiscing elit. Donec interdum ipsum vitae
  odio congue, in placerat mauris congue. Sed imperdiet sodales finibus.
  Lorem ipsum dolor sit amet,
  consectetur adipiscing elit. Donec interdum ipsum vitae
  odio congue, in placerat mauris congue. Sed imperdiet sodales finibus.'
  position=center
  link 'Read more' ref='#' class='{"style":"float:right;"}' position=center

```

Listing 4.2. GenAn opis stranice home

Kreirana je još jedna stranica *home* koja predstavlja početnu stranicu aplikacije. U njoj su korišćeni menubar, sidebar i footer – standardne grafičke komponente opisane u listingu 4.1. Sastoji se od velikog panela (jumbo) sa natpisom *Welcome* i 2 paragrafa. Generisane stranice na osnovu ovih zapisa date su na slikama 4.3. i 4.4.

GenAn User

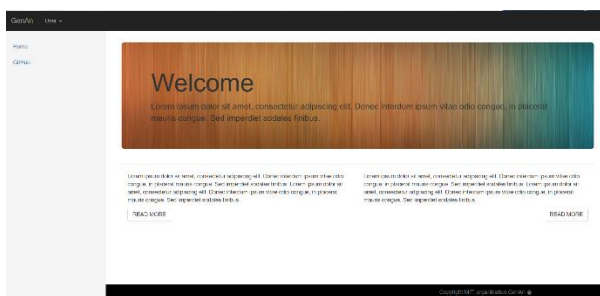
USER NAME
username1

EMAIL
username1@mail.com

GENDER
Male

Save Update Remove View

Slika 4.3. Generisana forma za objekat user



Slika 4.4. Generisana *home* stranica

5. ZAKLJUČAK

Zadatak ovog rada je bio da prikaže implementaciju jezika specifičnog za domen GenAn namenjenog opisu izgleda grafičkog korisničkog interfejsa web aplikacija i generatora koda za ovaj jezik. GenAn jezik je namenjen programerima kao domenskim ekspertima i omogućava

im brzo postavljanje osnova web aplikacije uz podršku za detaljno opisivanje korisničkog interfejsa. Generator je plagabilan i dozvoljava korisniku da integriše sopstvene generatore koda za druge platforme. Na osnovu opisa modela, generiše se i klijentska i serverska aplikacija posebnim, nezavisnim generatorima što daje veću fleksibilnost i slobodu korisniku koji može da reinplementira samo jedan od ova dva generatora. Generatori su implementirani u skladu sa postojećim konvencijama uz podršku za validaciju, prevođenje i definisanje upita.

6. LITERATURA

- [1] Internet Live Stats – Internet Users, <http://www.internetlivestats.com/internet-users/>
- [2] M. Voelter, DSL Engineering: Designing, Implementing and Using Domain-Specific Languages, 2010-2013.
- [3] S. Kelly, J.-P. Tolvanen, Domain Specific Modeling: Enabling Full Code Generation, Wiley-IEEE Computer Society Press, 2008
- [4] G. Milosavljević, materijali sa predmeta Metodologije brzog razvoja softvera, Fakultet tehničkih nauka, Novi Sad, 2016
- [5] I. Dejanović, G. Milosavljević, R. Vaderna, Arpeggio: A flexible PEG parser for Python, 2015
- [6] Zvanična dokumentacija za TextX, <http://igordejanovic.net/textX/>
- [7] MEAN stack, <http://mean.io/>
- [8] MongoDB, <https://www.mongodb.com/>
- [9] Express.js, <https://expressjs.com/>
- [10] AngularJS, <https://angularjs.org/>
- [11] Node.js, <https://nodejs.org/en/>
- [12] TextX built-in objects, <http://igordejanovic.net/textX/metamodel/#built-in-objects>

Kratka biografija

Bojana Zoranović je rođena 25.4.1992. godine u Novom Sadu. Osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu, odsek Elektrotehnika i računarstvo, smer Računarstvo i automatika upisala je 2011. godine. Osnovne akademske studije završava 2015. godine sa prosečnom ocenom 10,00. Master akademske studije Fakulteta tehničkih nauka, odsek Elektrotehnika i računarstvo, smer Računarstvo i automatika, usmerenje Softversko inženjerstvo upisuje školske 2015/2016. Sve ispite predviđene planom i programom je položila sa prosečnom ocenom 10,00.

ANDROID APLIKACIJA ZA PLAĆANJE STUDENTSKIH TROŠKOVA POSREDSTVOM PAYPAL SISTEMA**ANDROID APPLICATION FOR PAYMENT OF STUDENT EXPENSES USING THE PAYPAL SYSTEM**

Ivan Jašo, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U radu je predložen sistem za plaćanje troškova prijave ispita na fakultetu, putem Android mobilnih uređaja, posredstvom PayPal sistema za elektronsko plaćanje. Sistem omogućava pregled i prijavu ispita, kao i plaćanje i refundaciju troškova. Kao okvir za implementaciju korišćeni su AndroidSDK, Spring Boot i PayPal MobileSDK, a za komunikaciju sa servisima korišćen je REST protokol, a kao sigurnosni protokol korišćen je OAuth2.

Abstract – This paper proposes a system for paying college exam application expenses using the Android mobile device through the PayPal payment system. The system allows the user to list and apply for the exam as well as payment and refund of the expenses. Frameworks used in the implementation of this system are AndroidSDK, Spring Boot and PayPal MobileSDK and as a protocol for the communication with the services, REST protocol was used. OAuth2 was used as a security protocol.

Ključne reči: Studentski servis, Android, Elektronsko plaćanje, PayPal

1. UVOD

Uspeh elektronske trgovine je stvorio potrebu da se razvije sistem elektronskog plaćanja za plaćanje robe i usluga. Sistem plaćanja kreditnom karticom je najpopularniji način da se izvrši kupovina u elektronskoj trgovini. Glavni problem u vezi njegove upotrebe je što klijent daje svoje lične informacije i broj kreditne kartice trgovcu ili pružaocu usluge plaćanja kada kupuje na internetu i time gubi anonimnost i privatnost na internetu. Osim toga, javljaju se problemi i nepoverenje klijenata i zbog čestog nedostatka bezbednosti na javnoj mreži, ličnim računarima, serverima trgovaca, kao i propusta u obezbeđivanju samih web aplikacija.

Klijentima elektronske trgovine su u sistemu elektronskog plaćanja bile potrebne anonimnost, primenljivost, poverenje, konvertibilnost, efikasnost i pouzdanost [1].

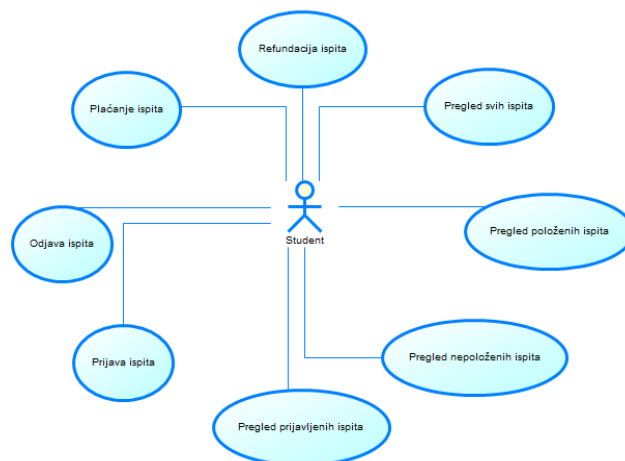
PayPal je trenutno najpopularniji sistem elektronskog plaćanja. Izgrađen je na postojećoj finansijskoj infrastrukturi bankovnih računa i kreditnih kartica.

Glavna prednost PayPal-a jeste to što se ponaša kao posrednik u kupovini. Kupac zahteva stvaranje naloga u sistemu i kada dobije nalog on može da pristupi sredstvima i uslugama koje nudi PayPal. PayPal skladišti lične podatke korisnika kao što su: ime, prezime, adresa, grad, republika, poštanski broj, država i kućni telefon, ali i informacije o njegovim kreditnim karticama: vrsta kreditne kartice, ime na kartici, broj kartice i datum isteka [1].

2. MODEL SISTEMA

U ovom odeljku predstavljen je model sistema za pregled, prijavu, plaćanje, odjavu i refundaciju ispita preko Android aplikacije, u kojoj se sve platne transakcije vrše preko PayPal sistema.

Na slici (Slika 1) su prikazane moguće interakcije između studenta i sistema. Studenti imaju mogućnost da se prijave na sistem i da obavljaju operacije vezane za pregled, prijavu, odjavu, plaćanje i refundaciju ispita.



Slika 1. Use case dijagram korišćenja sistema od strane studenta

2.1. Autentifikacija i autorizacija korisnika

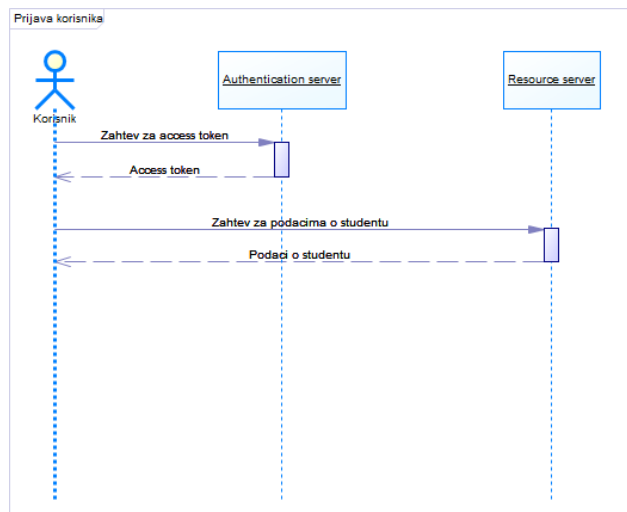
Da bi korisnik mogao da koristi sistem administratori moraju ubaciti podatke o njemu u bazu za autentifikaciju i autorizaciju, gde se nalaze samo njegovi kredencijali i uloga za pristup sistemu. Bez ovih podataka autentifikacioni server neće biti u mogućnosti da izdaje access tokene koji su obavezni za pristup bilo kom servisu na resource serveru.

NAPOMENA:

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

Ostali podaci o studentu, među kojima su broj indeksa, godina upisa, godina studija itd., moraju se ubaciti u bazu za *resource server*. Tek tada student je u mogućnosti da pristupi aplikaciji.

Ceo tok prijave studenta na aplikaciju prikazan je na dijagramu sekvenci za prijavu korisnika (Slika 2). Kada korisnik otvori mobilnu aplikaciju biće mu prikazan formular za popunjavanje kredencijala za prijavu, a to su: skraćeni naziv smeru, redni broj upisa, godina upisa i lozinka. Ovi kredencijali, sem lozinke, ujedno čine i broj indeksa studenta koji predstavlja jedinstveni broj studenta na fakultetu.



Slika 2. Dijagram sekvenci za prijavu korisnika

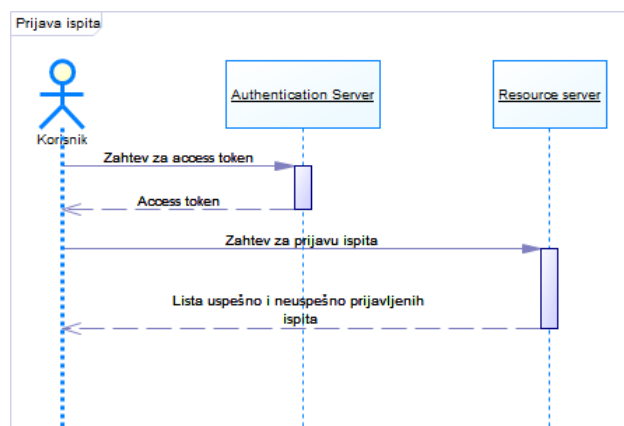
Kada korisnik unese kredencijale i potvrdi svoj unos aplikacija šalje zahtev za *access token* sa datim kredencijalima ka *authentication serveru* koji pronalazi korisnika i za njega vraća jedinstven *access token*. Aplikacija zatim uz pomoć dobijenog *access tokena* poziva *resource server* kako bi dobio ostale informacije o studentu, kao što su generisani identifikacioni broj studenta (*idStudenta*) u *resource bazi*, ime i prezime i ostali osnovni podaci o studentu. Od ovih podataka najvažniji je *idStudenta* jer se koristi u pozivima određenih servisa na *resource serveru*.

2.2. Prijava ispita

Ukoliko korisnik odabere opciju za prijavu ispita na glavnom meniju, biće mu prikazan ekran sa listom ispita koje može da prijavi u tekućem ispitnom roku. Kada korisnik odabere ispite iz liste i potvrdi svoj izbor prvo se šalje standardni zahtev za *access token authentication serveru* [4], koji se dalje koristi za pozivanje servisa za prijavu ispita na *resource serveru* (Slika 3). Servis zatim prijavljuje ispite u ime korisnika i pravi odgovarajuća zaduženja na osnovu trenutno aktivnog cenovnika usluga fakulteta. Sistem zatim proverava validnost prijave i ako su uspešno prijavljeni ispiti aplikacija nudi korisniku da odmah i isplati prijavljene ispite. Ako je prijava bila neuspešna, korisnik ostaje na formi za izbor predmeta za prijavu uz poruku o neuspehu i tu može da pokuša da ponovi proces.

Ukoliko je korisnik odabrao da želi da odmah isplati prijavljene ispite, aplikacija mu prikazuje PayPal ekran za

autentifikaciju korisnika i autorizaciju uplate. Nakon uspešne uplate korisnik će biti vraćen na ekran za prijavu ispita, gde može prijaviti ostale neprijavljene ispite.

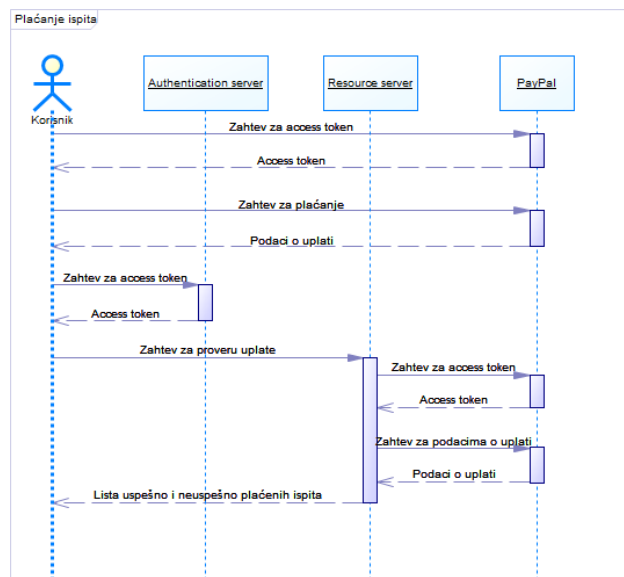


Slika 3. Dijagram sekvenci za prijavu ispita

Ako je korisnik odabrao da plaćanje izvrši kasnije, korisnik će biti vraćen na ekran za prijavu ispita, gde može prijaviti ostale neprijavljene ispite.

2.3. Plaćanje prijave ispita

Ukoliko korisnik stigne do toka za plaćanje tako što pri prijavi ispita odabere da troškove prijave plati odmah on preskače prva dva koraka, dakle, odabir ispita i potvrdu i aplikacija odmah prelazi na sumiranje troškova (Slika 4).



Slika 4. Dijagram sekvenci za plaćanje ispita

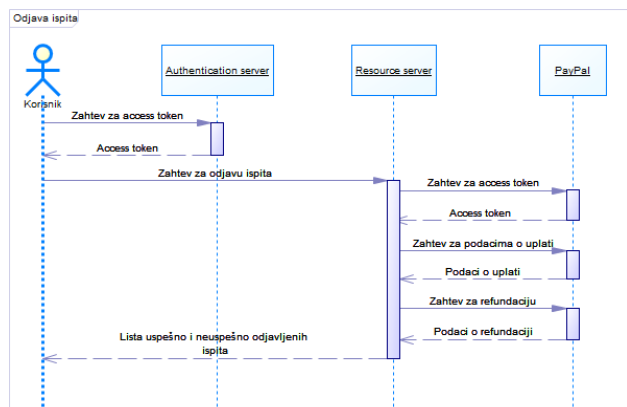
Ako korisnik nije želeo odmah da isplati svoje troškove, može to da uradi naknadno tako što će iz glavnog menija odabrati plaćanje prijave ispita. Tog momenta aplikacija će mu prikazati ekran na kojem će stajati lista svih neplaćenih prijava.

Korisnik na ovoj formi bira prijave koje želi da isplati i zatim potvrđuje svoj izbor. Nakon toga aplikacija sumira troškove i otvara PayPal ekran za autentifikaciju. Kada se korisnik autentifikuje koristeći PayPal kredencijale, aplikacija će mu prikazati troškove i tražiti potvrdu plaćanja.

Aplikacija zatim vrši plaćanje putem PayPala pri čemu kao odgovor dobija podatke o toj transakciji. Jedan od tih podataka je i *paymentId* koji aplikacija zatim, nakon preuzimanja *access tokena*, prosleđuje *resource serveru* koji uz pomoć PayPal REST servisa verifikuje uplatu i vezuje *paymentId* za svaku prijavu koja je bila plaćena tom transakcijom [3].

2.4. Odjava ispita

Ukoliko korisnik, koji je prijavio ispite u tekućem roku, ipak odluči da ne izađe na neki od prijavljenih ispita, ima mogućnost da taj ispit odjavi, čak iako je uplata prijave već izvršena, ali samo ako je još unutar graničnog roka do kog ima pravo odjave ispita (Slika 5).



Slika 5. Dijagram sekvenci za odjavu i refundaciju ispita

Odjavu ispita može izvršiti tako što će iz glavnog menija odabrati opciju objave ispita. Aplikacija će mu tad prikazati ekran za odjavu ispita sa listom svih prijavljenih ispita u tekućem ispitnom roku.

Student bira koje od prijavljenih ispita želi da odjavi i nakon toga potvrđuje svoj izbor. Aplikacija šalje odabrane ispite *resource serveru*, koji zatim proverava da li među odabranim prijavama ima već plaćenih prijava.

Ukoliko ima plaćenih prijava, sistem na osnovu *paymentId*-a koji je upisan u date prijave kontaktira PayPal servise kako bi dobio ostale podatke o toj uplati. Među njima se nalazi *saleId*, koji je potreban pri slanju zahteva za refundaciju PayPal servisu.

Sistem zatim šalje zahtev za refundaciju PayPal-u zahtevajući da se vrati suma koja je zabeležena kao suma koju je korisnik i platio pri plaćanju ove prijave.

Ako je refundacija uspešna sistem upisuje podatak o refundaciji u to zaduženje i time poništava prijavu. Sve neplaćene prijave se samo invalidiraju (poništavaju) u sistemu bez kontaktiranja PayPal-a.

3. IMPLEMENTACIJA SISTEMA

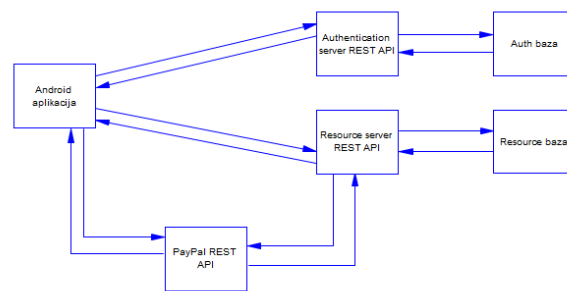
Za implementaciju predloženog modela sistema koristi se Java programski jezik, Spring Boot okruženje i AndroidSDK. Servisni deo je podeljen u dva dela i to na *Authentication server* i *Resource server* (Slika 6). *Authentication server* služi za autentifikaciju i autorizaciju korisnika kako bi mogli da pristupe servisima na *Resource serveru*. Oba koriste MySQL drajver za pristupanje MySQL bazama i Hibernate kao ORM alat za mapiranje Java entiteta na tabele u bazi.

Svi pozivi i odgovori servisa su u JSON formatu, a za mapiranje objekata na JSON zahteve i odgovore Spring Boot koristi Jackson JSON mapper biblioteku.

Za komunikaciju sa servisima na *Authentication serveru* i *Resource serveru*, kao i u samoj Android aplikaciji koristi se Spring OAuth2 Cloud modul. On omogućava da se uz svaki REST [5] poziv prosleđuje i *access token*, koji je neophodan za sve servise u ovom sistemu.

Kako MobileSDK trenutno ne podržava refundacije [2], *Resource server* takođe komunicira sa PayPal sistemom preko REST servisa, kako bi se obavila ova transakcija, ali pri tom ne koristi PayPal API jer bi bilo suviše ubacivati celu biblioteku kako bi koristili samo jednu funkciju, tako da će se za tu komunikaciju koristiti obični REST pozivi sa ubačenim *access tokenom*. O ovome vodi računa Spring OAuth2 Cloud modul.

Spring Boot biblioteka omogućava pravljenje tzv. *Uber Jar* fajlova od projekta koji u sebi sadrži i ugnježđen server (embedded server), pa ga je moguće pokrenuti kao običnu aplikaciju pri čemu on sam pokreće server na kojem će raditi aplikacija.



Slika 6. Arhitektura sistema

2.1. Autentifikacija i autorizacija korisnika

Svi servisi na *Resource serveru* su zaštićeni i zahtevaju *access token* za pristup. *Access token* izdaje *Authentication server*, koji na osnovu kredencijala korisnika ili *refresh tokena* izdaje novi *access token*, koji se može iskoristiti na *Resource serveru*. *Resource server* svaki *access token* verifikuje na *Authentication serveru* pre pristupa bilo kom servisu.

U zahtevu je potrebno poslati *client_id* i *client_secret* u obliku Base64 hash-a, parametar *grant_type* sa vrednošću *password*, parametar *username* sa vrednošću korisničkog imena (u ovom slučaju je to broj indeksa, kao jedinstveni broj studenta) i parametar *password* sa vrednošću lozinke tog korisnika.

Na isti URL se šalje zahtev za *access token* i sa *refresh tokenom*, s tom razlikom da se sada ne šalju kredencijali korisnika, dakle ne šalju se parametri *username* i *password*, već se umesto toga šalje parametar *refresh_token* sa vrednošću samog *refresh tokena*, a parametar *grant_type* dobija vrednost *refresh_token*.

2.2. Prijava ispita

Kada korisnik odabere ispite koje želi da prijavi i klikne akciono dugme za prijavu ispita, lista ispita se prosleđuje servisu za prijavu ispita POST metodom.

Kao parametri se osim *access tokena* šalju i *id* studenta i lista *id*-eva ispita koje je korisnik odabrao.

Za svaki ispit se proveriti da li student ima pravo da ga prijavi i ako ima formira se prijavljen ispit za čuvanje u bazi, a ujedno se stvara i zaduženje za prijavu ispita po trenutnom cenovniku usluga fakulteta.

2.3. Plaćanje prijave ispita

Postoje dva puta kojima korisnik može doći do plaćanja ispita. Prvi je da kada prijavi ispit odabere opciju da odmah i plati te prijave, a drugi je da iz glavnog menija odabere plaćanje ispita.

Oba puta svode se na istu proceduru, a to je da se obračuna ukupna cena svih prijava koje korisnik želi da isplati i da se onda za tako sumiranu cenu pokrene proces plaćanja preko PayPal MobileSDK ekrana za plaćanje. Ovom ekranu potrebno je proslediti valutu u kojoj se vrši naplata, ukupna vrednost novca koji se potražuje i naziv proizvoda ili usluge koja se naplaćuje.

Nakon pokretanja ekrana za naplatu i autentifikacije korisnika, korisnik verifikuje plaćanje, posle čega PayPal servisi vraćaju podatke o uplati, među kojima je i *paymentId*, koji koristimo za poziv ka *Resource serveru* da verifikuje uplatu i veže je za prijavljene ispite za koje je plaćena prijava.

Verifikacija uplate se vrši tako što se pozove *check payment* servis na PayPal-u i ako stanje plaćanja ima vrednost „complete“ i ako stanje prodaje unutar tog plaćanja ima vrednost „approved“, onda je uplata uspešna i može se vezati za prijavljene ispite.

2.4. Odjava ispita

Kada korisnik odabere ispite koje želi da odjavi i klikne akciono dugme za odjavu ispita, lista odabranih ispita se POST metodom prosleđuje servisu.

Za svaki ispit se prvo vrši provera da li je izvršeno plaćanje prijave datog ispita, ako nije onda se samo invalidira (poništi) prijava, a ako je već izvršeno plaćanje onda se prvo odradi refundacija sredstava za dati ispit i ako je refundacija uspešna onda se takođe invalidira sama prijava.

Da bi se izvršila refundacija ispita, potrebno je na osnovu *paymentId*-a koji je sačuvan u prijavi pronaći transakciju na PayPal-u koristeći servis koji smo koristili i za proveru uplate na PayPalu, dakle *payment check* servis i iz njegovog odgovora izvući *saleId*, koji nam je potreban za pozivanje servisa za refundaciju.

Servis za refundaciju uplate osim parametra *saleId* kao obavezne parametre prima valutu plaćanja, kao i sumu koja treba da se refundira. PayPal, naravno, ima proveru da li je suma koju pokušavamo da refundiramo veća od sume koja je uplaćena kako ne bi došlo do malverzacija.

4. ZAKLJUČAK

Većina univerzitetskih ustanova trenutno omogućava svojim studentima neki način obavljanja administrativnih obaveza elektronskim putem.

Neke su ograničene platnim sistemima na nivou države, a neke ne. Međutim, ne može se zanemariti nagli rast popularnosti mobilnih aplikacija, te se prirodno javlja potreba za integracijom svih postojećih web servisa sa mobilnim aplikacijama. Ovakva integracija olakšava korisnicima pristup ovim servisima bez obzira gde se nalazili, dok god imaju pristup internetu što je u eri „pametnih“ telefona postao standard.

PayPal predstavlja vodeće rešenje kojim je moguće realizovati elektronsko plaćanje bez rizika od prevare, jer je jedna od bitnijih karakteristika ovog sistema visok nivo zaštite kako ličnih, tako i bankovnih podataka, poput broja kreditne kartice korisnika.

Prilikom plaćanja podaci o kreditnoj kartici se niti prikazuju niti traže od korisnika. Oni su uneti pri otvaranju naloga na PayPal sistemu i vezani za korisnika pa ih nije potrebno overavati prilikom svake transakcije. Umesto toga korisnik koristi svoje PayPal kredencijale da proveriti i overi svaku transakciju.

Jedan od mogućih pravaca daljeg razvoja implementiranog rešenja jeste proširenje sistema tako da podrži overu semestra i plaćanje ostalih troškova kao što su overa semestra i plaćanje školarine, što bi se moglo implementirati uz male izmene ponuđenog rešenja, ali i integracija sa postojećim sistemima fakulteta.

5. LITERATURA

- [1] PayPal - The PayPal Wars: Battles with eBay, the Media, the Mafia, and the Rest of Planet Earth, Eric M. Jackson
- [2] PayPal MobileSDK - Internet: <https://developer.paypal.com/docs/integration/mobile/mobile-sdk-overview>
- [3] PayPal REST API - Internet: <https://developer.paypal.com/docs/api>
- [4] The OAuth 2.0 Authorization Framework - Internet: <https://tools.ietf.org/html/rfc6749>
- [5] Representational state transfer - REST in Practice, Hypermedia and Systems Architecture, Jim Webber, Savas Parastatidis, Ian Robinson

Kratka biografija:



Ivan Jašo rođen je u Novom Sadu 1986. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Softversko inženjerstvo i informacione tehnologije odbranio je 2016.god.

AUTOMATIZACIJA MOTOCIKLA I UPRAVLJANJE PUTEM ANDROID PAMETNOG UREĐAJA

MOTORCYCLE AUTOMATIZATION AND CONTROL VIA ANDROID SMART DEVICE

Petar Gostović, Vladimir Rajs, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu predstavljena je realizacija hardvera i softvera uređaja za automatizaciju motocikla, baziranog na Arduino razvojnim pločama. Realizovani uređaj predviđen je za ugradnju u motocikl i kao takav služi za povećanje bezbednosti u saobraćaju.

Abstract – This paper presents hardware and software realisation of device based on Arduino development system, whose function is to increase safety for motorcycle drivers and prevent motorcycle of theft.

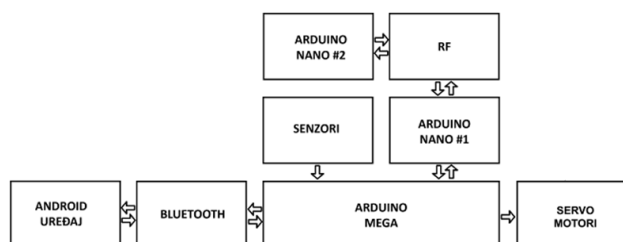
Ključne reči: Arduino, Android

1. UVOD

Ideja za realizaciju rada nastala je kao želja za povećanjem bezbednosti vozača motocikla, ali i zaštita motocikala od krađe. S obzirom da su vozači motocikala najugroženiji učesnici u saobraćaju, potrebno je podići stepen njihove zaštite na što veći nivo. Kako u današnje vreme svako uz sebe ima *Smart* telefon, upravo je on iskorišćen kao glavni alat za upravljanje ovog uređaja.

2. OPIS HARDVERA

Osnovna blok šema realizovanog uređaja prikazana je na slici 1. Upravljanje radom kompletnog uređaja vrši se pomoću razvojnog sistema – *Arduino Mega*, kome funkcije zadajemo preko *Android* uređaja. *Arduino Mega* vrši serijsku komunikaciju sa *Arduino Nano #1* razvojnim sistemom, koji obavlja aktivaciju zvučne i svetlosne indikacije preko *Arduino Nano #2* sistema, sa kojim je povezan preko RF komunikacije.



Slika 1. Blok šema realizovanog uređaja

Centralnu komponentu uređaja za automatizaciju motocikla predstavlja *Arduino Mega* razvojni sistem [1], zasnovan na 8-bitnom Atmega procesoru. Njegov izgled prikazan je na slici 1.1. Osnovne karakteristike ovog mikrokontrolera koje su bile od interesa prilikom realizacije uređaja su:

- Napon napajanja: 5V.
- 8 kB SRAM memorije i 256 kB *flash* memorije.
- Broj digitalnih ulaza/izlaza: 54.
- Broj analognih ulaza: 16.
- Svi ulazno-izlazni pinovi tolerantni na naponski nivo od +5V.



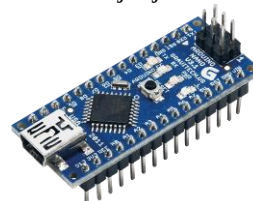
Slika 1.1. Arduino Mega razvojni sistem

2.2. Arduino Nano

Kao i *Arduino Mega*, i ovaj razvojni sistem baziran je na 8-bitnom procesoru Atmega familije, pod oznakom 328P. Odlikuju ga sledeće osnovne karakteristike:

- Napon napajanja: 5V.
- 1 kB SRAM memorije i 32 kB *flash* memorije.
- Broj digitalnih ulaza/izlaza: 14.
- Broj analognih ulaza: 8.
- Svi ulazno-izlazni pinovi tolerantni na naponski nivo od +5V.

Izgled *Arduino Nano*-a vidljiv je na slici 1.2.



Slika 1.2. Arduino Nano razvojni sistem

2.3. Senzori i servo motori

Rad uređaja je podeljen u dve grupe. Prva se odnosi na bezbednost vozača motocikla, a druga grupa obuhvata deo uređaja koji je fokusiran na bezbednost vozila, odnosno motocikla.

Obe navedene grupe imaju skup senzora na osnovu koji se izvršavaju radnje za koje su namenjeni. Deo uređaja koji se odnosi na bezbednost vozača sadrži sledeće senzore:

- Ultrasonični senzori
- Senzor vibracije
- Servo motori

Deo uređaja namenjen sistemu protiv krađe i same bezbednosti motocikla sastoji se od:

- Senzora kiše
- PIR senzora
- Senzora vibracije

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladimir Rajs.

Korišćeni ultrasonični senzori nose oznaku HC-SR04 [4]. Njihov izgled je prikazan na slici 2.1



Slika 2.1. Ultrasonični senzor

Poseđuju svoj prijemnik i predajnik, odnosno *Echo* i *Trigger* kanal. Predajnik neprestano emituje zvučne talase frekvencije 40kHz u intervalima od 10μS. Kada se zvuk odbije od neki objekat, vraća se nazad, prijemnik ga registruje i senzor emituje visok imupls na svom izlazu ka digitalnom ulazu kontrolera. Maksimalan domet ovih senzora iznosi 5m.

Postavljeni su u zadnjem delu motocikla, tako da vrše monitoring kako vozila koje se nalazi striktno iza, tako i vozila koja se nalaze sa leve odnosno desne strane u tzv. 'mrtvom uglu'. Princip je prikazan na slici 2.2.



Slika 2.2. Način postavljanja ultrasoničnih senzora

Ukoliko se vozilo nađe u 'mrtvom uglu' sa leve, odnosno desne strane motocikla, aktivira se led indikacija sa leve odnosno desne strane instrument table, koja u vidu treptenja upozorava vozača sa koje strane mu se nalazi vozilo. Vozilo koje se nalazi iza motocikla i previše se približi i time naruši bezbednu zonu zaustavljanja, pali se zvučna signalizacija, kao i svetlosna signalizacija koja se aktivira na kacigi putem RF komunikacije nanačin opisan u poglavlju 2.6.

Senzor vibracije [3], ili *Tilt* senzor, prikazan je na slici 2.3



Slika 2.3. Senzor vibracije SW-420

Predviđeno je da se nalazi takođe u motociklu, i ukoliko dođe do udara, automatski aktivira vazdušne jastuke, smeštene u jakni vozača motocikla.

Baziran je vibracionom senzoru SW-420 i komparatoru LM393.

Princip rada je veoma jednostavan. Ukoliko dođe do pojave vibracija, kuglica koja se nalazi u senzoru dodiruje njegove zidove i na taj način se generišu impulsi koji preko komparatora odlaze na kontroler. Sama osetljivost senzora može se, kako softverski, tako i preko integrisanog potencijometra veoma lako podešavati.

Servo motori [2], koji su postavljeni u farove motocikla, koriste se za zakretanja sočiva odnosno sijalica. Motori korišćeni u projektu nose oznaku Servo Micro9g SG90, a njihov izgled je prikazan na slici 2.4.



Slika 2.4. Servo Micro9g SG90

Neke od njihovih karakteristika su:

- Napon napajanja: 5VDC
- Struja tokom mirovanja: 2mA
- Struja rada bez opterećenja: 250mA
- Struja rada pri opterećenju: 850mA

Uz pomoć analognog džojstika, povezanog na ulaz kontrolera, vršimo zakretanje servo motora levo-desno.

Džojstik je postavljen na korman motocikla kako bi bio lako dohvativ tokom vožnje. Pomeranjem džojstika u levo, menja se vrednost na analognom ulazu kontrolera i levi far se zakreće srazmerno pomeranju džojstika, dok desni far nastavlja da osvetljava put ispred. Obrnuta situacija važi za pomeranje džojstika u desnu stranu. Puštanjem džojstika farovi se automatski vraćaju u svoj normalan položaj.

Senzor kiše [3], kao što mu samo ime kaže, ima funkciju detekcije kišnih kapi. Korišćeni senzor je prikazan na slici 2.5.



Slika 2.5. Senzor kiše

Njegov princip rada je sledeći: Po njegovoj površini protežu se provodne linije, koje su povezane na *pull-up* otpornik. Na izlazu senzora konstantno se dobija visoka vrednost, osim kada njegova površina dođe u kontakt sa tečnošću i tada se dobija signal na izlazu senzora. Postavljen je na gornjoj površini motocikla, i ukoliko kiša počne da pada, šalje signal na kontroler, koji preko *Bluetooth* modula, opisanog u poglavlju 2.5, komunicira sa *Android* uređajem. Nakon toga *Android* uređaj šalje SMS poruku vlasniku sa obaveštenjem da je počela kiša i da je potrebno skloniti motocikl.

PIR senzori [2] (engl. *Passive Infrared*) koncipirani su na piroelektričnom senzoru koji može da detektuje razna infracrvena zračenja. Na taj način može da detektuje čovekovo prisustvo i kao takav se koristi u bezbednosne svrhe u vidu zaštite od krađe. Malih je dimenzija pa ga je moguće postaviti bilo gde, a izgled je prikazan na slici 2.6.



Slika 2.6. PIR senzor

Uz pomoć dva ugrađena potencijometra moguće je podešavati njihovu osetljivost kao i distancu detekcije.

Ukoliko detektuje prisustvo čoveka, šalje signal na mikrokontroler. On dalje preko *Bluetooth* modula komunicira sa *Android* uređajem koji šalje SMS poruku kao obaveštenje da se u blizini nalazi potencijalni lopov.

Senzor vibracije, već opisan u poglavlju 2.3, ima za ulogu detekciju vibracije. Ukoliko neko pokuša da onemogući motocikl, nastale vibracije meri senzor SW-420, koji komunicira sa mikrokontrolerom.

Mikrokontroler, opet, vrši komunikaciju sa *Android* uređajem putem *Bluetooth* veze i na tako se vlasnik poziva na mobilni telefon. Na poziv je moguće odgovoriti javljanjem, i tako čuti šta se u tom momentu dešava oko motocikla. Za to vreme oglašava se ugrađena sirena kako bi se dalo do znanja da je motocikl osiguran alarmnim sistemom i na taj način sprečila potencijalna šteta.

2.4. Android aplikacija

Aplikacija je realizovana u softverskom paketu koji nosi naziv *Mitt App*. To je *Google*-ov alat, uz pomoć kog se *on-line* prave aplikacije za *Android* uređaje. Sastoji se iz dva dela – programerski i dizajnerski deo.

U programerskom delu realizuje se funkcionalni deo u vidu blokova, čijim se sklapanjem formira željeni kod, dok se u dizajnerskom delu pravi onaj deo aplikacije vidljiv krajnjem korisniku.

Izgled napravljene aplikacije je prikazan na slici 2.7.



Slika 2.7. Izgled Android aplikacije

Nakon povezivanja, odnosno uparivanja *Android* uređaja, na koji je instalirana prikazana aplikacija, sa mikrokontrolerom preko *Bluetooth* komunikacije, moguće je upravljati kompletnim uređajem klikom na vidljive dugmiće.

Na sredini ekrana nalazi se *Info ekran*, na kome se prikazuju sve bitne informacije i obaveštenja.

Ono što je integrisano u aplikaciju, a tiče se bezbednosti motocikla je to da slanjem SMS poruke na korišćeni *Android* uređaj, moguće je dobiti tačnu lokaciju motocikla u tom trenutku. Kada aplikacija registruje primljeni SMS, aktivira

GPS na uređaju i prikuplja informacije o lokaciji. Nakon toga talasnu dužinu i visinu generiše u *Google Maps* link i isti šalje putem SMS poruke na vlasnikom mobilni telefon.

2.5. Bluetooth modul

Kompletna komunikacija, odnosno upravljanje uređajem vrši se upravo preko *Bluetooth* [5] modula koji uspostavlja vezu između mikrokontrolera i *Android* uređaja. Korišćeni modul nosi oznaku HC-05. Prednost ovog modula u odnosu na druge je taj što se ponaša i kao *Slave* i kao *Master*. To znači da je moguća obostrana komunikacija između mikrokontrolera i *Android* uređaja. Korišćeni modul je prikazan na slici 2.7.



Slika 2.7. Bluetooth modul HC-05

Osim pomenute prednosti, odlikuju ga male dimenzije, naponski nivo od 5V koji mu omogućuje da se bez dodatnih elemenata napaja preko mikrokontrolera, kao i domet komunikacije od 10m.

2.6. RF komunikacija

RF [2] (engl. *Radio Frequency*) je bežična komunikacija koja podatke prenosi putem radio talasa čija je frekvencija u opsegu od 3kHz do 300GHz. Elementi prikazani na slici 2.8. korišćeni su za ovu vrstu komunikacije.

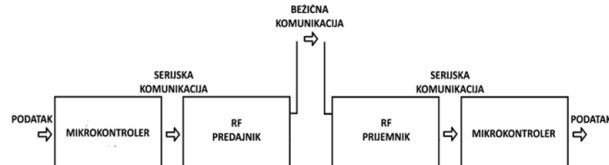


Slika 2.8. Moduli za RF komunikaciju

Sastoje se od dva dela- prijemnik i predajnik. Frekvencija rada između njih je 433Mhz. U fabričkom izdanju, ne poseduju svoju antenu, te im je domet ograničen na svega 2-3m. Dodavanjem eksterne antene, domet je moguće povećati i do nekoliko desetina puta.

Predajnik je postavljen na *Arduino Uno #1*, dok je prijemnik povezan sa *Arduino Uno #2*, koji se nalazi u kacigi vozača.

Blok dijagram prenosa podataka putem ove vrste komunikacije je prikazan na slici 2.9.



Slika 2.9. Blok dijagram RF komunikacije

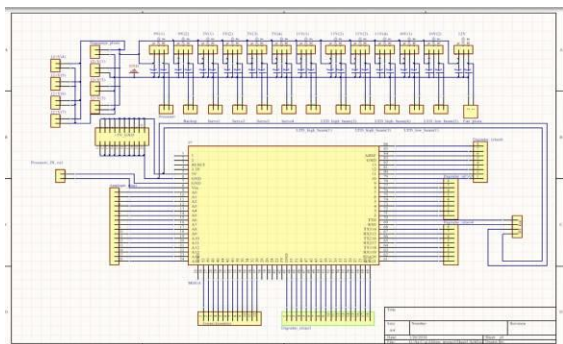
Kada Ultrasonični senzor, opisan u poglavlju 2.3, detektuje vozilo, šalje signal na *Arduino Mega* na čiji digitalni ulaz je povezan. Signal se putem serijske komunikacije prenosi na *Arduino Uno #1*, koji otpočinje komunikaciju sa *Arduino uno #2* putem RF veze.

Tada dolazi do aktiviranja svetlosne digitalizacije na kacigi vozača, koja svojim treperenjem skreće pažnju vozila koji se nalazi iza motocikla.

Isti princip prenosa signala se odnosi i za aktiviranje vazdušnih jastuka; kada senzor vibracija oseti udar, signal prolazi relaciju *Arduino Mega – Arduino Nano #1 – Arduino Nano #2*, koji aktivira vazdušne jastuke smeštene u jakni.

2.6. Konačna realizacija hardvera

Električna šema štampane ploče opisanog uređaja prikazana je na slici 3, dok je spoljašnji izgled uređaja prikazan na slici 3.1.



Slika 3. Šematski prikaz štampane ploče



Slika 3.1. Spoljašnji izgled kompletnog uređaja

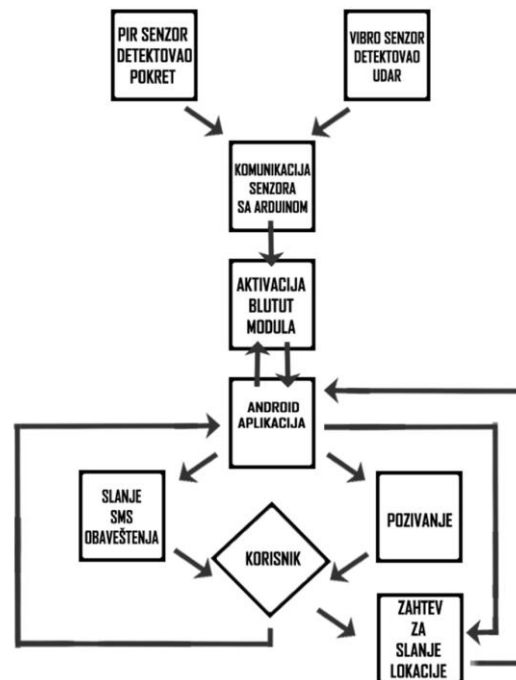
4. OPIS SOFTVERA

Algoritam rada softvera uređaja za automatizaciju motocikla realizovan je u vidu blokova prikazanih na slici 3.2.

Program se sastoji iz dva dela: *setup()* i *loop()*, odnosno deo koda koji se izvršava samo jednom i onaj koji se izvršava iznova i iznova.

Po uključenju uređaja, svi delovi uređaja dobijaju potreban radni napon napajanja i nalaze se u *standby* režimu. Svaki senzor opisan u poglavlju 2.3. ima svoj programski blok koji se izvršava tek nakon zadate komande, koje se manipulišu uz pomoć *Android* uređaja, prethodno konektovanog na mikrokontroler preko *Bluetooth* veze.

Nakon pokretanja, svaki programski blok je aktivan sve do njegovog ručnog isključivanja, osim bloka vezanog za alarm, odnosno dojavu putem poziva, opisanog u poglavlju 2.4. On se nakon svog izvršenja tj. uspostavljanja poziva, automatski isključuje. Tako se izbegava potencijalno, neželjeno dupliranje poziva, koje može dovesti do kolizije u mreži, što bi rezultovalo neuspešnoj realizaciji mobilne dojave.



Slika 3.2. Algoritam rada realizovanog softvera

5. ZAKLJUČAK

Uređaj za automatizaciju motocikla i upravljanje putem Bluetooth uređaja, čija je realizacija opisana u okviru ovog rada, u potpunosti zadovoljava polaznu koncepciju koja je podrazumevala kreiranje jednostavnog i jeftinog uređaja, edukativnog karaktera. Ovaj uređaj može međutim da se shvati i kao prototip, čijim bi daljim usavršavanjem eventualno mogao da se stvori i komercijalno zanimljiv proizvod.

Unapređivanje bi trebalo pre svega da ide u pravcu smanjenja dimenzija uređaja, kao i realizacija aplikacije i za ostale platforme, kao što su *Windows mobile*, *iOS*, *itd*.

6. LITERATURA

- [1] Arduino Tehnička dokumentacija i uputstvo za rad sa Mega2560 i Nano. Dostupno na www.arduino.cc, maj 2016
- [2] Tehnička dokumentacija i uputstvo za rad sa servo motorima. Dostupno na <https://en.wikipedia.org/wiki/Servomotor>, maj 2016
- [3] Tehnička dokumentacija i uputstvo za rad sa senzorom vibracije SW-420. Dostupno na www.instructables.com, maj 2016
- [4] Tehnička dokumentacija i uputstvo za rad sa senzorom daljine HCSR04. Dostupno na www.micropik.com/PDF/HCSR04.pdf, maj 2016
- [5] Vedran Kos "Princip komunikacije u Bluetooth bežičnoj mreži", maj 2006

Kratka biografija:



Petar Gostović rođen je 1987. godine u Novom Sadu. Master rad na Fakultetu tehničkih nauka u Novom Sadu iz oblasti Elektrotehnike i računarstva – Mikroročunarska elektronika odbranio je 2016. godine



Dr. Vladimir Rajs rođen je 1982. godine u Apatinu. Diplomirao je 2007, a doktorirao 2015. godine na Fakultetu tehničkih nauka u Novom Sadu. Od 2016. zaposlen je kao docent na Departmanu za elektroniku, energetiku i telekomunikaciju FTN-a. Oblasti interesovanja su mu elektronika i primenjena elektronika

**PROŠIRIVA PLATFORMA ZA GENERISANJE APLIKACIJA
IZ TEKSTUALNIH DSL MODELA****EXTENSIBLE PLATFORM FOR GENERATING APPLICATIONS
FROM TEXTUAL DSL MODELS**

Lazar Nikolić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu prezentovana je proširiva platforma za generisanje aplikacija iz tekstualnih DSL modela pod nazivom GenAn. Model generisane aplikacije se formira uz pomoć GenAn DSL (eng. Domain Specific Language) jezika implementiranog u sklopu platforme. Transformacija modela se postiže upotrebom generatora implementiranih kao priključci, čime se omogućuje proširivost platforme.

Abstract – This paper presents an extensible platform for generation of applications from textual DSL (Domain Specific Language) models named GenAn. Generated application's model is formed with GenAn DSL, implemented within the platform. Model transformation is achieved by using generators implemented as plugins, which enables extensibility of the platform.

Ključne reči: DSL, generisanje kode, komponentni softver, priključci, proširivost

1. UVOD

U ovom radu je prezentovana proširiva platforma za generisanje aplikacija iz tekstualnih DSL (eng. Domain Specific Language) modela pod nazivom GenAn. Nastala je u cilju poboljšanja procesa razvoja softvera, objedinjavanjem raznih oblasti softverskog inženjerstva. Sastoji se od četiri dela: GenAn DSL, GenAn Core, GenAn Plugins i GenAn Generators. GenAn platforma je složen sistem i zbog toga ovaj rad ukratko opisuje njene delove iz aspekta proširivosti platforme.

Moderni računarski sistemi imaju integralnu ulogu širom privrede. Preduzeća se nalaze u okruženju stalnih tržišnih turbulencija i potreban im je fleksibilan softver koji brzo može da se prilagođava promenama poslovnog ambijenta. Sa druge strane, softver zbog povećanja zahteva postaje sve složeniji što nepovoljno utiče na brzinu razvoja i cenu implementacije. Javlja se potreba za alatima koji omogućavaju ubrzanje razvoja softvera zajedno sa smanjivanjem njegove složenosti.

Glavni pristup rešavanju problema u softverskom inženjerstvu je povećavanje nivoa apstrakcije. Povećavanjem nivoa apstrakcije se gube specifičnosti rešenja, što omogućava razvojnom timu da rešavanje problema odvoji od implementacione tehnologije i fokusira se na suštinu problema. Grupe kao što su OMG (eng. Object Modeling

Group) su pokrenule MDA inicijativu (eng. Model Driven Architecture), kao odgovor potrebi za standardima i alatima za povećavanje nivoa apstrakcije. Kao glavni princip se nalaže da model, umesto pukog opisa sistema, treba da bude i deo implementacije. U tu svrhu je razvijen UML (eng. Unified Modeling Language) kao jezik za kreiranje modela visokog nivoa apstrakcije. Modeli kreirani na ovaj način mogu poslužiti kao osnova za generisanje aplikacija. MDA kroz UML nudi tek jedan način opisivanja modela visokog nivoa apstrakcije. Domen specifični jezici (eng. Domain specific Language – DSL) predstavljaju još jedan način za opis ovakvih modela.

Inženjerstvo upravljano modelima (Model Driven Engineering – MDE) zajedno sa DSL-om, koji je specijalizovan za opis modela u nekom domenu, može da ubrza razvoj i do deset puta u odnosu na tradicionalan pristup ručnog kodiranja[1].

Softversko rešenje se može posmatrati kao sistem načinjen od više celina sa jasno definisanom ulogom, koje međusobno komuniciraju u cilju rešavanja problema. Ove celine se nazivaju softverske komponente. Komponovanjem i orkestracijom međusobne komunikacije komponentata se dobija funkcionalan softver. Iz ove ideje je nastao Razvoj softvera baziran na komponentama (eng. Component Based Software Engineering – CBSE) kao oblast softverskog inženjerstva. Softver izgrađen prema ovim principima je fleksibilan i kvalitetan, jer se komponente nezavisno razvijaju od strane timova stručnjaka specijalizovanih u određenom domenu. Odabirom kombinacije komponentata se dobija softver sa željenim karakteristikama.

1.1. Motivacija

Gore opisani pristupi su se u praksi pokazali kao veoma povoljan uticaj na proces razvoja softvera. U cilju još daljeg poboljšavanja tog procesa, donešena je odluka da se GenAn platforma implementira objedinjavanjem dobre prakse iz posmatranih oblasti.

Odluka da se za specifikaciju modela aplikacija kreira novi tekstualni DSL je donešena na osnovu tri opaske. Prva opaska je da se podaci jednog tipa na isti način prikazuju širom aplikacije, u cilju održavanja konzistentnosti korisničkog interfejsa. Druga opaska je da prezentacija podataka strogo zavisi od njegovog tipa. Treća opaska je nezgrapnost alata za povezivanje podataka sa elementima interfejsa koji ih prikazuju. Iz ove tri opaske se takođe može zaključiti da se aplikacija može opisati kroz njen korisnički interfejs. Kao rezultat, GenAn koristi specifikaciju korisničkog interfejsa kao

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Igor Dejanović.

model na osnovu kojeg se generiše čitava web aplikacija u dva dela: *frontend* i *backend*.

2. TRENUTNO STANJE U OBLASTI

Za opis modela na osnovu kojeg se generiše aplikacija postoji više pristupa, od kojih su dva razmatrana u ovom radu. Prvi, tradicionalniji način je upotrebom UML-a, dok je drugi način kreiranjem i upotrebom namenskog DSL-a.

2.1. Sifu-DSL

Sifu-DSL je DSL niskog nivoa apstrakcije koji razvija firma dryTools. Namenjen kreiranju softverskih specifikacija na osnovu kojih softverski alat Sifu generiše web aplikaciju. Klijentska i serverska strana rezultujuće web aplikacije se posebno specificiraju u jeziku. Kreiranje novih tipova u jeziku se vrši u tri dela: *element settings*, *list of elements* i *list of behaviours*.

```
# define a model
Customer model(
  # define model properties and fine tune every property according to your needs
  email text[min 1, max 100, indexed]
  details PersonalDetails
  active boolean
  numberOfVisits integer
  # you can reference another model too
  company Company
  unique(company, email)
)
```

Slika 1. Primer modela u Sifu-DSL jeziku

Obavezan korenski tip svake specifikacije je *project*. Sadrži jedan ili više serverskih i klijentskih elemenata, koji opet mogu sadržati elemente drugog tipa. Stablo se gradi sve dok svi elementi nisu definisani primitivnim tipovima.

Server element omogućava definisanje servera ili servisa aplikacije. Ima dva obavezna podešavanja: *template* i *developer*. *Template* određuje tip backend aplikacije, tj. koji će se generator pozvati za ovaj element (Scala ili Java Spring).

Client element opisuje klijentsku stranu koja može biti android ili web aplikacija. Klijent se gradi upotrebom elemenata tipa *page* koji predstavljaju stranicu ili prozor klijentske aplikacije. Daljom kompozicijom klijentskih tipova unutar elementa ovog tipa se definiše izgled aplikacije.

Integrisanje ručno pisanog koda sa generisanim se vrši uz pomoć git-a. Sifu alat komandom `sifu init projectName` za svaki projekat kreira git repozitorijum sa dve grane: *develop* za ručno pisani kod i *sifu* za generisani kod.

2.2. WebDSL

WebDSL je istraživački projekat nastao 2007. na Tehničkom univerzitetu u Delftu (Delft University of Technology). U svom jezgru sadrži jezik za implementaciju dinamičkih web aplikacija sa bogatim modelom podataka [2] sa podjezicima za definiciju entiteta, stranica i poslovne logike. Apstrakcije visokog nivoa, modelovanje kontrole pristupa i radnog toka su definisani modularno kao ekstenzije jezgra [3]. U ovom radu su prikazane samo osnovne funkcionalnosti. Modeli se opisuju definicijom entiteta.

Glavni nedostatak WebDSL-a jeste to što podržava samo jedan stek tehnologija i ne postoje nikakve naznake da će se ovo promeniti. Da bi stvari bile gore, podržane tehnologije su u vreme pisanja ovog rada već bile zastarele.

```
application minimalac

entity User {
  name :: String
  password :: Secret
}

init{
  var u := User{ name := "1" password := ("1" as Secret).digest() };
  u.save();
}

define page root(){
  authentication()
  " "
  navigate protectedPage() { "go" }
}

define page protectedPage(){ "access granted" }

principal is User with credentials name, password

access control rules

rule page root(){true}
rule page protectedPage(){loggedIn()}
```

Slika 2. Primer WebDSL aplikacije

2.3 AndroMDA

AndroMDA je alat koji omogućava definisanje modela sistema zavisnog od platforme (eng. Platform Specific Model – PSM) uz oslonac na UML proširen odgovarajućim profilom. Stereotipi pružaju dodatne informacije potrebne za generisanje koda. Kao ulazni model se koristi model zapisan u standardnom XMI formatu, ali i pored toga razvojni tim preporučuje korišćenje MagicDraw alata.

Za generisanje koda se koriste tzv. kertridži. Kertridži su softverski moduli koji se mogu instalirati uvođenjem u classpath projekta. Jedan kertridž odgovara jednoj ciljnoj platformi. Generisanje se obavlja upotrebom šablona (eng. template) i obrađivača šablona (eng. template engine). AndroMDA dolazi sa velikim brojem kertridža za razne platforme, .NET, Java Hibernate i EJB. Proširivanje generisanog koda se vrši nasleđivanjem, dok se proširivanje ostalih datoteka (npr. XML konfiguracionih fajlova) vrši na unapred predodređenim mestima koja se zovu tag-ovi.

Proširivanje meta-modela je omogućeno kroz meta-fasade. Implementacija ktertadža podrazumeva kreiranje nove meta-vasade, tj. klasa koje su fasade za prave meta-klase. Te klase vraćaju string-ove koji pružaju informacije o meta-objektima modela. Mapiranje metoda iz šablona na metode iz meta-klase se obavlja u konfiguracionim fajlovima.

3. IMPLEMENTACIJA

GenAn platforma je implementirana u programskom jeziku Python i sastoji se od četiri dela: GenAn DSL, GenAn Core, GenAn Plugins i GenAn Generators.

3.1 GenAn DSL

GenAn DSL je eksterni, tekstualni DSL za izgradnju aplikacija opisivanjem njihovog korisničkog interfejsa.

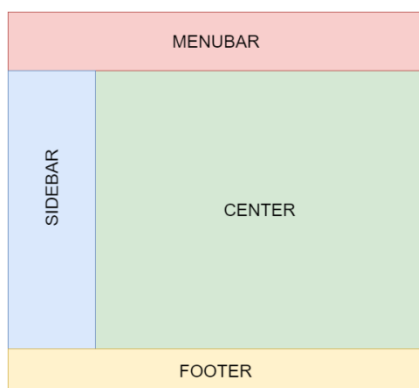
Kreiran je upotrebom textX [5] alata za opis meta-modela jezika baziranog na Arpeggio pacrat parseru [6]. Specifikacija korisničkog interfejsa se vrši upotrebom tri ključna koncepta: *object*, *view* i *page*.

Object (objekat) enkapsulira podatke o načinu prikaza entita iz modela podataka kroz njegove attribute. Mapira se na entitete perzistentnog modela podataka. U terminologiji relacionih baza podataka, svaki objekat se mapira na jednu tabelu. Svakom tipu atributa objekta odgovara jedan pogled koji sadrži način njegove prezentacije. Objekti se vezuju za poglede ili stranice kroz koje se prikazuju upotrebom upita. Na osnovu tog upita pogled, odnosno stranica dobavlja konkretni objekat i ubacuje ga u svoj kontekst. Relacije između objekata se definišu posebnim pravilom *Meta*.

```
object Film:
  Meta:
    list Actor by first_name, last_name 'Actors'
    single ProductionHouse by name 'ProductionHouse'
  ;
  title text label='Title'
  description text label='Description'
  imdb_rating number label='Rating'
;
```

Slika 3. Primer objekta u GenAn DSL-u

Page (stranica) pravilo omogućava deklarativan opis izgleda stranice ili prozora aplikacije. Nastaje kompozicijom jednog ili više pogleda.



Slika 4. Prostorni raspored stranice

Prostorni raspored stranice se sastoji od četiri unapred određenih zona u koje se ubacuju pogled (slika 4):

- **sidebar** sa leve strane prostora, namenjen da sadrži opcije menija. Može sadržati bilo koji pogled, ali je preporučeno koristiti Menubar.
- **footer** u donjem delu prozora. Preporučuje se pogled tipa Footer.
- **menubar** u gornjem delu prozora. Za formiranje menija se koristi pogled tipa Menubar koji sadrži stavke menije kao poglede tipa Menu.
- **center** u centralnom delu prozora. Služi za smeštanje glavnog sadržaja prozora i jedini je obavezan od navedenih zona. Za razliku od ostalih, centar nema specifično namenjeni pogled. Obično će stranica imati jedan pogled u ovom glavnom sadržaju, a dalja kompozicija se vrši ugnježdavanjem pod-pogleda (subviews).

Ako se neka od zona ne navede, centar će se proširiti na njen račun. Dakle, stranicu je moguće strukturirati upotrebom bilo koje kombinacije zona, dokle god je centar definisan.

```
page films_page:
  title = 'Films'
  use menubar mainMenu
  use sidebar mainSidebar
  use footer mainFooter
  show films_view position=center
;
```

Slika 5. Primer objekta u GenAn DSL-u

View (pogled) definiše jedan pogled unutar aplikacije i predstavlja osnovnu jedinicu kompozicije korisničkog interfejsa. GenAn DSL sadrži osnovne poglede za predstavu primitivnih tipova atributa objekata. Složeni pogledi se mogu graditi kompozicijom osnovnih ili drugih složenih pogleda. Pogled se ne može prikazati nezavisno i mora biti deo neke stranice. Slično kao i kod stranice, konkretni objekti se mogu vezati za neki pogled kroz upit u atributu query. U svrhu kompozicije pogledi se dele u dvanaest kolona i neograničen broj redova.

```
view actor_single_view:
  !row 4
  show Actor.first_name at 2 size 8
  !row 5
  show Actor.last_name at 2 size 8
  !row 6
;
```

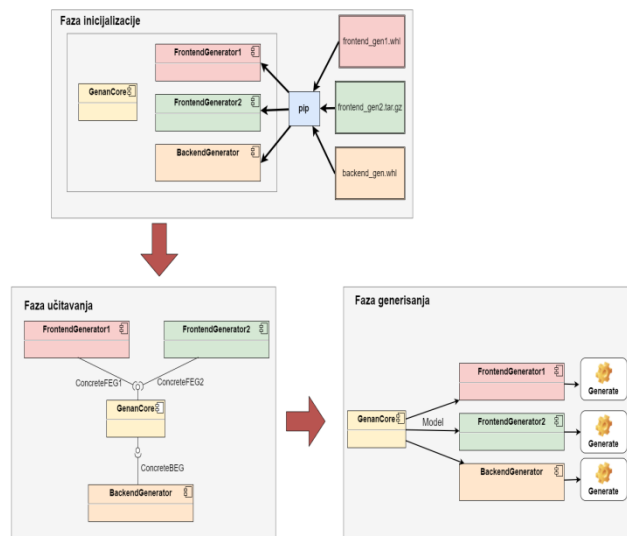
Slika 7. Primer pogleda u GenAn DSL-u

3. 2 GenAn core

Jezgro GenAn platforme sadrži meta-model u obliku Python klasa, mehanizme za učitavanje meta-modela i modela i osnovne servise za formiranje proširive platforme. Klase potrebne za implementaciju generatora su takođe definisane definisane ovde: FrontendGenerator za frontend generatore i BackendGenerator za backend generatore. Ove klase su implementirane prema Visitor šablonu [4].

Izvršavanje GenAn platforme se odvija u tri faze. U prvoj fazi se inicijalizuje platforma tako što se kreira virtuelno okruženje u čijem se kontekstu izvršava. Zatim otkrivaju i učitavaju priključci sa generatorima. U drugoj fazi se učitavaju i instanciraju instalirani paketi. Paketi sadrže generatore koje pružaju preko mehanizma ulaznih tačaka. Jezgro platforme propisuje dve vrste ulaznih tačaka: **genan.backend_generator** za backend generatore i **genan.frontend_generator** za frontend generatore. Da bi paketi mogli da ih koriste i na taj način budu deo platforme, potrebno je da sadrže klase koje nasleđuju klasu BackendGenerator (genan.backend_generator) ili klasu FrontendGenerator (genan.frontend_generator). U trećoj fazi se instanciraju generatori na osnovu dobavljenih klasa. Na početku ove faze se od korisnika preko komandne linije dobija putanja do DSL skripte sa specifikacijom aplikacije. Generatorima se zatim prosleđuje model učitao ove skripte, i utvrđuju konekcioni

parametri, poput adrese servisa backend-a. Na kraju se pokreće proces generisanja za svaki generator.



Slika 8. Faze izvršavanja GenAn platforme

3. 3 GenAn plugins

Za modularizaciju platforme je potrebno da generatori budu implementirani kao priključci. U tu svrhu se razvijaju i distribuiraju kao Python paketi, tako da će se u daljem tekstu reći komponenta i paket tretirati kao sinonimi. Svaki Python paket sadrži deskriptor kreiran korišćenjem Setuptools alata. Deskriptori se sastoje od meta-podataka o paketu i služe kao njegov komponentni ugovor. Bitniji meta-podaci za ovaj rad su:

- **name** definiše ime komponente, tj. njen identifikator.
- **version** pruža informaciju o verziji komponente. Zavisne komponente mogu specificirati verziju komponente od koje zavise.
- **requires** definiše koji paketi moraju biti dostupni u sistemu, tj. definiše eksplicitne zavisnosti komponente.
- **provides** definiše module koje izvozi paket, tj. definiše spisak servisa koje pruža komponenta.
- **entry_points** definiše operacije komponente, tj. njen interfejs. Pošto je u Python programskom jeziku sve objekat, operacije mogu biti funkcije, klase, objekti ili globalne konstante. GenAn platforma nudi nekoliko priključnih tačaka (eng. extension point) koje se identifikuju imenom grupe ulaznih tačaka. Komponente se na njih priključuju implementacijom ulazne tačke odgovarajuće grupe.

Kao što je navedeno u poglavlju 3.2, postoje dve vrste generatora za koje postoje dve grupe ulaznih tačaka. Servisi generatorskih paketa moraju biti klase koje nasleđuju odgovarajuću bazičnu generatorsku klasu, propisane od strane grupe njihove ulazne tačke. Klase i njihove pristupne tačke zajedno formiraju komponentni model platforme. Detalji implementacije, kao što su način generisanja određenih čvorova i integracija ručno pisanog koda, su ostavljeni na razvojnom timu generatora. Jedino što je neophodno je da priključci, tj. paketi koji implementiraju generatore prate komponentni model.

Jedan paket može otkrivati veći broj različitih vrsta generatora.

```
setup(
    name='genan-angular',
    version='0.1.0',
    description='AngularJS generator based on GenAn DSL',
    py_modules=['angular_generator', 'jinja_filters'],
    data_files=data_files,
    install_requires=[
        'textx',
        'arpeggio',
        'jinja2',
        'click'
    ],
    entry_points={
        'genan.frontend_generator': [
            'generator=angular_generator:AngularGenerator',
        ],
    },
)
```

Slika 9. Primer deskriptora paketa

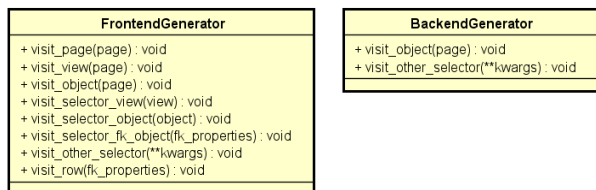
3. 4 GenAn generators

Opis GenAn generatora se prožima kroz prethodna poglavlja, ali se u ovom poglavlju sve informacije o njima objedinjuju zbog preglednosti. Generatori moraju da poštuju sledeća pravila:

- Generatori su Python paketi dobijeni pomoću Setuptools alata. Kao takvi se instaliraju u sistem na dva načina: direktnom instalacijom u virtuelno okruženje korišćenjem pip alata, ili dodavanjem distribucije u binarnom formatu (*wheel*, *egg* ili arhivirano u *tar.gz*) u *plugins* direktorijum.
- Paketi implementiraju i otkrivaju najmanje jednu generatorsku klasu. Generatorske klase nasleđuju jednu od dve bazične klase: BackendGenerator ili FrontendGenerator. Implementirane su po Visitor šablonu, tako da poseduju metodu za svaku klasu objekta modela. Te klase se zatim pružaju na raspolaganje ostatku sistema kao servisi kroz odgovarajuću grupu ulaznih tačaka. Jedan generator može imati obe vrste generatora, i za svaku vrstu može imati više konkretnih generatora.
- Ne postoje propisi za način integracije ručno pisanog koda i postupak transformacije objekata modela u programski kod. Odluke vezane za njihovu realizaciju su ostavljene razvojnom timu generatora. Jedino treba voditi računa o tome da backend i frontend aplikacije moraju biti usklađeni: frontend mora imati informacije o adresi backend-a aplikacije i o načinu upotrebe njegovog API-ja.

Generatorske klase su implementirane po Visitor šablonu i imaju metodu za svaku klasu modela. Platforma pre generisanja snabdeva generatore modelom dobijenim iz DSL skripte upotrebom textX alata. Model je graf Python objekata prilikom čijeg se obilaska pozivaju metode generatora koje odgovaraju klasi trenutno posmatranog

objekta. Konvencija za imenovanje visitor metoda je `visit_ime_klase(param1, param2...)`.



Slika 10. Bazične generatorske klase

4. ZAKLJUČAK

U ovom radu je prezentovana GenAn platforma za generisanje aplikacija iz tekstualnih DSL modela. Platforma je nastala u cilju objedinjavanja principa CBSE MDA i DSL zarad unapređivanja procesa razvoja softvera. Korišćenjem DSL-a postignuta je mogućnost lakog specifikiranja i izmene modela nezavisnog od platforme, na osnovu kojeg se generiše aplikacija. DSL skripta koja specifikira aplikaciju sadrži mali broj linija koda, čime se olakšava održavanje i smanjuje ukupan broj grešaka.

Podrška za različite ciljne platforme postiže se kreiranjem i instalacijom priključaka koji sadrže generatore koda za željenu platformu.

Kao rezultat dobijena je fleksibilna platforma sa mogućnošću pokrivanja potencijalno velikog broja ciljnih platformi.

5. LITERATURA

[1] D. C. Schmidt, "ModelDriven Engineering", Computer, IEEE Computer Society, Februar 2006, pp. 2531.

[2] E. Visser. WebDSL: A Case Study in Domain-Specific Language Engineering. In R. Lammel, J. Saraiva, and J. Visser, editors, Generative and Transformational Techniques in Software Engineering (GTTSE 2007), Lecture Notes in Computer Science. Springer, 2008. Tutorial for International Summer School GTTSE 2007

[3] D. M. Groenewegen, Z. Hemel, L. C. L. Kats, and E. Visser. WebDSL: A domain-specific language for dynamic web applications. In N. Mielke and O. Zimmermann, editors, Companion to the 23rd ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2008), pages 779–780, New York, NY, USA, October 2008. ACM. (poster).

[4] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Grady Booch, Design Patterns: Elements of Reusable Object-Oriented Software, ISBN-10: 0201633612, ISBN-13: 978-0201633610, Addison-Wesley Professional; 1 edition (November 10, 1994)

[5] Web stranica <http://igordejanovic.net/textX/>, (pristupano 10.10.2016.)

[6] Dejanović I., Milosavljević G., Vaderna R.: Arpeggio: A flexible PEG parser for Python, Knowledge-Based Systems, 2016, 95, 71 - 74, doi:10.1016/j.knosys.2015.12.004

Kratka biografija:

Lazar Nikolić rođen je u Somboru 1992. godine. Master rad na Fakultetu tehničkih nauka, u Novom Sadu iz oblasti Elektrotehnike i računarstva – softversko inženjerstvo, odbranio je 2016. godine

CUDA БАЗИРАНА ПАРАЛЕЛИЗАЦИЈА МНОЖЕЊА СИМЕТРИЧНИХ МАТРИЦА И ВЕКТОРА

MULTIPLICATION OF SYMMETRIC MATRICES AND VECTORS BASED ON CUDA PARALLELIZATION

Горана Гојић, Факултет техничких наука, Нови Сад

ОБЛАСТ– ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – У овом раду су приказане две имплементације SYMV алгорита за множење симетричне матрице и вектора, имплементирани за извршавање на CUDA (енг. Compute Unified Device Architecture) графичким процесорима.

Abstract – In this paper we present two implementations of SYMV algorithm for symmetric matrix-vector multiplication, implemented for execution on CUDA graphics processors.

Кључне речи: CUDA, C, SYMV, GEMV, симетрична матрица, вектор, GPU

1. УВОД

Множење симетричне матрице и вектора [1] (SYMV) је једна од основних математичких операција из области линеарне алгебре. Своју примену има и у рачунарству у области нумеричких алгоритама. Како величине матрице и вектора могу веома да варирају, нумеричко решавање проблема који укључују SYMV може бити веома дуготрајно и због тога непогодно за извршавање на централним процесорима (енг. *central processing unit - CPU*). Алгоритам множења матрице и вектора се може разложити на међусобно независне кораке, што га чини добрим кандидатом за паралелизацију на графичким процесорима. Међутим, изазов у имплементацији SYMV алгоритама за извршавање на графичким процесорима лежи у чињеници да приступ елементима матрице не подлеже регуларним шаблонима приступа, што смањује перформансе алгорита.

Овај рад је настао по узору на рад под називом *Optimizing Symmetric Dense Matrix-Vector Multiplication on GPUs* [2]. Слично као и у претходно наведеном раду и овде су представљена два приступа решању проблема множења симетричне матрице и вектора. Имплементирани су GEMV (енг. *General Matrix Vector Product*) алгоритам за множење несиметричне матрице и вектора и SYMV алгоритам за множење симетричне матрице и вектора уз коришћење CUDA екстензије C програмског језика. Перформансе оба типа алгорита су мерене за извршавање над симетричним матрицама и векторима и поређене са референтним паралелним и секвенцијалним извршавањем.

НАПОМЕНА:

Овај рад проистекао је из мастер рада чији ментор је био др Жарко Живанов, доцент.

Паралелни алгоритми су тестирани на NVIDIA GeForce GTX970 [3] графичкој карти, а референтни секвенцијални на Intel Xeon E5-1620 v3 процесору [4].

2. КОРИШЋЕНА ТЕРМИНОЛОГИЈА

За несметано праћење излагања у овом раду је потребно упознати се са коришћеном терминологијом. У наставку ће се блок нити означавати са TB . Величина рачунског блока са N_{tb} , а величина блока нити са N_b , односно $N_x \times N_y$ за дводимензионалне блокове нити. Са $tile_size$ је обележен секундарни ниво блокирања унутар блока нити.

3. GEMV

GEMV [5] дефинише поступак множења матрице која не мора бити симетрична и вектора. Спада у групу основних потпрограма другог нивоа у библиотекама које се баве линеарном алгебром. Формалан приказ рутине је дат изразима (1) и (2).

$$y := \alpha Ax + \beta y \quad (1)$$

$$y := \alpha A^T x + \beta y \quad (2)$$

Где су α и β скалари, x и y вектори, A матрица димензија $m \times n$, а A^T транспонована матрица димензија $n \times m$. При имплементацији алгоритама је узет у обзир случај где важи $\alpha=1$ и $\beta=0$. Подразумевана се да је матрица A у меморији линеаризована врсту по врсту (енг. *row-major*).

3.1 GEMV нетранспоновани алгоритам

Имплементирани су три верзије кернела за GEMV нетранспоноване матрице које се разликују по типу коришћене меморије и шаблонима приступа елементима матрице и вектора.

У првој верзији свака нит чита један по један елемент врсте матрице и одговарајући елемент вектора, рачуна елемент резултујућег вектора и смешта га у глобалну меморију.

Приступ елементима матрице није узастопан, а елементи вектора се читавају више пута.

Друга верзија алгорита рационалније рукује вектором, па свака нит у блоку читава један елемент вектора у дељену меморију, а све нити га користе.

Трећа верзија користи дељену меморију за остваривање узастопног приступа на нивоу једне врсте матрице. За разлику од прве верзије где свака нит читава један по један елемент исте врсте матрице, у овој верзији свака нит читава по један елемент једне врсте матрице одређен индексом те нити.

3.2 GEMV транспоновани алгоритам

GEMV транспоновани алгоритам ради над матрицама линеаризованим колону по колону (енг. *column-major*). Направљене су укупно две имплементације овог алгоритма.

Прва верзија функционише аналогно првој верзији GEMV нетранспонованог алгоритма. Међутим, у овом случају је приступ глобалној меморији узастопан због начина линеаризовања матрице. Ипак, овај алгоритам учитава елементе вектора више пута него што је то потребно.

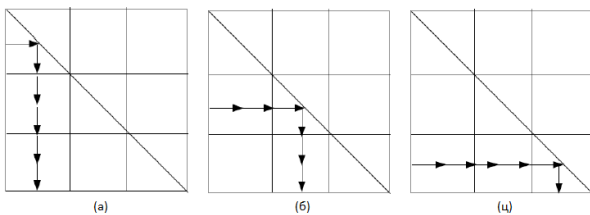
Друга верзија овог алгоритма, као и у случају GEMV нетранспонованог алгоритма учитава елементе вектора у дељену меморију, где се они више пута користе од стране више нити. Трећа верзија није имплементирана, јер је приступ матрици већ узастопан и није потребно оптимизовати га.

4. SYMV

SYMV [5] поступак је дефинисан изразом (1) из GEMV поглавља, уз разлику да је матрица A димензија $n \times n$ и симетрична у односу на главну дијагоналу. И у овом случају ће се сматрати да важи $\alpha=1$ и $\beta=0$.

4.1 SYMV Storage Oblivious

Као и код GEMV кернела, рачунање је организовано као једнодимензионална мрежа рачунских блокова величине N_{tb} . Свака рачунска мрежа је додељена једном блоку нити. За овај алгоритам (у наставку *алгоритам 1*) ће се сматрати да важи $tile_size=N_{tb}$. На овај начин је матрица A издељена на плочице (енг. *tile*) величине $n \div tile_size$. Сваки блок нити приступа плочицама $\{A_{i,j}; 1 \leq j \leq i\} \cup \{A_{i,j}; i \leq j \leq |TB|\}$. Ова верзија алгоритма је осмишљена тако да се искористе претходно написани GEMV кернели. Стога се над сваком плочицом матрице и њој одговарајућом плочицом вектора, позива транспоновани, односно нетранспоновани GEMV или SYMV рекурзивно.



Слика 1. Три типа рачуна блокова нити: (а) Тип 1 (б) Тип 2 (ц) Тип 3

На слици 1 су приказана три карактеристична случаја који се појављују у овом алгоритму. То су:

- *Тип 1* – Блок нити примењује SYMV над првом, па транспоновани GEMV над преосталим плочицама. Примењује се у случају када се све плочице налазе у истој колони плочица, као што је приказано под (а) на слици 1.
- *Тип 2* – Блок нити примењује нетранспоновани GEMV над свим плочицама у истој врсти, осим над плочицом на главној дијагонали матрице (дијагоналне плочице) над којима се примењује SYMV. Над

преосталим плочицама се примењује GEMV транспоновани алгоритам Овај случај је приказан под (б) на слици 1.

- *Тип 3* – Блок нити примењује нетранспоновани GEMV над свим плочицама осим над последњом, над којом се примењује SYMV. Овај случај је приказан под (ц) на слици 1.

SYMV алгоритам наведен у претходна три карактеристична случаја представља GEMV алгоритам који се примењује над плочицом добијеном пресликавањем половине матрице учитане из глобалне меморије. Сваки блок нити обрађује више плочица. Блок нити има $N_t = N_{tx} \times N_{ty}$ које из глобалне меморије паралелно учитавају потплочицу величине $N_{ty} \times tile_size$, а обрађују потплочицу величине $tile_size \times N_{ty}$. Резултујући вектор се добија сумирањем свих N_{ty} парцијалних резултата.

4.2 SYMV Storage Aware

Ова верзија алгоритма (у наставку *алгоритам 2*) је настала као унапређење *алгоритма 1*. Састоји се из два кернела – *кернала 1* за рачунање парцијалних резултата множења матрице и вектора и *кернала 2* за сумирање парцијалних резултата и добијање резултујућег вектора.

Рачунање у *кERNELU 1* је организовано као једнодимензионална мрежа рачунских блокова величине N_{tb} . Блокови нити су дводимензионални као и у случају *алгоритма 1*. Подразумева се да важи $tile_size=N_{tb}$, што доводи до логичке подељености матрице на $|TB| \times |TB|$ плочица величине $n \div tile_size$. Сваки блок нити треба да обради скуп плочица матрице. У наставку ће блок нити бити означен са TB_i , где је i редни број блока. Сваки блок нити обрађује плочице одређене изразом $\{A_{i,j}; 1 \leq j \leq i\}$. Са $A_{i,j}$ су обележене плочице матрице, са x_i плочице вектора којим се множи, а са y_i плочице резултујућег вектора. Сваки TB_i рачуна по један парцијални резултат за сваку плочицу коју обради. Парцијални резултати за TB_i се означавају са $\{y_i^j; 1 \leq j \leq i\}$ и рачунају се према према изразу (4) за $i \neq j$ и изразу (5) $i = j$.

$$y_i^j := A_{i,j}^T x_i \quad (3)$$

$$y_i^i := \sum_{j=1}^{j=i} A_{i,j} x_j \quad (4)$$

Као и *алгоритам 1*, *алгоритам 2* ради пресликавање учитаних елемената дијагоналних плочица матрице, како би смањено број приступа глобалној меморији. По завршетку *кернала 1*, покреће се *кERNEL 2* који треба да сумира парцијалне резултате према изразу (5).

$$y_i := \sum_{j=1}^{j=|TB|} y_i^j \quad (5)$$

Имплементирани су две подверзије овог алгоритма које се разликују по начину имплементације *кернала 1*. Прва имплементација *кернала 1* користи $tile_size$ елемената у дељеној меморији за чување плочице вектора и $tile_size$ елемената за чување парцијалних резултата рачуна. Друга верзија *кернала 1* користи

дупло више дељене меморије за чување плочице вектора и парцијалних резултата, али и постиже боље резултате при извршавању.

4.3 Оптимизације алгоритма 2 за извршавање на матрицама нерегуларних величина

Већина BLAS (енг. *Basic Linear Algebra Subprograms*) кернела даје добре перформансе када су димензије матрице умножак димензија плочице, односно, када су матрице регуларне величине. У случају неиспуњености овог услова перформансе алгоритма опадају. У наставку ће бити представљене три технике за оптимизацију у случају извршавања над матрицама нерегуларних величина.

- Условно извршавање (енг. *conditional statement*) – У код кернела је убачено више гранања да би се из даљег рачунања искључиле нити које излазе изван граница матрице. Ова техника продужава време извршавања кернела због потребе вишеструког извршавања основа (енг. *warp*), где основа представља најмању јединицу извршавања нити.
- Допуњавање матрице (енг. *padding*) - Врсте и колоне матрице нерегуларних величина се допуњавају нулама до прве следеће веће димензије која представља умножак димензија плочице. Ова техника захтева додатно заузеће простора за допуну матрице и додатни утрошак времена на пребацивање више података из радне меморије рачунара у глобалну меморију графичке карте.
- Преусмеравање показивача (енг. *pointer redirecting*) – У овој техници све нити врше рачунање, да би се на крају отказало чување резултата нити које излазе изван граница

матрице. Да би се спречио приступ недозвољеним локацијама, нити у ивичним плочицама се преусмеравају на приступ последњој валидној врсти тј. колони матрице. Преусмеравање се врши нумеричком корекцијом индекса уместо условним гранањима, да би се избегло вишеструко извршавање истих основа.

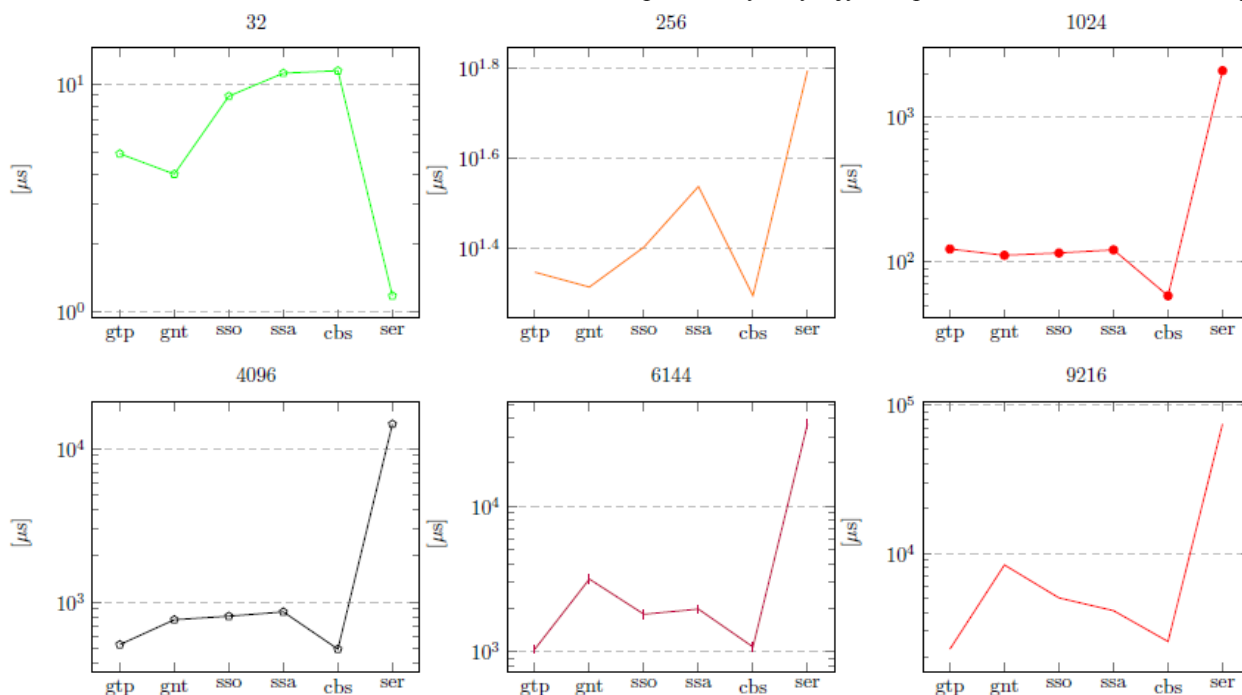
5. АНАЛИЗА ВРЕМЕНА ИЗВРШАВАЊА

На слици 2 су приказана времена извршавања претходно описаних паралелних алгоритама, референтног секвенцијалног и cuBLAS SYMV [6] алгорита. На у оси сваког од шест приказаних графика је приказано логаритамско време извршавања алгорита у микросекундама. На х оси је постављена ознака алгорита за који се мери време извршавања. Називи који одговарају алгоритмима су приказани у табели 1.

Ознака	Алгоритам
<i>gtp</i>	GEMV транспоновани
<i>gnt</i>	GEMV нетранспоновани
<i>sso</i>	SYMV Storage Oblivious
<i>ssa</i>	SYMV Storage Aware
<i>cbs</i>	cuBLAS SYMV
<i>ser</i>	Секвенцијална имплементација за CPU

Табела 1. Скраћенице и алгоритми

За приказ измерених перформанси је одабрано шест матрица регуларних величина. За сваки извршавани алгоритам је изабрана и извршавана његова најефикаснија верзија. Генерално, време извршавања свих алгорита се повећава са порастом димензија матрице и вектора. Анализом графика са слике 2 се може закључити да паралелизација није добро решење у случају матрица веома малих димензија.



Слика 2. Времена извршавања алгоритама на узорцима матрица за једноструку прецизност

Међутим, са порастом димензија време извршавања секвенцијалног алгоритма множења матрице и вектора расте и веома брзо надмашује све остале паралелне алгоритме. Од паралелних алгоритама, на свих шест графика најбоље перформансе показују NVIDIA cuBLAS SYMV и GEMV транспоновани алгоритам, због регуларног шаблона приступа дељеној меморији. Перформансе GEMV нетранспонованог алгоритма опадају са порастом матрице, због израженијег нерегуларног шаблона приступа. За веће величине матрица, оба имплементирана SYMV алгоритма постају временски ефикаснија од GEMV нетранспонованог алгоритма, јер су ови алгоритми извршавани са дводимензионалним конфигурацијама блокова нити, што је довело до паралелне обраде елемената унутар плочице матрице. Од два SYMV алгоритма се за мање матрице ефикаснијим показао SSO алгоритам, док је за веће матрице ефикаснији SSA алгоритам. Разлог за ово би могла бити чињеница да SSO при извршавању користи GEMV транспоновани и нетранспоновани кернел. За матрице чије су димензије веће од 6144, време за које се изврши GEMV нетранспоновани постаје знатно веће од времена за које се изврши GEMV транспоновани алгоритам, што успорава време извршавања SSO алгоритма.

6. ЗАКЉУЧАК

На основу претходно изложеног закључено је да је од свих имплементираних алгоритама који се могу користити за множење симетричне матрице и вектора најефикаснији GEMV транспоновани алгоритам, који је по перформансама сличан cuBLAS SYMV алгоритму. Иако најефикаснији, GEMV транспоновани алгоритам не може да ради директно над матрицама линеаризованим врсту по врсту, због чега је имплементирана верзија GEMV за нетранспоноване матрице.

Ова верзија је показала лошије перформансе, због недостатка узастопног приступа глобалној меморији графичког процесора. Обе имплементирани верзије SYMV алгоритма су дале лошије перформансе од GEMV транспонованог алгоритма, што је и очекивано. Међутим, перформансе ових алгоритама се повећавају уколико постоје две различите матрице над којима треба применити SYMV, јер се по цени пребацивања n^2 елемената матрице из радне меморије рачунара у глобалну меморију графичког процесора, могу извршити два SYMV алгоритма.

Од имплементираних SYMV алгоритама, SSO се показао ефикаснијим у случају матрица мање величине. Са порастом величине матрице SSA постаје ефикаснији. Над SSA алгоритмом су примењене технике оптимизације условног извршавања, допуњавања матрице и преусмеравања показивача за рад са матрицама нерегуларних величина. Најбоље перформансе је дала техника допуњавања матрице.

Међутим, мерена су само времена извршавања кернела, а ова техника захтева додатни утрошак времена за алокацију и пренос података.

Најбоље перформансе су очекиване за преусмеравање показивача, како ова техника у већој мери елиминише додатни утрошак времена за вишеструко извршавање основа и ефикасно користи простор. Међутим, да би се ова техника успешно применила, потребно је извршити оптимизације кернела на нижем нивоу (нивоу инструкција) што излази изван опсега овог рада.

Даљи правци развоја се усмеравају на побољшање постојећих кернела, пре свега на оптимизацију преноса података између радне меморије рачунара и глобалне меморије графичког процесора, као и на оптимизације на нижим логичким нивоима.

7. ЛИТЕРАТУРА

- [1] Р. Дорословачки, Алгебра, Stylos, 1994.
- [2] R. Nath, S. Tomov, T. T. Dong / J. Dongarra, „Optimizing Symmetric Dense Matrix-Vector Multiplication on GPUs,“ у *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, Seattle, страна 6, 2011.
- [3] „NVIDIA GeForce GTX970 Specifications,“ [На мрежи]. Available: <http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-970/specifications>. [Последњи приступ 4.10.2016].
- [4] „Intel® Xeon® E5-1620V3 Specifications,“ [На мрежи]. Available: http://ark.intel.com/products/82763/Intel-Xeon-Processor-E5-1620-v3-10M-Cache-3_50-GHz. [Последњи приступ 15.10.2016].
- [5] J. J. Dongarra, J. Du Croz, S. Hammarling / R. J. Hanson, „An Extended Set of Fortran Basic Linear Algebra Subprograms,“ Argonne National Laboratory, 1986.
- [6] „cuBLAS Documentation,“ [На мрежи]. Available: <http://docs.nvidia.com/cuda/cublas/#axzz4NiHLI2V>. [Последњи приступ 15.10.2016].

Кратка биографија:



Горана Гојић рођена је у Дервенти, у Босни и Херцеговини 1992. године. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Рачунарства и аутоматике одбранила је у 2016. години.

IMPLEMENTACIJA KORISNIČKOG INTERFEJSA FENEK ALATA

FENNEC TOOL USER INTERFACE IMPLEMENTATION

Šandor Klebečko, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – Rad prikazuje implementaciju sistema za grafičko modelovanje šeme baze podataka, kao i povezivanje sa već postojećim modulom za kontrolu šeme baze podataka. Fenec alat je implementiran kao web aplikacija koja se oslanja na Angular i MVC šablon.

Abstract – This paper presents graphical modeling tool for database schema modeling. Emphasis of this paper is the user interface for modeling database schema. Fenec tool is implemented as a web application.

Ključne reči: Fenec, user interface for modeling, Angular, D3, Git, Python, django.

1. UVOD

Fenek je veb aplikacija otvorenog koda (*open-source*) za modelovanje i kontrolu verzija šema baza podataka. Napravljen je i dizajniran da omogući modelovanje više članova tima na istim šemama u okviru jednog projekta. Da bi zadovoljio današnje potrebe, dizajniran je tako da podrži pojam “agile and Just In Time” (JIT) modelovanje. Fenec takođe podržava kolaborativno modelovanje, pod čime se podrazumeva da distribuirani timovi mogu raditi nezavisno da bi postigli rešenje za postojeći problem. Predstavlja pojam integrisane verzije modelovanja slično kao GIT.

Fenek se gradi kao modularna aplikacija, gde je svaki modul tema jednog master rada. To su sledeći moduli: modul za kontrolu verziju šema baza podataka [1], implementacija korisničkog interfejsa Fenec alata i algoritmi za spajanje (merge) grana [2].

Tema ovog master rada je implementacija korisničkog interfejsa modelovanje u okviru Fenec alata.

U okviru rada je kreiran veb korisnički interfejs za modelovanje šema baza podataka koji je povezan sa već postojećim modulom za kontrolu verzija šema baza podataka.

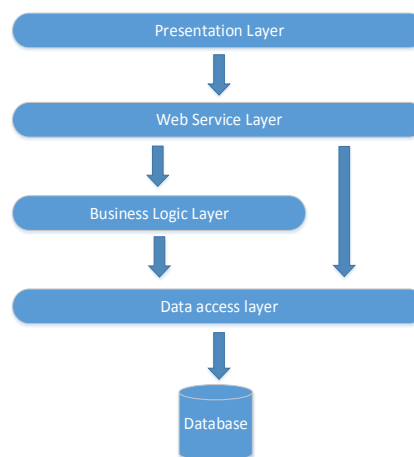
Kao veb aplikacija, pruža mogućnost rada u bilo kom trenutku iz svakog modernog veb pretraživača. Implementiran je tako da u što većoj meri podseća na već postojeće alate za modelovanje, da bi korisnicima izgledalo prirodno i intuitivno.

2. SPECIFIKACIJA SISTEMA

Fenek alat se zasniva na klijent – server arhitekturi i sastoji se od jedne serverske i jedne klijentske aplikacije. Pošto je u ovom radu fokus na implementaciji

korisničkog interfejsa za modelovanje, implementacija serverske strane Fenec alata neće biti prikazana. Ona je prezentovana u posebnom master radu [1] na koji se ovaj rad oslanja. Radi lakšeg praćenja, u nastavku je ukratko prikazana arhitektura celokupnog sistema, podskup CWM-a (Common Warehouse Metamodel [3]) korišćenog u aplikaciji, metamodel grafičkih elemenata, organizacija tabela za čuvanje podataka o projektu i organizacija čuvanja privremene i trajne promene modela. Nakon toga će biti prikazana i arhitektura prezentacionog sloja, tj interfejsa za modelovanje koji je tema ovog rada.

Fenek je dizajniran kao slojevita aplikacija, gde su komponente logički odvojene i komuniciraju međusobno. Pregled slojevite arhitekture je prikazan na slici 2.1.:



Slika 2.1. Slojevita arhitektura Fenec alata. (preuzeto iz [1])

Kao što je prikazano na slici 2.1. aplikacija je podeljena u 4 zasebna sloja: prezentacijski sloj, sloj web servisa, sloj poslovne logike i sloj pristupa bazi podataka.

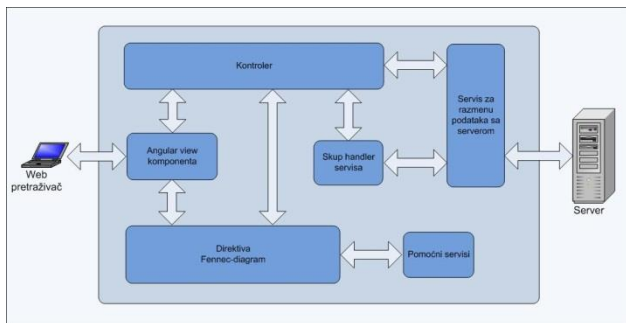
Prezentacijski sloj je jedini dostupan klijentima. Sastoji se iz tri dela: generalna administracija, administracija projekata i sekcija za modelovanje. Odgovoran je za omogućavanje korisničkih akcija za rukovanje projektima i modelovanje samih šema baza podataka. Prezentacijski sloj je dostupan klijentima preko internet pregledača (browser-a). Prezentacijski sloj komunicira isključivo sa slojem web servisa.

Sloj web servisa omogućava prezentacijskom sloju da pristupi poslovnoj logici potrebnoj za opsluživanje korisničkih akcija. Sloj web servisa pruža servise preko REST API-ja.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević.

Na slici 2.2. prikazana je arhitektura interfejsa za modelovanje.



Slika 2.2. Arhitektura interfejsa za modelovanje

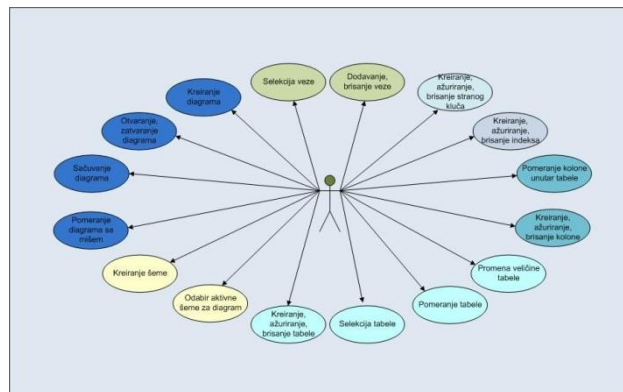
Fenek aplikacija za modelovanje je veb aplikacija koja se pokreće u bilo kojem modernom Internet pretraživaču. Rađena je u javascript programskom jeziku koristeći Angular JS[4] biblioteku. Aplikacije napisane na ovaj način se oslanjaju na MVC (model view controler) šablon. MVC šablon je jako popularan jer izoluje logiku aplikacije od view-a i podataka.

Direktiva **angular view komponente** je mesto gde angular injektuje komponente view-a. Izgled interfejsa za modelovanje u što većoj meri treba da podseća na desktop aplikacije, sa ciljem da korisnicima bude lakša i intuitivnija upotreba. Da bi se to postiglo, glavna stranica interfejsa za modelovanje je podeljena na tri dela. Na gornjoj strani je prikazan padajući meni, sa leve strane su prikazane informacije o projektu, na kojoj grani se korisnik trenutno nalazi, trenutno aktivan diagram, aktivna šema za taj diagram i paleta alatki za operacijama za modelovanje. Na desnom tj centralnom delu se nalazi direktiva **fennec-diagram**, preko koje se vrši modelovanje šema baze podata.

Fennec-diagram direktiva se sastoji od skupa servisa i funkcija koje omogućavaju vizuelnu reprezentaciju modelovanja. Zadužena je za renderovanje prikaza diagrama, iscrtavanje tabela i veza i vizuelizaciju korisničkih akcija, kao i za matematičke računice prilikom pomeranja i promene veličine tabele. Iscrtavanjem veza se takođe rukuje unutar ove direktive. Pored toga, prilikom promene veličine tabele, sakrivanja kolona koje nisu vidljive nakon promene se obavlja u ovoj direktivi. Direktiva je preko scope-a povezana sa kontrolerom i koristi funkcije pomoćnih servisa. Preko direktive se vrši cela vizuelizacija, dok kontroler kontroliše organizaciju za razmenu podataka sadrži metode za svaki tip promene podataka i definisanje odgovarajućih funkcija neophodnih za modelovanje. **Kontroler interfejsa** za modelovanje zadužen je za izvršavanje i kontrolisanje toka izvršavanja korisničkih akcija, kontrolu toka modelovanja i celokupnu kontrolu nad alatom. Logika kontrolera je podeljena u više usko definisanih servisa i svaki ima odgovarajući skup funkcija koje obavlja. Svi servisi su singltoni (*singletons*) koji se injektuju u kontroler. Kontroler preko servisa za razmenu podataka sa serverom šalje podatke na server. **Servis za razmenu podataka** je angular servis koji sadrži metode koje pomoću angular \$http servisa razmenjuju podatke sa

serverom. Serverska strana aplikacije je implementirana tako da beleži promene za svaku granu na kojoj se korisnik nalazi. Te promene su npr. izmena elementa šeme baze podataka ili izmena grafičkog prikaza nekog od elemenata. Servis za razmenu podataka sadrži metode za svaki tip promene koje server prima.

Interfejs za modelovanje treba da omogući korisniku, osnovni set funkcionalnosti za nesmetano modelovanje. Slika 2.3 prikazuje use case model alata za modelovanje

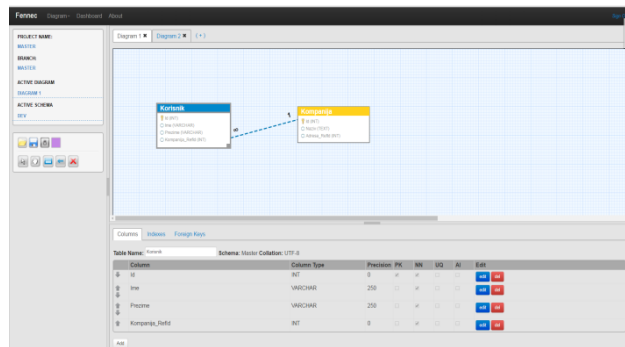


Slika 2.3. Use case model alata za modelovanje

Preduslov da bi se počelo modelovati je kreiranje novog dijagrama na grani u kojoj se korisnik trenutno nalazi. Nakon toga je neophodno kreirati šemu ako nije kreirana ni jedna za datu granu. Posle toga je moguće početi sa modelovanjem. Dodavanje tabele, kolona, indeksa, veza između tabele i na kraju čuvanje izmena na serveru. Svaka funkcionalnost sa slike pripada jednom tipu promene. Prilikom čuvanja dijagrama te promene se šalju na server.

3. IMPLEMENTACIJA SISTEMA

Fenek aplikacija je urađena u Python programskom jeziku i oslanja se na Django okvir za razvoj (*framework*) i AngularJS javascript biblioteku. Glavna stranica aplikacije je fajl **main.html** koji je Django šablon (*template*) stranica i koja nasleđuje **base.html** stranicu i u sebi uključuje parcijalnu stranicu **navbar.html**. U ovim html fajlovima se definišu svi javascript, css i ostali fajlovi koji su potrebni za pokretanje aplikacije. Glavna stranica je podeljena na tri dela. (slika 3.1)



Slika 3.1. Izgled aplikacije

Na gornjoj strani je prikazan menu bar, sa leve strane su prikazane informacije o projektu, na kojoj grani se korisnik trenutno nalazi, trenutno aktivan diagram, aktivna šema za taj diagram i paleta alatki za

modelovanje. Na desnom tj. centralnom delu se nalazi diagram za modelovanje.

Implementacija rešenja se može podeliti na tri dela:

- obrada korisničkih akcija
- vizuelna reprezentacija dijagrama
- razmena podataka sa serverskom stranom aplikacije

Za obradu korisničkih akcija je zadužen *fennecDiagramCtrl* kontroler koji poziva funkcije i pozive servisa za izvršavanje korisničkih akcija, kontrolu toka modelovanja i celokupnu kontrolu nad alatom. Logika kontrolera je podeljena u više usko specifičnih servisa i svaki ima odgovarajući skup funkcija koje obavlja. Kontrolerovi servisi su:

- servis za rukovanje dijagramima
- servis za rukovanje šemama
- servis za rukovanje tabelama
- servis za rukovanje kolonama
- servis za rukovanje stranim ključevima
- servis za rukovanje indeksima

Servis za rukovanje dijagramima predstavlja servis koji rukuje dijagramima unutar aplikacije. Zadužen je za dobavljanje i organizaciju podataka sa servera, dodavanje novog, ažuriranje dijagrama, prelazak na drugi dijagram, otvaranje i zatvaranje postojećih dijagrama i čuvanje podataka na serveru (*save diagram*).

Servis za rukovanje šemama predstavlja servis koji rukuje šemama. Šeme u bazama podataka služe da logički grupišu objekte kao što su tabele, pogledi (views), stored procedure itd. Šema je kontejner za objekte i samim time je neophodno da se kreira nakon kreiranja novog dijagrama. Svaki dijagram mora da ima definisanu jednu ili više šema. Šemama možemo dodeliti korisničku autorizaciju, tako da bi korisnik mogao da pristupi nekom objektu mora biti autorizovan. Šeme mogu da pripadaju bilo kom korisniku i pravo nad šemama se može preneti sa jednog na drugog korisnika.

Servis za rukovanje tabelama predstavlja servis koji rukuje tabelama. Sadrži metode za kreiranje nove tabele, brisanje tabele, brisanje elemenata tabele (objekat koji sadrži vizuelnu reprezentaciju tabele), brisanje veza između tabele. Ažuriranje tabele se vrši unutar direktive za modelovanje a podaci se ažuriraju u scopeu automatski zahvaljujući angularu.

Servis za rukovanje kolonama predstavlja servis koji sadrži skup funkcija za rukovanje kolonama unutar tabele. Sadrži sledeće metode:

- *addColumn* – metoda služi za dodavanje nove kolone u tabelu, inicializuje objekat kolone sa podrazumevajućim vrednostima koje će korisnik da promeni i na dugme sačuvaj, se dodaje u tabelu

- *saveColumn* – dodaje novokreiranu kolonu u tabelu. Ova metoda se poziva i nakon izmene kolone
- *removeColumn* – metoda omogućava brisanje kolone. Nakon poziva ove metode briše se kolona i veza ako postoji za kolonu i ažurira se redosled kolona (*ordinal*)
- *deleteColumnLink* – metoda briše vezu ako je postojala za kolonu.
- *moveColumnUp* – pomeranje kolone unutar tabele na gore
- *moveColumnDown* – pomeranje kolone unutar tabele na dole

Servis za rukovanje indeksima predstavlja servis koji rukuje indeksima. Indeksi služe da ubrzaju pretraživanje u bazama podataka. Ako ne postoji indeksirano polje u where klonuli, sql server mora da prođe kroz celu tabelu i da proveriti svaki red da vidi da li se poklapa sa upitom, što može da bude jako spora operacija u slučaju velikih tabela. Zbog toga, kod alata za modelovanje neophodna je podrška za rad sa indeksima.

Fenek aplikacija nudi sledeće indekse:

- *index* – odnosi se na indekse koji nisu jedinstveni.
- *unique* – odnosi se na indeks gde svi redovi moraju da imaju jedinstvenu vrednost za kolonu. Jedino se *null vrednosti* mogu ponavljati.
- *primary* – ponaša se isto kao *unique*, ali može da se samo jedan dodeli tabeli.

Primarni index se kreira automatski za kolonu koja je primarni ključ i može se obrisati samo ako se obriše kolona.

Servis za rukovanje stranim ključevima predstavlja servis koji rukuje stranim ključevima (*foreign key*). Strani ključ je način da se omogući referencijalni integritet unutar relacionih baza podataka. Strani ključ znači da vrednost iz jedne tabele mora da bude prisutan i u drugoj. Referencijalna tabela se zove roditelj (parent) tabela dok tabela koja sadrži referencijalni ključ je dete (child) tabela. Strani ključ se kreira automatski kad se kreira veza između dve tabele. Alat za modelovanje nudi mogućnost promena naziva stranog ključa, dalje moguće je dodavanje komentar za strani ključ i promena referencirane kolone. Kolona koja se referencira mora da ima indeks.

Za vizuelnu reprezentaciju dijagrama se koristi Fenek diagram direktiva. Zadužena je za renderovanje prikaza dijagrama, iscrtavanje tabele i veza i vizuelizaciju korisničkih akcija. Prima šest ulaznih parametara od kontrolera i to su sledeći:

- *data* – parametar koji sadrži listu tabele i veza koja će se iscrtati na diagramu
- *adiagram* – parametar koji prima podatke o trenutno aktivnom dijagramu. Projekat može da

ima više dijagrama, al je uvek aktivan onaj na kom se korisnik trenutno nalazi

- `activeSchema` – parametar koji sadrži podatke o aktivnoj šemi
- `stable` – objekat koji sadrži detalje o tabeli koja je selektovana, koristi se kod property dela alata za prikaz detalja o tabeli naziv, collation, kolone itd.
- `stableForeignKeys` – lista koja za selektovanu tabelu sadrži strane ključeve
- `stableIndexes` – lista koja za selektovanu tabelu sadrži indekse

Svi gore navedeni parametri su dvosmerno (*two way data binding*) povezani sa kontrolerovim scopovima (podacima u kontroleru), što nam omogućava jednostavnije ažuriranje podataka kako u kontroleru tako i u direktivi. Iscrtavanje tabele, tabela je grupa koja se gradi od sledećih SVG elemenata:

- jedan svg pravougaonik se koristi za okvir tabele
- pravougaonik i tekst polje – predstavlja naslov tabele
- grupa tekst polja i ikonica – predstavljaju kolone tabele
- pravougaonik -sivi kvadrat – služi za menjanje veličine table

Na određene delove tabele su vezani događaji kao što su `mouseClick`, `drag`, `resize` koji se pozivaju kad korisnik napravi interakciju sa tabelom. Na slici 3.1. prikazana je jedna tabela.



Slika 3.1. Tabela korisnik

Pomeranje tabele, na naslov tabele se vezuje događaj koji poziva `move()` metodu koja služi za pomeranje tabele. Metoda će translirati tabelu za udaljenost do nove pozicije. Tok izvršavanja metode:

- dobavljaju se početne koordinate tabele
- izračunava se udaljenost do nove pozicije tabele
- računa se aktuelna pozicija table
- proveravaju se koordinate tabele da li su one unutar dijagrama
- tabela se translira za pomerenu udaljenost
- ažuriraju se pozicije tabele i veze u scopovima
- osvežava se prikaz

Promena veličina tabele, prilikom selektovanja pojavljuje se mali sivi kvadrat na donjem desnom uglu koji kad se povuče menja se veličina. Kako se tabela sastoji od više delova, (a to su okvir, naslov i kolone) u

ovoj metodi prilikom promene veličine moramo da vodimo računa da ažuriramo sve delove tabele.

Tok izvršavanja metode:

- ažurira se mali kvadrat prilikom promene veličine tabele
- računa se visina i širina table
- proverava se da li je tabela veća od predefinisane minimalne vrednosti
- ažurira se naslov tabele
- ažuriraju se 'vidljivost' kolone – kolone koji nisu vidljive se sakrivaju
- ažuriraju se podaci o tabeli i vezi u scopeu
- osvežava se prikaz

4. ZAKLJUČAK

U okviru ovog rada kreiran je grafički veb interfejs za modelovanje šeme baze podataka unutar Fenek alata i povezan je sa već postojećim modulom za kontrolu verzija šema baze podataka koji je bio tema drugog master rada [1].

Za arhitekturu veb aplikacije se koristio MVC šablon, gde view predstavlja grafički alat za modelovanje. Kontroler je zadužen za izvršavanje i kontrolisanje toka izvršavanja korisničkih akcija, kontrolu toka modelovanja i celokupnu kontrolu nad alatom. Model se preko servisa razmenjuje sa serverskom stranom aplikacije.

5. LITERATURA

[1] Darko Čolak - Master rad, *Implementacija sistema za kontrolu verzija u okviru fenek alata* (2015)

[2] Nikola Latinović - Master rad, *Fennec softver za upravljanje verzijama šeme baze: algoritmi I pripadajući korisnički interfejs* (2016)

[3] <http://www.omg.org/spec/CWM/>, Common Warehouse Metamodel

[4] Angular JS, <https://angularjs.org/>

Kratka biografija

Klebečko Šandor je rođen 13.04.1985. godine u Novom Sadu. Fakultet tehničkih nauka upisao je 2004. godine, odsek Računarstvo i Automatika. Osnovne akademske studije završio je 2011.

AUTOMATSKO KONFIGURISANJE DISTRIBUIRANOG DMS SCADA SISTEMA

AUTOMATIC CONFIGURATION OF DISTRIBUTED DMS SCADA SYSTEM

Saša Krajina, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – U ovom radu je opisan postupak automatizacije konfigurisanja DistribuSyS-a (Distributed Systems) i Datapump-a u okviru DMS-a (Distributed Management System). Takođe, opisani su osnovni parametri i konfiguracija DistribuSyS-a, kao i korišćeni alati za realizaciju projektnog dijela zadatka. Automatizacija konfigurisanja DistribuSyS-a se sastoji u kreiranju inicijalizacionih fajlova (.init) parsiranjem odgovarajućeg Excel i XML dokumenta. Automatizacija konfigurisanja Datapump-a se sastoji u kreiranju SQL i PL skripti, parsiranjem odgovarajućeg Excel dokumenta i popunjavanjem parametara Datapump Tool-a.

Abstract – This paper describes the automation process of DistribuSyS configuration (Distributed Systems) and Datapump on DMS (Distributed Management System). Also, the basic parameters and configuration of DistribuSyS are described, as well as tools for implementation of project part of the task. The automation of DistribuSyS configuration consist in creating the initial files (.init) parsing appropriate Excel and XML documents. The automation of Datapump configuration consist in creating SQL and PL scripts, parsing appropriate Excel document and filling parameters of Datapump Tool.

Ključne reči: DistribuSyS, Datapump, DMS

1. UVOD

DistribuSyS je konfiguraciona opcija koja dozvoljava da više sistema dijele podatke i kontrolu. Konfiguracija distribuiranih projektnih rekorda zahtjeva poznavanje topologije mrežnih sistema i operativne akcije između sistema u okviru distribuiranog projekta.

U distribuiranim sistemima podaci su organizovani u skupovima podataka - dataset-ovima. Dataset-ovi obezbjeđuju način obrade podataka, takav da kao jedinice koriste grupe (skupove podataka). Podaci su grupisani u dataset-ovima i nad njima se definišu privilegije koje se odnose na sve podatke koji pripadaju toj grupi.

Cilj ovog rada je automatsko konfigurisanje DistribuSyS-a i Datapump-a na DMS sistemu. Automatizacija konfigurisanja DistribuSyS-a se sastoji u kreiranju .init fajlova parsiranjem odgovarajućih .xml i .xls fajlova. Nakon toga, potrebno je izvršiti odgovarajuće komande u CMD-u. Realizacija parsiranja .xml i .xls fajlova je pisana u programskom jeziku Python.

NAPOMENA:

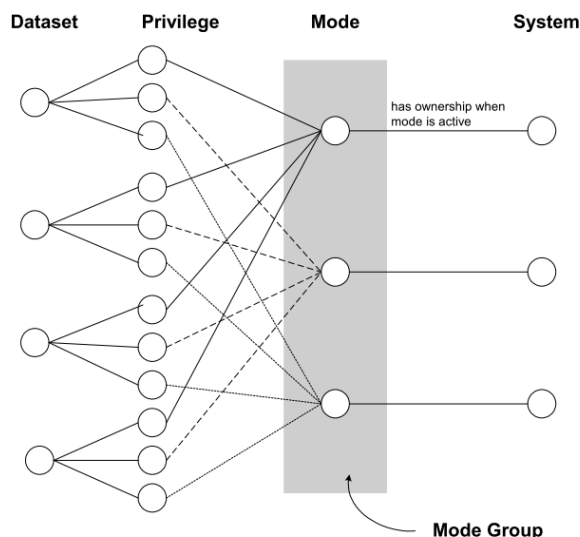
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Darko Čapko, docent

Automatizacija Datapump-a se sastoji u parsiranju odgovarajućeg .xls fajla i kreiranju .sql i .pl skripti, koje se nakon kreiranja distribuiraju ka drugim sistemima.

Navedeni fajlovi se kreiraju korišćenjem Datapump Tool-a. Realizacija parsiranja .xls fajla je pisana u programskom jeziku C# i korišćen je alat Visual Studio 2013.

2. Arhitektura DistribuSyS sistema

U cilju sprečavanja konflikta, kada je više sistema u stanju da prikuplja podatke sa terena, samo jedan sistem može da prikuplja podatke aktivno. Dakle, samo jedan sistem u to vrijeme može da "posjeduje" povezane rekorde.



Slika 1. Šema distribuiranog sistema

Na slici 1. je prikazana šema distribuiranog sistema. Distribuirani sistem (DistribuSyS), kao što je prikazano na slici 1. čine:

- ♦ **Dataset** - entitet unutar baze podataka koji povezuje podatke u grupe, nad kojima se definišu kontrolne privilegije koje se primjenjuju na sve podatke koji pripadaju toj grupi.
- ♦ **Privilege** - definiše kako se sistem ponaša. (npr. Da li je tačka u vlasništvu ovog sistema? Može li tačka biti kontrolisana ovim sistemom? Može li ovaj sistem mijenjati tačku? Kako je alarmiranje konfigurisano za ovu tačku na ovom sistemu?).
- ♦ **Mode** - se koristi da uspostavi različite operativne scenarije. Na primjer, jedan mod može biti podešen tako da privilegije, kao što su događaji (eventing) i alarmi (alariming), budu omogućene na nekoliko sistema, i drugi mod može biti podešen tako da je vlasništvo

omogućeno samo na jednom sistemu. Modovi se koriste za aktiviranje privilegija definisanih za dataset-ove i sisteme.

- ♦ **Mode Group** - Modovi se tretiraju kao međusobno isključivo na osnovu privilegija vlasništva. Samo jedan sistem može biti određen vlasnik dataset-ova u bilo kom operativnom modu. Prema tome, *Mode Groups* definišu modove koji su međusobno isključivi, i koji su nezavisni.

Prebacivanje između dva moda se naziva “*Mode-Switch*”. U takvim switch sistemskim privilegijama dataset-ovi mogu biti promijenjeni.

3. KORIŠĆENI ALATI

U ovom poglavlju će biti opisani alati koji su korišćeni prilikom realizacije praktičnog dijela rada.

3.1 Python programski jezik

Python je programski jezik opšte namjene. Objektno orijentisano, strukturno i aspekt orijentisano programiranje su stilovi dozvoljeni korištenjem *Python*-a. Ova fleksibilnost čini *Python* programski jezik sve popularnijim. *Python* se najviše koristi na *Linux*-u, ali postoje verzije i za druge operativne sisteme. U ovom radu, korišćen je *Python* na *Windows* operativnom sistemu.

Python je vrlo moćan programski jezik. Koriste ga mnoge firme i pojedinci, prvenstveno radi toga što je multifunkcionalan, te može izvesti i najzahtjevnije zadatke. *Python* omogućava kako pravljenje jednostavnih programa, tako i programiranje sa GUI-em (Graphic User Interface), pravljenje igara, integraciju u web stranice itd.

3.2 Extensible Markup Language (XML)

XML predstavlja markup jezik koji definiše skup pravila za enkodiranje dokumenta u format koji je čitljiv i za čovjeka i za mašinu. Definisan je od strane World Wide Web Consortium-a (W3C) kroz *XML* 1.0 specifikaciju i predstavlja besplatan i otvoren standard. Predstavlja tekstualni format podataka sa podrškom Unicode karaktera za veliki broj govornih jezika.

Kao dodatak tome što je *XML* dobro formatiran, može se provjeriti validnost *XML* dokumenta. Ovo znači da je moguće izvršiti provjeru da li su svi elementi u dokumentu opisani u skladu sa sintaksnim pravilima određenom *XML* šemom. *XML* procesori koji služe za obradu podataka mogu voditi računa o validnosti samog dokumenta, ali ukoliko vode, onda moraju imati mehanizam za prijavu grešaka ukoliko je dokument nevalidan.

3.3 Microsoft Excel

Excel je jedan od najefikasnijih programa za grafo-analitičku i tabelarnu obradu podataka. Dio je programskog paketa *Microsoft Office*. On posjeduje alate koji ubrzavaju i olakšavaju rad u radnim tabelama, jednostavan je za učenje, korišćenje i pokazuje brzo rezultate rada.

Za čuvanje koristi ekstenziju *.xls*, a u novijim verzijama *.xlsx* i *.xlsm*. *XLS* datoteka se bazira na binarnom pristupu, dok *XLSX* i *XLSM* implementira *Open Office*

XML format. U jednom dokumentu može postojati proizvoljan broj tabela i *sheet*-ova.

3.4 Microsoft Visual Studio

Microsoft Visual Studio je *Microsoft*-ovo integrisano razvojno okruženje. Koristi se za razvoj računarskih programa za *Microsoft Windows*, web sajtove, web aplikacije i web usluge. Visual Studio koristi *Microsoft*-ove integrisane razvojne platforme, kao što su *Windows API*, *Windows Forms*, *Windows Presentation Foundation*, *Windows Store*. Podržava različite programske jezike kao što su: *C*, *C++*, *VB.NET*, *Python*, *XML*, *HTML*, *JavaScript*.

4. DISTRIBUTED MANAGEMENT SYSTEM (DMS)

DMS je kolekcija aplikacija dizajniranih za efikasno i pouzdano praćenje i kontrolu cijele distributivne elektroenergetske mreže. Ponaša se kao sistem za podršku u odlučivanju, da asistira osoblju kontrolne sobe i osoblju na terenu sa prikazivanjem i kontrolisanjem električnog distributivnog sistema. Poboljšanje pouzdanosti i kvaliteta usluga u smislu smanjenja ispada, minimiziranja vremena ispada, održavanja prihvatljivog nivoa frekvencije i napona su ključni faktori isporuke jednog DMS sistema.

DMS je sistem složene arhitekture koji se najčešće sastoji od velikog broja različitih servera, klijenata i domen kontrolera (DC). Posebno treba istaći da se upravo u takvom okruženju rade testiranja veoma složenog softverskog proizvoda, koja omogućavaju precizne provjere rada softvera.

Prednosti DMS sistema:

- ♦ Smanjuje trajanje prekida napajanja.
- ♦ Poboljšava brzinu i preciznost predviđanja ispada (kvara).
- ♦ Smanjuje patrolu posade i vrijeme vožnje kroz poboljšanje lociranja ispada.
- ♦ Poboljšava operativnu efikasnost.
- ♦ Utvrđuje resurse posade neophodne za postizanje ciljeva restauracije.
- ♦ Efikasno koristi resurse između radnih regiona.
- ♦ Određuje kada je najbolje da planira zajedničku pomoć posada (ekipa).
- ♦ Povećava zadovoljstvo kupaca.

5. DISTRIBUSYS KONFIGURACIJA

Kada konfiguriramo distribuirani sistem, set rekorda treba da bude definisan da uputi sisteme kako da se ponašaju u datom modu.

Jedan rekord je definisan sa:

- ♦ *Dataset*-om – Ovo polje prikazuje povezani dataset, i koristi se u distribuiranom sistemu da poveže privilegije koje su dodijeljene svakom sistemu u svakom operativnom modu.
- ♦ *System*-om – Povezani sistem.
- ♦ *Mode*-om – Povezani mod.

Za datu kombinaciju *dataset-system-mode*, ove privilegije moraju biti definisane:

- ♦ *Ownership* (Vlasništvo) – Ovo polje pokazuje da li su, ili ne, rekordi koji se koriste u dataset-u u vlasništvu sistema. Prikupljanje i proračuni podataka se vrše samo kada je privilegija odobrena.
- ♦ *Alarm* – Ovo polje pokazuje da li rekordi koje koristi ovaj dataset mogu ignorisati alarme, silent update pregleda alarma, ili omogućiti punu podršku alarmu.
- ♦ *Event* (Događaj) – Ovo polje pokazuje da li su ili ne snimljeni logovi event-a za rekorde koje koristi ovaj dataset.
- ♦ *Control* (Kontrola) – Ovo polje pokazuje da li ili ne rekordi koje koristi ovaj dataset mogu biti kontrolisani (na primjer, zadate vrijednosti i digitalni izlaz komandi).
- ♦ *Collect* (Prikupljanje) – Koristi se od strane Timeline softvera da odluči ako Collect podaci pripadaju povezanom dataset-u da ih prikupljaju i čuvaju lokalno ili ako samo pripadaju sistemu, on treba da skladišti podatke.
- ♦ *Configure* (Konfiguracija) – Ovo polje pokazuje da li, ili ne, rekordi koje koristi ovaj dataset mogu biti konfigurisani.
- ♦ *Enable Statistics* (Omogućavanje statistike) – Ovo polje pokazuje da li, ili ne, rekordi koje koristi ovaj dataset mogu omogućiti statističke procese (na primjer, runtimes ili prosjeci).

Privilegije za alarm imaju tri moguće vrijednosti: *Ignore*, *Flash*, *Store*. Druga polja privilegije imaju dvije moguće vrijednosti: *Yes* (*Checked*), *No* (*Unchecked*).

5.1 Primjer konfiguracije

U nastavku će biti opisan jednostavni primjer konfiguracije DistribuSyS-a. Primjer pokazuje kako jedna konfiguraciona tabela za DistribuSyS (Tabela 1.) može biti prikazana za date pretpostavke:

- ♦ Dva Sistema:
 - NS - Novi Sad ima projektante i dispečere koji rade 24 časa,
 - SO - Sombor ima dispečere koji rade od 8:00-16:00 časova.
- ♦ Dva Moda:
 - DAY - Mod za dan (8:00-16:00 časova),
 - NIGHT - Mod za noć (16:00-8:00 časova narednog dana).
- ♦ Tri Dataset-a:
 - dmsSoDS - Dinamički podaci za područje Sombora,
 - dmsNSDS - Dinamički podaci za područje Novog Sada,

- nmDS - Podaci za mrežni model za Vojvodinu.

♦ Tri Privilegije:

- OWN - Vlasnik sistema,
- CONFIG - SCADA konfiguracija,
- CTRL – Kontrola.

Tabela 1. Primjer konfiguracije DistribuSyS-a

			Privileges		
Dataset	System	Mode	OWN	CONFIG	CTRL
dmsSoDS	NS	DAY	no	yes	no
dmsSoDS	NS	NIGHT	no	yes	yes
dmsSoDS	SO	DAY	yes	no	yes
dmsSoDS	SO	NIGHT	yes	no	no
dmsNSDS	NS	DAY	yes	yes	yes
dmsNSDS	NS	NIGHT	yes	Yes	yes
dmsNSDS	SO	DAY	no	no	no
dmsNSDS	SO	NIGHT	no	no	no
nmDS	NS	DAY	yes	yes	no
nmDS	NS	NIGHT	yes	yes	no
nmDS	SO	DAY	no	no	no
nmDS	SO	NIGHT	no	no	no

5.2 Datapump replikacija

Datapump replikacija se zasniva na DistribuSyS konfiguraciji.

Replikacija je proces, pomoću kojeg je moguće stvoriti kopije jedne ili više baza podataka, tako da čak i ako primarni server ne radi, podaci se još uvek mogu vratiti iz jednog od sekundarnih servera.

6. AUTOMATSKO KONFIGURISANJE DISTRIBUSYS-A I DATAPUMP-A NA DMS-U

Automatizacija ima izuzetno veliku i značajnu ulogu u DMS-u. Automatizacija raznih poslova u velikim distribuiranim sistemima dovodi do izrazitog povećanja produktivnosti u radu.

Mogu se automatizovati razni procesi, od onih najjednostavnijih ka mnogo složenijim. Prednosti koje donosi automatizacija jesu u tome što se sve performanse automatskih procesa mogu pratiti, mjeriti, kao i sve greške ili zastoji, što upravljanje čini kvalitetnijim, jednostavnijim i bržim.

6.1 Automatsko konfigurisanje DistribuSyS-a

Automatizacija konfigurisanja DistribuSyS-a se sastoji u kreiranju .init fajlova parsiranjem odgovarajućih .xml i .xls fajlova. Nakon toga, potrebno je izvršiti odgovarajuće komande u CMD-u.

Iz inicijalizacionih fajlova se čitaju podaci (korišćenjem odgovarajućih komandi) i upisuju u RealTime bazu. Imamo više inicijalizacionih fajlova:

- ♦ *doas_dataset.init* – u njega se upisuju svi dataset-ovi,
- ♦ *doas_system.init* – u njega se upisuju imena sistema,
- ♦ *doas_mode.init* – u njega se upisuju imena modova,
- ♦ *doas_privilege_type.init* – u njega se upisuju tipovi privilegija,
- ♦ *doas_privilege.init* – u njega se upisuju sve privilegije,
- ♦ *doas_mode_group.init* – u njega se upisuju mod grupe,
- ♦ *storeAndForward.init* – u njega se upisuju imena sistema koji međusobno komuniciraju.

Inicijalizacioni fajlovi se popunjavaju parsiranjem odgovarajućeg .xml (Slika 2.) i .xls fajla, korišćenjem metoda realizovanih u *Python* programskom jeziku.

[illegible]

Slika 2. Prikaz dijela xml fajla za DistribuSyS

6.2 Automatsko konfigurisanje Datapump-a

Automatsko konfiguriranje Datapump-a se sastoji u parsiranju odgovarajućeg .xls fajla i kreiranju .sql i .pl fajlova, koji se nakon kreiranja distribuiraju ka drugim sistemima. Ti fajlovi se kreiraju korišćenjem Datapump Tool-a.

Datapump Tool dozvoljava definisanje DMS sistema i putanja za replikaciju podataka između tih sistema. Kada definišemo sisteme i replikacione putanje između njih, onda možemo generisati .sql i .pl skripte da konfigurišemo te sisteme za replikaciju.

7. ZAKLJUČAK

Cilj ovog rada je bio automatizacija konfigurisanja DistribuSyS-a i Datapump-a na distribuiranom sistemu.

U prvom dijelu, potrebno je bilo realizovati automatizaciju konfigurisanja DistribuSyS-a na distribuiranom sistemu u programskom jeziku Python. Izvršeno je konvertovanje fajla iz .xls formata u .csv format, nad kojim je bilo jednostavnije primjeniti odgovarajuće metode za parsiranje.

U drugom dijelu, potrebno je bilo realizovati automatsku konfiguraciju Datapump-a na distribuiranom sistemu u programskoj jeziku C#.

Sam proces automatizacije konfigurisanja DistribuSyS-a i Datapump-a dosta je ubrzao rad na distribuiranom sistemu. Prilikom “ručne” realizacije ovih koraka, potrebno je bilo oko 5-6 sati, dok se automatski proces izvrši za manje od jednog minuta. Automatski proces je doprineo ne samo na brzini, nego je smanjena i vjerovatnoća ljudske greške prilikom izvlačenja parametara iz .xls i .xml fajlova i kreiranja odgovarajućih fajlova.

8. LITERATURA

- [1] Birman, Kenneth.: *Reliable Distributed Systems: Technologies, Web Services and Applications*. New York: Springer-Verlag, 2005.
- [2] Batista, Facundo.: *Python Enhancement Proposals*. Python Software Foundation. 2009.
- [3] Yih-Fang Huang; Werner, S.; Jing Huang; Kashyap, N.; Gupta, V.,: *State Estimation in Electric Power Grids: Meeting New Challenges Presented by the Requirements of the Future Grid*. Signal Processing Magazine, IEEE , vol.29, no.5, pp.33,43, Sept. 2012.Real-time database,
- [4] De, Alan.: *Visual SourceSafe: Microsoft's Source Destruction System*. Highprogrammer.com., 2009.
- [5] Ozsoyoglu, Gultekin, and Richard T. Snodgrass.: *Temporal and Real-Time Databases: a Survey*. Knowledge and Data Engineering, 2006.

Kratka biografija:



Saša Krajina rođen je u Vlasenici 1990. godine. Fakultet Tehničkih nauka u Novom Sadu upisao je 2010. godine, smjer Energetika, Elektronika i Telekomunikacije. Diplomirao je 2014. godine i iste godine upisao master studije, takođe na Fakultetu Tehničkih Nauka.

MATEMATIČKO MODELIRANJE OBARANJA NIVOVA PODZEMNE VODE NA LOKACIJI NOVE GLAVNE CRPNE STANICE U NOVOM SADU (NGC1) PRIMENOM GMS MODELA**MATHEMATICAL MODELING OF GROUNDWATER LEVEL LOWERING AT THE LOCATION OF NEW MAIN PUMPING STATIONS IN NOVI SAD (NGC1) USING GMS MODEL**

Snežana Utvić, *Fakultet tehničkih nauka, Novi Sad*

Oblast - GRAĐEVINARSTVO

Kratak sadržaj – U ovom radu su prezentovane neke od mogućnosti softverskog paketa MODFLOW-GMS 7.1 razvijenog u Americi na USGS-United States Geological Survey (Geološki zavod SAD) u svrhu hidrodinamičkog modelovanja dvodimenzionalnog i trodimenzionalnog toka podzemne vode kao i transportnih procesa, a koji je korišćen kao alat za kreiranje hidrodinamičkog modela za snižavanje nivoa podzemne vode iz građevinske jame, kako bi se izvršilo građenje objekta NGC-1 (nova glavna Crpna Stanica kod Železničkog mosta) u Novom Sadu. U okviru ovog rada je testiran rad matematičkog modeliranja i poređen sa uobičajenim rešavanjem datog problema, odnosno analitičkom analizom.

Abstract - This paper presents some of the features of the software package MODFLOW-GMS 7.1 developed in USA at the USGS United States Geological Survey (US Geological Survey) for purpose of hydrodynamic modeling of two-dimensional and three-dimensional groundwater flow and transport processes, and which was used as a tool to create a hydrodynamic model for lowering groundwater from construction pit, in order to perform the construction of the object NGC-1 (new main pumping station by the Railway Bridge) in Novi Sad. In this paper work of mathematical modeling has been tested and compared with the usual, analytical analysis resolution of a given problem.

Cljučne reči – podzemna voda, bunar, MODFLOW, GMS 7.1, matematičko modeliranje.

1. UVOD

Matematičko modeliranje, kao i 3D matematičko modeliranje, savremene SU metode sa ciljem da se dođe do rešenja kompleksnih problema koji se javljaju prilikom izučavanja kretanja podzemnih voda. Prednosti primene matematičkog modeliranja u odnosu na druge metode izučavanja podzemnih i površinskih voda je u mogućnostima stvaranja simulacija za složene uslove strujanja i prognoze efekata planiranog tehničkog rešenja za praktično neograničen broj simulacija, odnosno proračuna.

Matematički modeli mogu da obrade veliki broj podataka neophodnih za precizna rešenja i može da se dođe do saznanja o fizičkim, hemijskim i biološkim procesima koji se odigravaju u podzemnoj vodi i sredini kroz koju

ona prolazi, a koji se drugim metodama ne mogu ili se određuju sa nepouzdanošću.

Ovaj rad se sastoji iz dve celine teorijskog i praktičnog dela. Teorijski deo se bavi poznavanjem zakona strujanja podzemnih voda u različitim hidrološkim uslovima i njihovo opisivanje matematičkim izrazima. U praktičnom delu su prikazane neke od mogućnosti softverskog paketa MODFLOW-GMS 7.1 i testiranje analitičkih izraza u cilju isušivanja građevinske jame.

2. TEORIJSKI DEO – PODZEMNE VODE

Podzemne vode predstavljaju oko 90% svetskih rezervi dostupne pitke vode. Podzemne vode nastaju od vode koja ponire sa površine terena, iz površinskih tokova i bazena, ili kroz pukotine u stenama. Ta voda ponire kroz zemljište sve dok ne naiđe na nepropusnu podinu gde se formira izdan. Izdani podzemne vode se mogu formirati u poroznim sredinama i mogu biti:

- Sa slobodnom površinom (slučaj u ovom radu), i
- Pod pritiskom

Strujanje sa slobodnom površinom je ako je gornja granica slobodna površina, odnosno ako vlada atmosferski pritisak. Za strujanje pod pritiskom kažemo da je ako se propusna površina nalazi između dve nepropusne površine.

Za strujanje podzemne vode primenjuje se Darsijev zakon (H. Darsy, 1856. god.). Darsijeva brzina nije stvarna brzina kretanja podzemne vode kroz poroznu sredinu, jer je ona dobijena zanemarivanjem površine potencijalnog profila koju zauzimaju čestice materijala kao i šupljine u kojima je voda vezana kapilarnim silama. Darsijeva brzina filtracije glasi:

$$v = \frac{Q}{A}$$

Stvarna brzina se dobija kada se fiktivna brzina podeli sa poroznošću.

Darsijeva jednačina filtracije za strujanje u poroznim sredinama glasi:

$$v = KI_{\Pi}$$

v -brzina filtracije

K - koeficijent filtracije

I_{Π} –nagib pijezometarske linije $= -\frac{d\eta}{ds}$; gde je s pravac strujanja.

Darsijeva brzina je sredstvo da se pređe sa poroznog prostora skeleta zrna peska, na konkretnu sredinu, koja se

NAPOMENA:

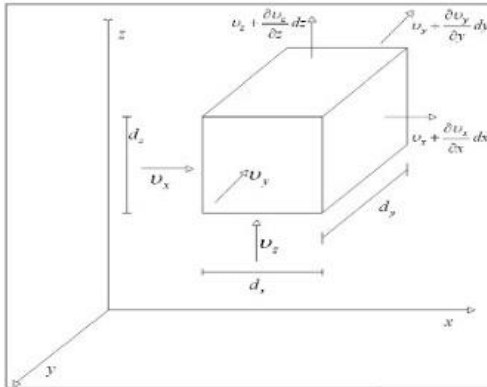
Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Ljubomir Budinski.

može opisati mikroskopskim parametrom kao što je koeficijent filtracije K.

Počevši od Poasonove jednačine strujanja koja je dobijena kombinovanjem uopštene Darsijeve jednačine i jednačine kontinuiteta, i ima oblik:

$$\frac{\partial^2 \Pi}{\partial x_1^2} + \frac{\partial^2 \Pi}{\partial x_2^2} + \frac{\partial^2 \Pi}{\partial x_3^2} = -\frac{W}{T}$$

Kako je strujanje podzemne vode potencijalno, to za njega važi i Laplasova jednačina koja opisuje prostorno stanje podzemne vode u uslovima ustaljenog tečenja u homogenoj i izotropnoj poroznoj sredini.



Slika 1 Ulazne i izlazne brzine strujanja

Na osnovu slike 1, mogu se napisati jednačine proticaja (Q_x, Q_y, Q_z) i jednačina kontinuiteta:

$$Q_x = (v_x + \frac{\partial v_x}{\partial x} d_x) d_y d_z - v_x d_y d_z$$

$$Q_y = (v_y + \frac{\partial v_y}{\partial y} d_y) d_x d_z - v_y d_x d_z$$

$$Q_z = (v_z + \frac{\partial v_z}{\partial z} d_z) d_x d_y - v_z d_x d_y$$

Sređivanjem ovih jednačina i uz uslov da je $Q_x + Q_y + Q_z = 0$ dobijamo jednačinu kontinuiteta:

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} = 0$$

Pre izvođenja Laplasove jednačine moramo pretpostaviti da je porozna sredina kroz koju struji voda homogena i izotropna i da je koeficijent filtracije konstantan ($K_x = K_y = K_z$).

$$-K \frac{\partial^2 \Pi}{\partial x^2} - K \frac{\partial^2 \Pi}{\partial y^2} - K \frac{\partial^2 \Pi}{\partial z^2} = 0$$

Kada se prethodna jednačina sredi, pomnoži sa (-1) i podeli sa (K), dobija konačan oblik:

$$\frac{\partial^2 \Pi}{\partial x^2} + \frac{\partial^2 \Pi}{\partial y^2} + \frac{\partial^2 \Pi}{\partial z^2} = 0$$

Dobijena diferencijalna jednačina predstavlja Laplasovu jednačinu. Sva strujanja koja zadovoljavaju Laplasovu jednačinu spadaju u potencijalno strujanje.

3. POJAM MATEMATIČKOG MODELA STRUJANJA PODZEMNIH VODA

Matematički model za simulaciju strujanja podzemnih voda, za osnovu ima primenu numeričkih metoda rešavanja diferencijalnih jednačina, kojima se izražava strujanje.

3.1. Teorijske osnove i metodologija

Problematika polja fizičkih veličina kao što su potencijal i brzina strujanja mogu biti efikasno rešeni različitim numeričkim metodama. Među najčešće korišćenim su: metoda konačnih razlika i metoda konačnih elemenata. Iako se odlikuje brojnim ograničenjima u pogledu geometrije i gustine mreže potrebne za dobijanje zadovoljavajuće tačnosti, prednost metode konačnih razlika je u jednostavnosti i efikasnosti implementiranja. Jednačina kretanja podzemne vode na kojoj je baziran Modflow je u obliku

$$\frac{\partial}{\partial x} \left(K_x \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_y \frac{\partial h}{\partial y} \right) + \frac{\partial}{\partial z} \left(K_z \frac{\partial h}{\partial z} \right) - W = S_s \frac{\partial h}{\partial t}$$

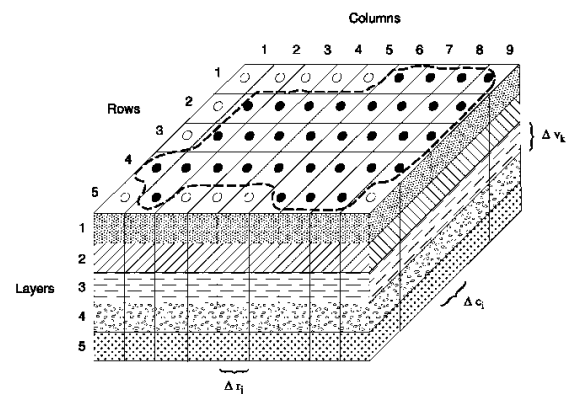
K_x, K_y, K_z – koeficijenti filtracije u pravcu x, y i z koordinatnih osa [m/s]

S_s – koeficijent specifične izdašnosti [1/m]

h – pijezometarski nivo podzemne vode [m]

W – zapreminski proticaj po jedinici zapremine izdani za jedinično vreme i predstavlja izvor/ponor vode [1/s]

Za rešavanje ove jednačine Modflow koristi numerički metod konačnih razlika. Korišćenjem ove numeričke metode, kontinualni domen simulacije zamenjuje se diskretnim koji se sastoji od mreže polja u 3D slučaju, a kontinualna forma parcijalne diferencijalne jednačine sa konačnim brojem algebarskih jednačina. Na slici 2, je prikazana diskretizacija izdani sa mrežom ćelija čija je lokacija definisana redom, kolonom i slojem gde su korišćeni indeksi i, j, i k.



Slika 2 Diskretizacija fiktivne izdani

3.2. Elementi matematičkog modela i granični uslovi

Matematički model strujanja podzemnih voda u suštini predstavlja skup matematičkih izraza, odnosno diferencijalnih jednačina, kojim se opisuje dato strujanje. Rešavanjem sistema jednačina za date granične i početne uslove, dobijaju se numeričke vrednosti traženih parametara hidrodinamičkog sistema i njihove promene

tokom vremena. Prilikom matematičkog modeliranja parametri sredine su uglavnom geometrija i filtracione karakteristike, dok su ulazni parametri pijezometrijski nivoi i proticaji, a kao rezultat ovakvog proračuna se takođe dobijaju nivoi i proticaji.

Osnovni elementi matematičkog modeliranja se mogu definisati kao:

- Skup jednačina koje su predstavljene softverskim paketom
- Reprezentativni parametri prirodne sredine
- Početni i konturni (granični) uslovi

Program Modflow omogućava korišćenje tri tipa graničnih uslova na više načina:

- Granični uslov prve vrste-konstantna vrednost pijezometrijske kote.
- Granični uslov druge vrste-konstantni doticaj/oticaaj sa granice ili sa modela (modul VEL1), simulacija neto infiltracije (modul RCH1).
- Granični uslovi treće vrste-simulacija evapotranspiracije (modul EVT1), doticaja/oticaaja iz reke kroz kolminarno rečno dno (modul RIV1), simulacija rada drenaže (modul DRN1) i generalni granični uslov (modul GHB1).

4. PRORAČUN STRUJANJA PODZEMNIH VODA ANALITIČKOM ANALIZOM I PRIMENOM SOFTVERSKOG PAKETA MODFLOW-GMS 7.1

Dat je primer jednog projekta koji je rađen na datoj lokaciji sa urađenim jednim predlogom obaranja nivoa podzemne vode.

Danas se za rešavanje problema u praksi koriste analitički izrazi za proste linijske slučajeve. U okviru rada korišteni su određeni analitički izrazi, koji su testirani: proračun grupe bunara pored reke pod pritiskom i proračun grupe bunara pored reke sa slobodnom površinom.

Na osnovu podataka koji su priloženi u projektu određene su granice u modelu. Dunav i kanal Dunav Tisa Dunav su simulirani kao granični uslovi sa poznatom vrednošću nivoa, odnosno vodostajem od 74,31 m.n.m. Usvojene su i osrednjene visinske kote kao i koeficijent filtracije (slike 3. i 4.):

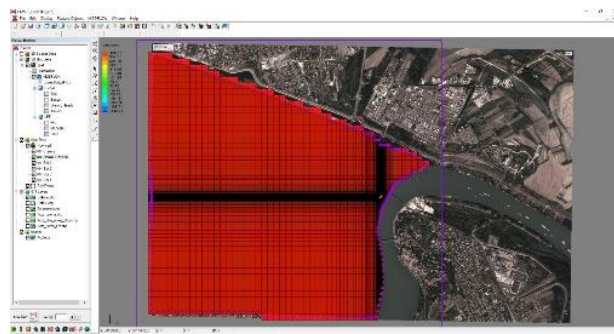
- Kota terena 79 m.n.m
- Kota nivoa podzemne vode 75,14 m.n.m
- Koeficijent filtracije $K=0,000375$ m/s
- Koeficijent transmisivnosti $T=0,003$ m²/s
- Nepropusna podina se nalazi na koti 55,5m.n.m.

Usvojena je predložena konstrukcija bunara sa dužinom nadfilterske cevi 11,5m, filterskom cevi od 8m i taložnikom od 4m.

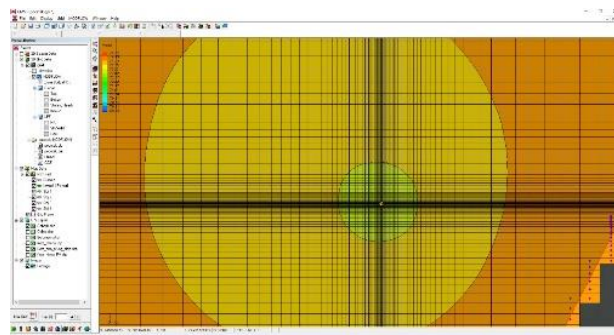
Razmatrano je i testirano više modela:

- Usamljeni bunar pored reke
- Grupa od osam bunara pored reke, kao i različite konfiguracije rasporeda bunara.
- Povećavanje broja bunara kako bi se dobilo željeno sniženje.

- Zadavanje koeficijenta filtracije po USBR-u po slojevima.



Slika 3 Pretvaranje konceptualnog modela



Slika 4 Prikaz modela oko usamljenog bunara

5. ZAKLJUČAK

Iako je primenom programskog paketa MODFLOW-GMS 7.1 bilo predstavljeno više različitih modela, svi su samo pokušaj da se prikaže koliko se strujanje podzemne vode može promeniti za različite uslove na terenu od promene kapaciteta crpljenja do promene graničnih uslova.

Iz svih predstavljenih modela i njihovog analiziranja se može zaključiti sledeće:

1. Promena gustine mreže u horizontalnom pravcu ne utiče na rešenja dobijena modeliranjem. Svakako treba progustiti mrežu oko bunara, jer su depresije najizraženije oko samih bunara i što su ćelije gušće rezultat će biti precizniji.
2. Promena gustine mreže u vertikalnom pravcu takođe ne utiče na krajnji rezultat. Ovim progustavanjem slojeva se svakako dobija jasnija slika šta se dešava sa modelom po vertikali, odnosno po slojevima.
3. Prikazani su i modeli sa različitim kapacitetima crpljenja iz bunara za zadatu konfiguraciju bunara, koja je predložena projektom, kao i za različite konfiguracije rasporeda bunara. Ovim su pokazane neke od mogućnosti modeliranja da se lakim i brzim testiranjem pomoću matematičkog modeliranja, može kombinovati više fizičkih problema.
4. U priloženom delu projekta je korišćen proračun pod pritiskom za snižavanje podzemne vode. U radu je upoređivan i priložen i proračun sa

slobodnom površinom. Model rađen u programu GMS 7.1 daje rešenja koja se nalaze između rešenja ova dva analitička postupka, ali je približniji rešenju dobijenom analitičkim proračunom sa slobodnom površinom. Ova dva postupka, sa predloženom konfiguracijom i brojem bunara, nisu zadovoljila rešenje u vidu snižavanja nivoa podzemne vode na potrebnu kotu. Kako nije bilo zadovoljavajućih rezultata ni kad je promenjena konfiguracija rasporeda bunara i kapaciteta crpljenja iz bunara, rešenje se potražilo u vidu povećanja broja bunara. Iz priloženog se može zaključiti da rezultati dobijeni analitičkim proračunom sa slobodnom površinom i rezultati dobijeni modeliranjem nisu zadovoljili traženo snižavanje nivoa podzemne vode i kada se povećao broj bunara, dok je analitički proračun pod pritiskom zadovoljio već kod projektom predložene konfiguracije i broja bunara.

5. Kako bi se pokazale još neke od mogućnosti matematičkog modeliranja u programu GMS 7.1 priložen je i model sa više slojeva sa različitim koeficijentima filtracije po USBR-u po slojevima. Ovo je još jedna velika prednost u odnosu na postupke analitičkih proračuna pored, već pomenutog u radu, hidrostatičkog pritiska.

S obzirom da je model dao veliki broj bunara potrebno je uraditi reviziju koeficijenta filtracije. Odnosno, potrebno je uraditi nova merenja na terenu. U radu je korišćen koeficijent filtracije koji je procenjen na osnovu priloženih rezultata projekta, gde je vršen proračun analitikom za strujanje pod pritiskom u čijim izrazima konfiguriše koeficijent transmisivnosti T.

U ovom se radu jasno vide neke od prednosti matematičkog modeliranja pri rešavanju složenih problema pri kretanju podzemne vode u odnosu na postupke analitičkih proračuna za iste probleme.

6. LITERATURA

- [1] Mladen Boreli: *Hidraulika*, Beograd, 1976.
- [2] Dr Milan Vuković, dipl.inž. i Anđelko Soro, dipl.inž.: *Dinamika podzemnih voda kroz rešene problem ustaljena strujanja*, Beograd, 1984.
- [3] Dr Dragan Kaluđerović: *3D matematički modeli kretanja podzemnih voda i transporta zagađenja u hidrogeologiji*, Beograd, 2009.
- [4] Georgije Hajdin: *Mehanika fluida*, Građevinski fakultet, Beograd, 2009
- [5] Grupa autora: *Aquaveo, Groundwater modeling system version 7.1.0, UT, USA*, build date, March 1, 2011 (tutorials Conceptual Model Approach II)

Kratka biografija:



Snežana Utvić rođena je u Rumi 1983. godine. Diplomski-master rad na Fakultetu tehničkih nauka iz oblasti Građevinarstvo – Hidrotehnika odbranila je 2016. godine.

MATEMATIČKO MODELIRANJE PLAVLJENJA KOD RAČINOVACA I RAJEVOG SELA PRIMENOM 2D MODELA U HEC-RAS-u**MATHEMATICAL MODELING OF FLOODING NEAR RACINOVCI AND RAJEVO SELO BY USING 2D MODELS IN HEC-RAS**

Aleksandar Narandžić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAĐEVINARSTVO

Kratak sadržaj – Zadatak rada je da se simuliraju uslovi koji su vladali u koritu Save u toku poplava, maja 2014. godine, kao i plavljenje teritorija Slavonije i Srema vodom, koja je u branjeni teren ušla kroz prodore na nasipu kod Račinovaca i Rajevog Sela. Da bi se sproveda analiza, neophodno je bilo da se formira 1D model neustaljenog tečenja u koritu reke, kao i 2D model plavljenja pomenutih prostora u programskom paketu HEC-RAS.

Abstract – The task of this work is to simulate the conditions that prevailed in the trough of Sava during floods in May 2014, as well as the flooding of the territory of Slavonia and Srem. Water which is defended terrain breakthroughs came through on the embankment near Račinovci and Rajevo Selo. To conduct the analysis, it was necessary to form a 1D model of unsteady flow in the river trough, as well as 2D model flooding territory mentioned in the software package HEC-RAS.

Ključne reči: 2D strujanje, plavljenje, 1D neustaljeno strujanje, granični uslovi, kalibracija, Maningov koeficijent

1. UVOD

Zadatak ovog rada je da analizira događaje u srednjem toku reke Save u toku aprila, maja i juna 2014. godine, kada su se izmerili istorijski maksimumi vodostaja i proticaja na ovoj reci i desile katastrofalne poplave.

Na hidrološkim stanicama u Županji, Gunji i Jameni registrovani su podaci o vodostajima i proticajima u svakom satu, te su iskorišteni kao granični uslovi. Sredinom maja kada su vodostaji premašili projektovane stogodišnje, na koje su dimenzionisani odbrambeni nasipi, dogodila su se dva proboja u kratkom vremenskom razmaku. Proboj u Rajevom selu prvi, kojem je prethodio slom nosećih slojeva tla, i sleganje nasipa u kruni, a voda se širila velikom brzinom i plavila okolni teren. U Račinovcima je nedugo zatim popustio nasip, prokvašen od velike količine kiša i enormno dugotrajne velikoe vode Save. Kada je voda dospela u branjeni prostor, spojila se sa plavnom vodom sa zapada i formirala paralelni tok reci Savi. Kao posledica ovih događaja poplavljena su naselja, plodno poljoprivredno zemljište, šume, tako da su štete bile ogromnih razmera.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Ljubomir Budinski.

Obuhvativši sva merenja direktno sa terena i dostupne podatke od nadležnih ustanova pristupiće se formiranju modela, čiji će rezultati prikazati faktičko stanje i uporediti se sa nekim poznatim parametrima.

2. SOFTVERSKI PAKET

Za ovakve probleme u praksi se koriste gotovi programi. Najfunkcionalniji softver za ovakvu analizu svakako je HEC-RAS (River Analysis System – Stady Flow, Water Profiles, US Army Corps of Engineering – Hydrologic Engeneering Center) koji se može preuzeti na www.hec.usace.army.mil/software/hec-ras/ gde se može naći i detaljno uputstvo o korišćenju.

Program se koristi za hidrauličke proračune ustaljenog i neustaljenog strujanja za otvorene tokove po pravilu nepravilne geometrije uz unutrašnje i spoljašnje granične uslove. Jednačina na kojoj se zasniva proračun ustaljenog strujanja je energetska linijska jednačina, gde se gubici definišu trenjem preko Maningovog koeficijenta i lokalnim gubicima.

Osnovne jednačine koje program koristi su Saint-Venant-ove jednačine koje podrazumevaju jednačinu održanja mase i jednačinu održanja količine kretanja. Ove jednačine su parcijane diferencijalne jednačine hiperboličkog tipa, ali uz određene pretpostavke se uprošćavaju. Tako uprošćene se putem pogodne numeričke šeme rešavaju, tako da i sam program radi na principu rešavanja parcijalnih diferencijalnih jednačina metodom konačnih razlika, koristeći za to Preissmann-ovu šemu.

Neophodno je istaći da je novitet u softverskom paketu dvodimenzionalno modeliranje plavljenja uz prikaz strujnica. Dvodimenzionalno strujanje se bazira na parcijalnim diferencijalnim jednačinama osrednjenim po dubini, koje nemaju analitičko rešenje. Rešavaju se isključivo numerički putem metod konačnih elemenata. Razvija se računaska mreža sa elementarnim ćelijama, a unutar svake se zasebno rešavaju jednačine i dobijaju podaci o dubini, brzini, nivou slobodne vodene površine itd.

3. USPOSTAVLJANJE MODELA PLAVLJENJA

Reka Sava izvire u Slovenskim Alpama i nastaje od Save Bohinjke i Save Dolinjske, te se kod beograda uliva u Dunav. Dugačka je 966 km, a velikim delom svog toka predstavlja prirodnu granicu između Hrvatske i BiH. Sa stanovišta pritoka najvažnije su joj desne pritoke, Una, Vrbas, Bosna, Drina, Kolubara.

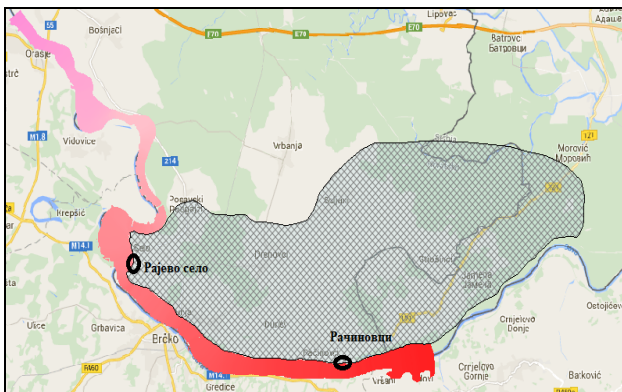
U radu se proučava korito reke od km 206+000 do km 264+892, te branjeni teren na teritoriji Slavonije i Srema, omeđen savskim nasipom na jugu i zapadu, a tokom reka Studva i Bosut na severu i istoku (slika 1).



Slika 1. Pregledna situacija

Rad na formiranju digitalnog modela terena je izuzetno zahtevan posao i traži angažovanje u programima za digitalnu obradu i georeferenciranje karata i mapa kakav je ArcGIS. Formiran digitalni model terena se spaja sa digitalnim modelom korita, a posredstvom HEC-GeoRAS-a se dobija konačan oblik modela spreman za proračun. Preostaje da se postavi mreža za proračun 2D strujanja u zoni plavljenja. Mreža se generiše na prostoru koji obeležimo, a čije granice su nam poznate uz niz zadatih ulaznih parametara (slika 3.). Mreža koja se koristi pri proračunu 2D strujanja ima sledeće karakteristike:

- Ukupan broj ćelija 120.701
- Najveća ćelija 38.245,24 m²
- Najmanja ćelija 100,83 m²
- Prosečna ćelija 2.510,47 m²



Slika 2. Proračunska mreža plavljenja i lokacije proboja

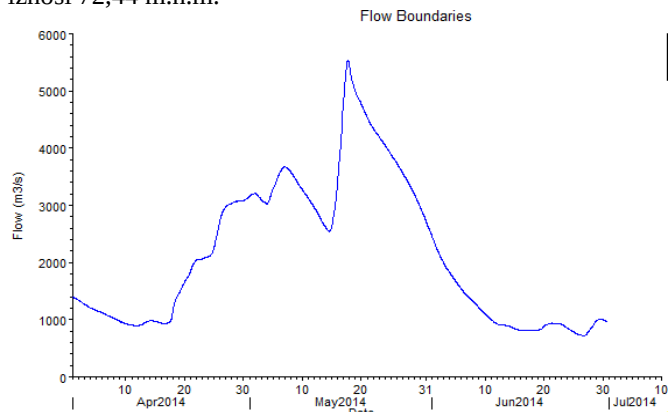
Formiranje proboja je poslednji korak pri uspostavljanju modela. Na mestima gde dolazi do prelivanja vode i početka 2D strujanja, definiše se niz parametara. Najbitnije stavke su da je početak prelivanja kod Rajevog Sela u 14:55, da je kota krune nasipa na tom mestu 86,75 m.n.m, a da je kota dna preliava 84,50. U Račinovcima je nasip popustio u 16:25 istog dana 17.maja, pri koti vode 85,30 m.n.m, dok je nasip na koti 85,45 m.n.m, a dno prelivanja na 82,12 m.n.m.

3.1 Granični uslovi

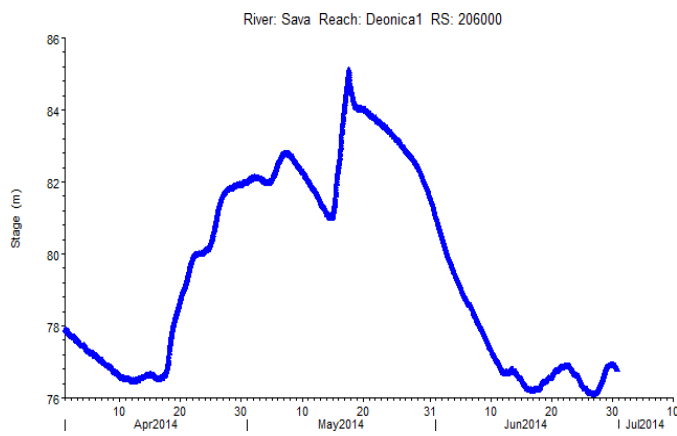
Da bismo nakon unete geometrije modela mogli pustiti proračun, granični uslovi moraju biti zadati. Za uzvodni granični uslov postavljen je časovni hidrogram za period

april, maj, jun 2014.godine sa hidrološke stanice u Županji prikazan na slici 3.

Za nizvodni granični uslov uzeta je kota slobodne vodene površine u zavisnosti od vremena – nivogram na hidrološkoj stanici u Jameni, gde je vodostaj meren na svakih pola sata njegov dijagram je na slici 4. Kota slobodne površine vode proistekla je prostim dodavanjem merenih vodostaja na kotu „0“ vodomera, koja u Jameni iznosi 72,44 m.n.m.



Slika 3. Uzvodni granični uslov-hidrogram kod Županje



Slika 4. Nizvodni granični uslov-nivogram kod Jamene

4. ULAZNI PODACI

Za hidrauličku analizu u projektu regulacije reka potrebne su tri kategorije podataka: geometrijski podaci, hidrološki podaci i podaci o nanosu, koji će u ovom slučaju biti van domena istraživanja.

Projekti zaštite od poplava vezuju se za veliku vodu određene verovatnoće pojave, dok se regulacija osnovnog korita dimenzioniše na male vode nekog trajanja ili verovatnoće. Hidrološki podaci se dobijaju merenjem na stanicama duž toka ili se računaju. Rešavanje ovog zadatka ne bi bilo moguće bez podataka o proticaju i kotama, koji predstavljaju uzvodne i nizvodne granične uslove, a preuzeti su sa Republičkog hidrometeorološkog zavoda Srbije i Državnog hidrometeorološkog zavoda Hrvatske.

Geometrijski podaci o koritu i inundacijama, trasi vodotoka, mostovima i propustima su neizostavni deo hidrauličkog modela. Od njihove tačnosti i verodostojnosti umnogome zavisi tačnost kompletnog proračuna. Manipulacije sa Manningovim koeficijentom se rade upravo zbog geometrijskih nesavršenosti i interpolacija između profila koji ne odgovaraju stvarnosti.

5. ANALIZA REZULTATA

Tačnost rezultata proračuna zavisi od iskustva i sposobnosti projektanta da postavi hidraulički model, da li je izabrani model odgovarajući za predmet hidrološke analize i od tačnosti ulaznih podataka.

5.1 Kalibracija modela

Da bi se nesavršenosti pri postavljanju modela svele na minimum, model se kalibriše odnosno tarira na određene poznate situacije. U ovom radu model se kalibrisao prema ustaljenom strujanju sa proticajima 500, 1000, 2000 i 3000 m³/s, koji su bili ujednačeni 7 dana. Prilikom ovih vrednosti proticaja, voda se zadržavala u glavnom koritu, pa je Maningov koeficijent određivan samo za glavno korito. Za uzvodni granični uslov uzeta je kriva protoka kod Županje, a za nizvodni poznati nivo kod Jamene za pomenute proticaje. Puštani su proticaji, a za svaki se deonica delila na 3 etape, na kojima su se usvajali različiti Maningovi koeficijenti. Radi usklađivanja sa merenim vrednostima na kontrolnoj stanici u Gunji, za svaki protok su etape različitih stacionaža, a pregled usvojenih Maningovih koeficijenata za glavno korito nalazi se u Tabeli 1.

Tabela1. Prikaz Maningovih koeficijenata po deonicama

Q(m ³ /s)	231+937	216+746	214+854	212+783	206+000
500	0.007	0.0125	0.015	0.015	0.015
1000	0.007	0.01	0.01	0.015	0.015
2000	0.007	0.008	0.008	0.008	0.011
3000	0.007	0.007	0.007	0.007	0.008

Za inundaciju prilikom kalibracije, Maningov koeficijent je 0,05, osim za protok 3000 m³/s kad je usvojen 0,03. Tabela 2. predstavlja rezultate dobijene kalibracijom u poređenju sa merenim rezultatima.

Tabela 2. Poređenje rezultata kalibracije

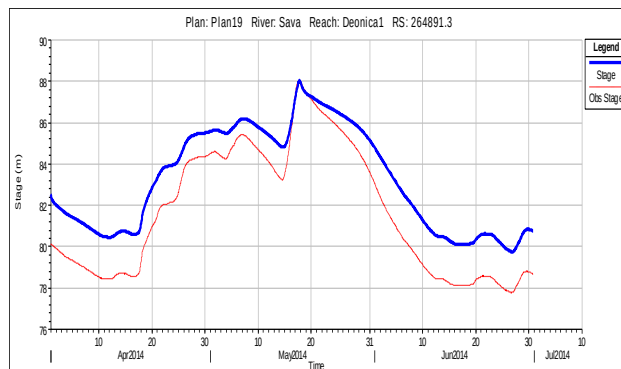
Проток (m ³ /s)	Ж у п а њ а			Г у њ а		
	Мерена кота нивоа (mnm)	Кота нивоа на моделу (mnm)	Разлика (m)	Мерена кота нивоа (mnm)	Кота нивоа на моделу (mnm)	Разлика (m)
500	77.05	77.08	0.03	76.35	76.33	0.02
1000	79.06	79.03	0.03	78.01	78.06	0.05
2000	81.84	81.81	0.03	80.55	80.58	0.03
3000	84.05	84.04	0.01	82.80	82.76	0.04

5.2 Rezultati plavljenja dobijeni proračunom

Prilikom modeliranja poplavnog talasa za glavno korito usvojen je Maningov koeficijent 0,025, a za inundaciju 0,05, jer se samo sa tim koeficijentima postiglo slaganje merenih nivoa i nivoa u modelu pri vršnim proticajima. Već pri proticajima ispod 4200 m³/s, dolazi do neslaganja nivoa i po 2 metra, što je prikazano dijagramom na slici 5.

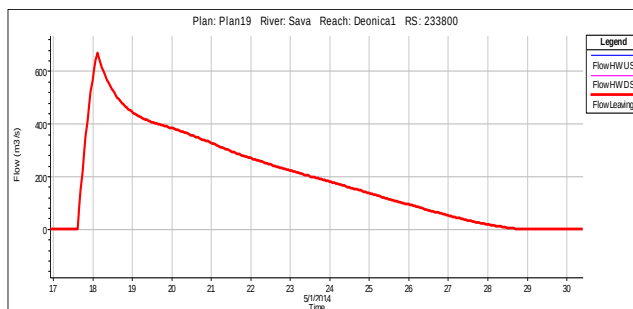
Dijagram nam prikazuje klasičan problem Maningovog koeficijenta i njegovu zavisnost od proticaja. Da bi se postiglo poklapanje nivoa pri svim proticajima za ovaj model, trebali bi se usvojiti Maningovi koeficijenti za svaki profil posebno u zavisnosti od proticaja, kao i

horizontalni koeficijenti za glavno korito po etapama dobijenim kalibracijom.



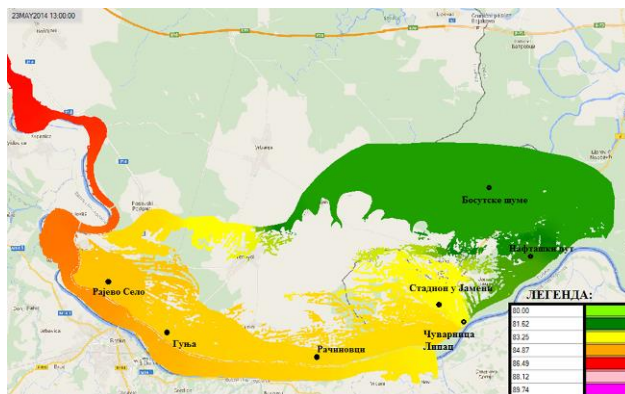
Slika 5. Dijagram merenih i računatih nivoa na stanici Županja

Proboji na nasipu, radiće kao bočni preliv u hidrauličkom smislu. Prilikom proračuna za svaki preliv dobijeni su podaci o proticajima preko preliva, nivoima vodenog ogledala, uzvodnom i nizvodnom proticaju. Jednu od analiziranih stvari prikazaćemo za proboj u Rajevom selu na slici 6.



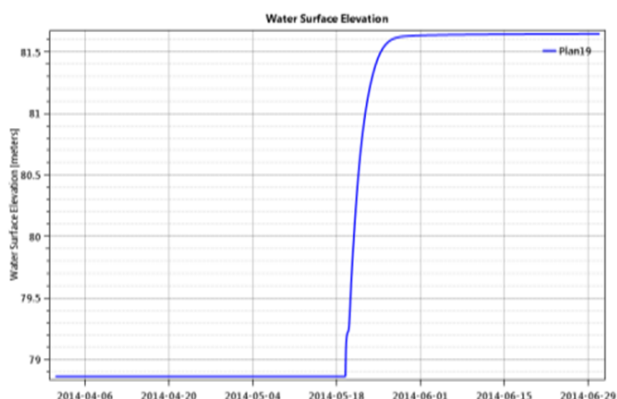
Slika 6. Proticaj preko preliva u Rajevom selu u vremenu

Sa slike se vidi da se proboj otvarao 12 sati do konačne otvorenosti, a da je proticaj, sa opadanjem nivoa u glavnom koritu vremenom slabio. Vršni proticaj je bio ok 670 m³/s. Za Račinovce je u radu prikan karakterističan dijagram za zavisnost proticaja preko preliva od vremena. Voda koja se prelila popunjavala je branjeni prostor i plavila prostor uz nasip, kao i niske teritorije udaljenije od nasipa. Poplavljeno je više od 24.000 ha površine, a kote do kojih je voda dosegala zavisile su od blizine, odnosno udaljenosti od nasipa. Slika 7. Prikazuje stanje na kraju plavljenja, odnosno pokrivenost teritorije poplavama i nivo do kojeg je dostizala voda u određenim oblastima.



Slika 7. Poplavljena oblast sa kotama nivoa vode

Određena mesta su tokom razlivanja vode postale retenzije u kojima se voda akumulirala i time ukrotila vodenu stihiju. Slika 8. kazuje o tome na lokaciji u šumi nadomak reke Studve.



Slika 8. Zavisnost kote nivoa vode u bosutskim šumama u toku vremena

Sa dijagrama na slici 8. i na osnovu slike 7. možemo zaključiti da se ova oblast ponaša kao retenzija i da može umnogome doprineti rasterećenju od velikih voda Save i učinkovitijoj odbrani od poplava naselja i teritorija nizvodno.

5. ZAKLJUČAK

Postaviti hidraulički model nije jednostavna stvar, jer ne postoje tačna uputstva. Dosta stvari je na projektantu da svojom intuicijom pretpostavi ponašanje modela u određenim situacijama. Tačnosti i kvalitetu modela, da bi se mogao primeniti na praktične probleme umnogome doprinosi kvalitet geodetskih merenja i podataka. Hidrološki parametri su sledeća stavka, bez koje proračun ne može, i bitno je da su redovno ažurirani. Uz navedene stavke, prilikom modeliranja nailazimo na probleme prilikom formiranja 2D mreže, koji se opisuje kao problem sa numeričkom stabilnošću ćelija mreže, što je jedan od nedostataka HEC-RAS-a 2D. Ovaj softverski paket je relativno mlad, posebno njegova sekcija o dvodimenzionalnom strujanju.

U skladu sa savremenim tendencijama i sve potrebnijoj analizi rizika od poplava mora mu se dati na značaju, i unaprediti njegova uloga u aktivnostima u ove svrhe.

Pored analize rizika od poplava, odnosno izrade karata plavnih zona, gde GIS tehnologije takođe imaju jaku bazu, ovakvi modeli se mogu koristiti prilikom prognoze vodostaja tokom odbrane od poplava, što može da bude veoma korisno, simulaciji ekstremnih hidroloških situacija, prilikom gradnje u priobalnom području, za pripreme projekata rekonstrukcije objekata u rečnom koritu i objekata za evakuaciju voda.

Konačno, ključna stvar je unapređenje mera i strategija odbrane od poplava, odnosno uključivanje baš ovakvih teritorija kao potencijalnih retenzija u priobalju. Jasno je da prema klimatskim promenama moramo postaviti jasan stav u narednom periodu, i da im moramo pristupiti na ozbiljan način.

Sa stanovišta hidrologije, prema studijama mnogobrojnih evropski i svetskih instituta, nesumljivo će u sledećim decenijama doći do povećanja proticaja i nivoa u rekama, a naročito savskim koritom, jer su njegove geografske prilike posebne.

Imajući u vidu takve procene i tendencije Evropskih organizacija za očuvanje životne sredine i projekte odbrane od poplava na način da se rekama ostavi prostora, da slobodno plave i ne ugrožavaju infrastrukturne objekte, ljudske živote i poljoprivredno zemljište, moramo početi baviti se objektima kakve su retenzije. One spadaju u aktivne mere odbrane od poplava, koje su znatno učinkovitije od klasične pasivne odbrane bedemima i nasipima, jer je priroda „ćudljiva“.

U svrhu detaljnije analize ovog područja i želje da se uistinu razmotre mogućnosti aktiviranja retenzije na ovom području, za model rečnog korita bi se morali postaviti Manningovi koeficijenti na način da svakom profilu dodelim vrednosti koeficijenta po vertikali u zavisnosti od proticaja.

Takođe, poželjno bi bilo tren detaljno snimiti savremenim, ali veoma skupim geodetskim metodama, kakvo je nesumljivo lasersko snimanje poprečnih profila korita. Sa ovako moćnim programskim alatom uz adekvatne ulazne podatke, imaćemo jasne i tačne modela i verodostojne simulacije dešavanja u prirodi, te ćemo moći predvideti naredne krizne situacije.

6. LITERATURA

se spisak korišćene literature, kao u ovom uputstvu. Sva navedena literatura mora da bude citirana. Na kraju rada daje se kratka biografija autora (font veličine 9) i prilaže slika.

7. LITERATURA

- [1] HEC-RAS User's manual, US Army Corps of Engineering, Hydrologic Engineering Center
- [2] Miodrag Spasojević, Numerička hidraulika, Građevinski Fakultet u Subotici, Univerzitet u Novom Sadu, 1996.
- [3] Miodrag B. Jovanović, Osnove numeričkog modeliranja ravanskih otvorenih tokova, Univerzitet u Beogradu, Građevinski Fakultet Beograd, 1998.
- [4] Dragutin Muškatirović, Regulacija reka, II izdanje Građevinski Fakultet Beograd, 1979.

Kratka biografija:

Aleksandar Narandžić rođen je u Zagrebu 31. decembra 1990. godine. Bachelor rad iz oblasti Hidrotehnički objekti i sistemi odbranio je 2013. godine na Fakultetu Tehničkih Nauka. Master rad na Fakultetu Tehničkih Nauka iz oblasti Hidraulike je odbranio 2016. godine na katedri za hidrotehniku.

**RAZVOJ GEOGRAFSKOG INFORMACIONOG SISTEMA ZA SAOBRAĆAJNICE
LOKALNIH SAMOUPRAVA****DEVELOPMENT OF A GEOGRAPHIC INFORMATION SYSTEM FOR LOCAL
GOVERNMENT ROADS**

Milan Marinković, Bojan Matić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAĐEVINARSTVO

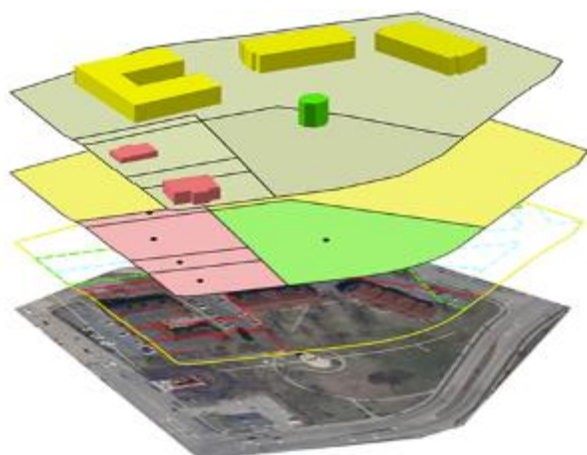
Kratak sadržaj – U radu su navedeni osnovni pojmovi o geografskom informacionom sistemu. Nabrojane su sve prednosti uvođenja GIS-a. Data je struktura troškova uvođenja GIS-a. Navedene su komponente GIS-a. Svaka komponenta je detaljno objašnjena. Navedeni su i načini formiranja baze podataka. Prikazana je podjela podataka za GIS i razlika između njih.

Abstract – In this paper basic information about geographic information system is presented. All advantages of installing GIS are listed. The costs structure of installing GIS is shown. The components of GIS are described. Each component is explained in more detail. The ways of forming the database are shown. The division of data for GIS and difference between them is shown.

Cljučne reči: GIS, komponente, baze, podaci

1. UVOD

Geografski informacioni sistem (GIS) je organizovan skup hardvera, softvera, osoblja, podataka i mreža radi efikasnog prikupljanja, obrade, skladištenja, ažuriranja, rukovanja, analiziranja, modelovanja, prenosa i prikaza svih oblika prostornih informacija. Geografski informacioni sistem nam omogućava da našim podacima pridružimo njihov prostorni položaj. Ovako organizovani podaci prikazani na digitalnim mapama (Slika 1. *Primjer digitalnih mapa*) nose informacije visokog kvaliteta koje doprinose efikasnijem poslovanju i boljem donošenju poslovnih odluka.

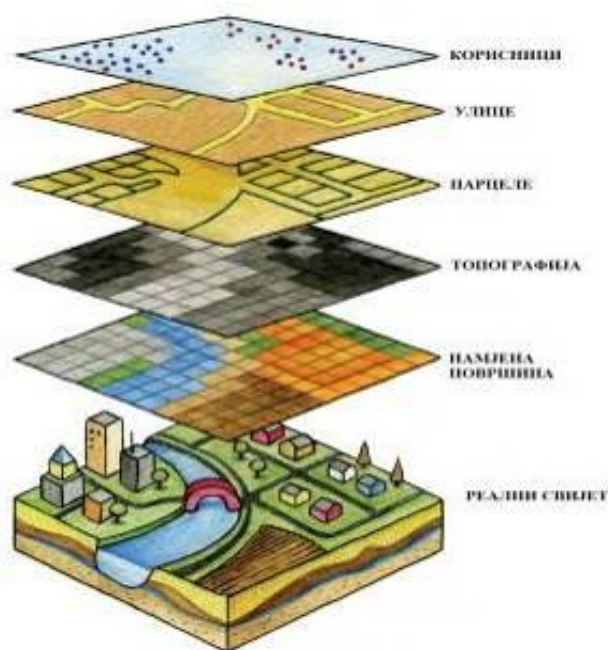


Slika 1. *Primjer digitalnih mapa*

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Bojan Matić.

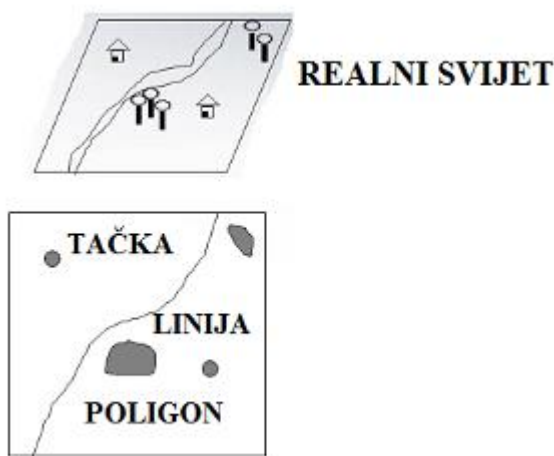
GIS predstavlja kompleksan sistem sa podacima o prostoru, organizovanim tako da se mogu prikazivati u tematskim kartama, kao na primjer: osnovne geodetske podloge, topografske, hidrološke, geološke, karte namjene površina, karte sa katastarskim parcelama, karte putne mreže, saobraćajne, ekološke, karte sa instalacijama, administrativne karte i slične karte. Sve karte dijele isto geografsko područje na osnovu pojedinačnih parametara kao što je namjena površina (Slika 2. *Primjer podele istog područja na osnovu različitih parametara*). Preklapanjem tih pojedinačnih listova dobijamo sinteznu kartu.



Slika 2. *Primjer podele istog područja na osnovu različitih parametara*

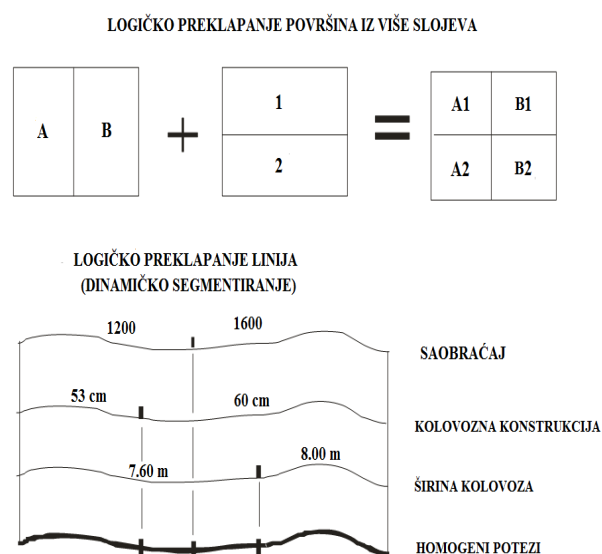
U okviru informacionog sistema za puteve, održavaju se oni dijelovi GIS-a koji se odnose na mrežu vangradskih magistralnih, regionalnih ili lokalnih (gradskih) puteva. Ove aktivnosti se moraju sprovesti uz saradnju sa ostalim organizacijama u Srbiji koje rade na GIS sistemu uz obavezu da se održava međusobna kompatibilnost i integracija. Softverski paketi koji služe za upravljanje geografskim informacionim sistemom obično se sastoje od sljedeće dvije komponente: grafičke i alfanumeričke baze podataka. Grafička komponenta generiše tematske karte i ostale karte na osnovu koordinata. Alfnumeričke baze podataka manipulišu atributima povezanim za pojedine grafičke objekte. GIS razlikuje tri vrste grafičkih objekata (Slika 3. *Primjer prikazivanja objekata u GIS-u*)

za koje mogu biti vezani neki atributi iz relacionih baza podataka. Tri vrste prikaza grafičkih objekata su: tačke (primjer: vertikalni saobraćajni znaci, stubovi rasvjete), linije (primjer: putevi, rijeke, instalacije), poligoni (primjer: parcele, teritorijalne jedinice, objekti).

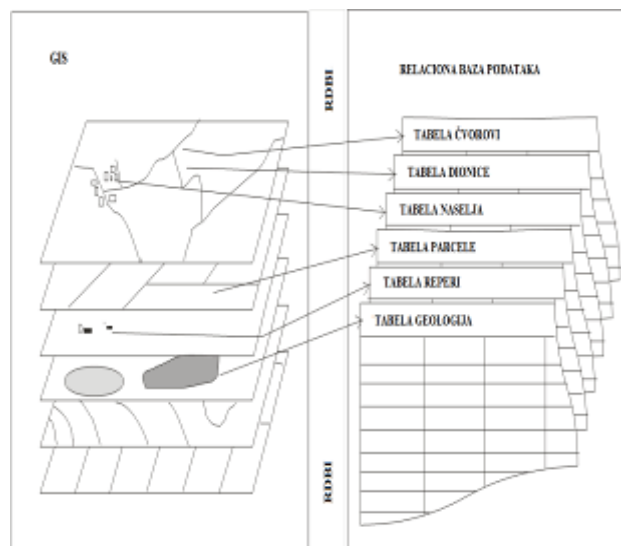


Slika 3. Primjer prikazivanja objekata u GIS-u

Posebne funkcije mogu da omoguće i pronalaženje optimalnog puta između dvije tačke uzimajući u obzir ograničenja prilikom trasiranja, pronalaženje pojava unutar određenog zadatog pojasa, preklapanje i logička podjela na segmente uzimajući u obzir zadate parametre (Slika 4. *Preklapanje i logička podjela dionice puta na segmente uzimajući u obzir parametre*) [1], štampanje kvalitetnih tematskih karata, veze sa relacionim bazama podataka (RDBI – Relational Data Base Interface) (Slika 5. *Veze grafičkih objekata sa relacionim bazama podataka*) [1] što nam omogućava korišćenje prednosti SQL (Structural Query Language) jezika. SQL jezik služi za pretraživanje baza podataka.



Slika 4. Preklapanje i logička podjela dionice puta na segmente uzimajući u obzir parametre



Slika 5. Veze grafičkih objekata sa relacionim bazama podataka

2. KORISTI UVOĐENJA I TROŠKOVI UVOĐENJA GEOGRAFSKOG INFORMACIONOG SISTEMA

2.1. Koristi uvođenja GIS-a

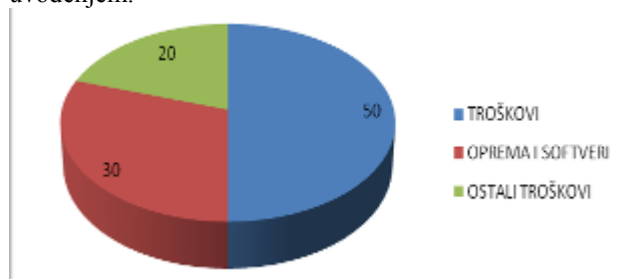
Najvažnija korist uvođenja GIS-a jeste unapređenje upravljanja resursima lokalnih samouprava pa tako i upravljanje putevima. Postiže se efikasnije poslovanje, bolje angažovanje radne snage kao i bolja raspodjela i ušteda finansijskih sredstava. Uvođenje GIS-a treba da pomogne da se objedine tematske karte jednog područja i tako će sve organizacije koje su nadležne za održavanje jednog područja raditi u koordinaciji. Na jednom području svaka popravka izvedena od jednog preduzeća biće vidljiva svakom drugom preduzeću koje treba da radi na tom području. Primjer za to je rekonstrukcija puteva. Pri rekonstrukciji puteva potrebno je poznavati položaj svih podzemnih instalacija kako ne bi došlo do havarije na instalacijama.

2.2. Troškovi uvođenja GIS-a

Uvođenje geografskog informacionog sistema zahtjeva materijalna sredstva koja nisu mala. Sve troškove možemo podijeliti na jednokratne i na godišnje. Jednokratni troškovi su troškovi koji se plaćaju jednom i predstavljaju osnovu za sve ostale procese (primjer. nabavka opreme, prikupljanje podataka, digitalizacija podataka, troškovi konsultanata). Godišnji troškovi se uglavnom odnose na troškove ažuriranja softvera i podataka kao i na troškove obuke kadrova za nove elemente u okviru GIS-a.

Najveći dio troškova su podaci. Obično podaci koštaju preko 50 % ukupnih troškova. Prikupljanje podataka zahtjeva potrebu u ljudstvu i vrijeme koje je potrebno da se prikupe svi podaci. Oprema i softveri koštaju oko 30 % (Slika 6. *Struktura troškova prilikom uvođenja geografskog informacionog sistema*). Najveći dio troškova uvođenja GIS-a je skoncentrisan na samom početku implementacije sistema. Optimizacijom se mogu postići određene uštede. Postupci kojima se mogu postići određene uštede: korišćenjem postojeće IT infrastrukture, korišćenjem radnih stanica i servera, korišćenjem

postojećih podataka i razmjenom podataka sa drugim preduzećima, korišćenjem Open Source softvera, faznim uvođenjem.



Slika 6. Struktura troškova prilikom uvođenja geografskog informacionog sistema

2.2. Problemi uvođenja GIS-a

Osim nedostatka materijalnih sredstava jedan od osnovnih podataka može da predstavlja i nedostatak podataka ili postojanje podataka čija tačnost nije zadovoljavajuća. Zato je potrebno izvršiti detaljnu analizu podataka koje imamo. Obično takvi podaci zahtjevaju doradu kako bi postali upotrebljivi čime gubimo vrijeme koje smo mogli da utrošimo na neki korisniji način. Značajan problem predstavljaju i kadrovi, tj. njihova obučenosť za primjenu elemenata GIS-a.

3. KOMPONENTE GIS-A

Komponente GIS-a su: hardver, softver, korisnici, podaci i procedura (Slika 7. Sastavni dijelovi GIS-a).



Slika 7. Sastavni dijelovi GIS-a

3.1. Hardver

U zavisnosti od postavke GIS-a variraju i potrebe za hardverom. Ako imamo jedno radno mjesto onda će nam biti potreban samo jedan računar a svi podaci će se čuvati na njemu. Ako je potrebno da se podacima pristupa sa više radnih mjesta onda ćemo osim radne stanice podatke skladištiti i na nekom serveru (ovo omogućava koordinaciju rada preduzeća o čemu je ranije bilo riječi).

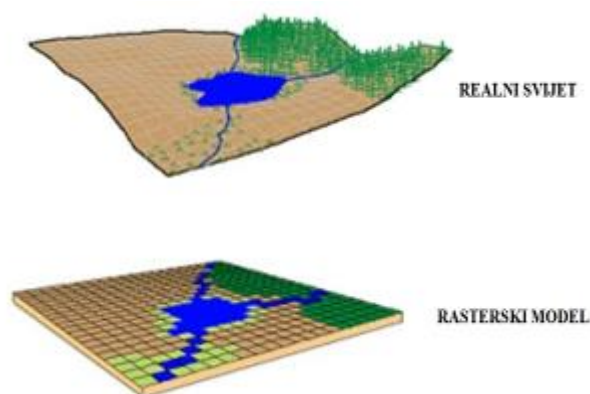
3.2. Softver

Na tržištu danas postoji dosta komercijalnih rješenja od kojih je najpoznatije Esri ArcGIS koji je i najzastupljeniji na našim prostorima. Osim komercijalnih rješenja postoje i besplatni, Open Source, paketi kao što je QGIS koji je i korišćen u primjeru koji je naveden u ovom radu. Prednost komercijalnih softvera je u tome što osim plaćene podrške imamo na raspolaganju i mnogo veći broj funkcija. Mana je cijena, kako početna tako i ona koja se plaća kroz redovno godišnje održavanje softvera. Osim standardnih funkcionalnosti koje omogućuju GIS softveri postoji i mogućnost nadogradnje u vidu dodatnih aplika-

cija koje su napisane specijalno za potrebe nekog korisnika (aplikacije za upravljanje građevinskim zemljištem, aplikacije za upravljanje putevima, aplikacije za saobraćajnu signalizaciju i slično). Današnji trendovi vode ka tome da je moguć pristup sa bilo kog mjesta i sa bilo kog uređaja (Geoportala-namjenske internet aplikacije).

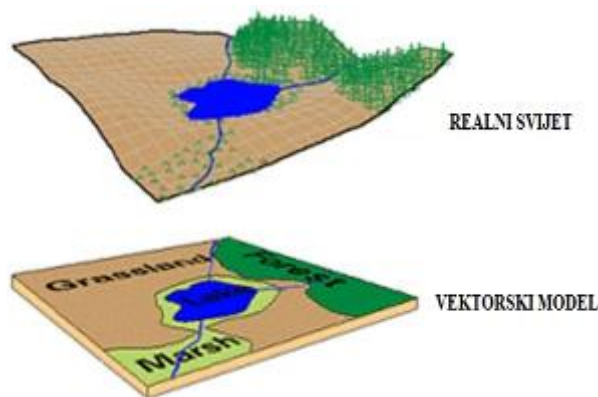
3.3. Podaci

Geografski podaci čine osnovu GIS-a. Kvalitet podataka je od velikog značaja. Kvalitet podataka se određuje na osnovu: tačnosti (pohranjena i stvarna vrijednost parametara, atributa ili pozicije), preciznost (obim detalja koji se mogu vidjeti), ažurnost (upoređuje se stanje u našoj bazi podataka i stanje na terenu), potpunost (količina podataka), konzistentnost (nema protivriječnosti između podataka). Osnovna podjela podataka je na vektorske i rasterske podatke. Rasterski podaci se sastoje od redova i kolona ćelija (piksela) gdje svaka ćelija ima svoju x i y koordinatu i neku vrijednost koja označava njenu boju. Grupisane ćelije predstavljaju sliku. U zavisnosti od rezolucije slike varira i veličina terena koja je obuhvaćena jednom pikselom. Rezolucija 1 cm/piksel znači da svaki piksel na slici predstavlja 1 cm u stvarnosti. Rasterski podaci (Slika 8. Primjer rasterskog modela podataka) koji se najčešće primjenjuju su: ortofot snimak, satelitski snimak ali i razni skenirani materijali.



Slika 8. Primjer rasterskog modela podataka

Vektorski podaci (Slika 9. Primjer vektorskog modela podataka) za prikaz koriste osnovne geometrijske oblike: tačke, linije i poligone.



Slika 9. Primjer vektorskog modela podataka

**PROCENA STANJA I ENERGETSKA SANACIJA STAMBENE ZGRADE U ULICI
ALEKSE ŠANTIĆA U NOVOM SADU****ASSESSMENT AND ENERGY REENEWAL OF RESIDENTIAL BUILDING IN
ALEKSA SANTIC STREET IN NOVI SAD**Slavko Tomić, *Fakultet tehničkih nauka, Novi Sad***Oblast – GRAĐEVINARSTVO**

Kratak sadržaj – Rad obuhvata procenu stanja postojećeg objekta vizuelnim pregledom, kao i proračun energetske efikasnosti. Na osnovu analize uočenih oštećenja i proračuna energetske efikasnosti dat je opis sanacionih mera u cilju povećanja trajnosti i unapređenja energetske efikasnosti objekta.

Abstract – This paper includes assessment of the current state of the building by visual inspection, as well as its calculation of energy efficiency. Based on analysis of observed damage and calculation of energy efficiency a description of the repair measures in order to increase the service life of structures and energy efficiency improvements is given.

Ključne reči: evropski standard, građevinska fizika, gubici toplote, energetska efikasnost, sanacija.

UVOD

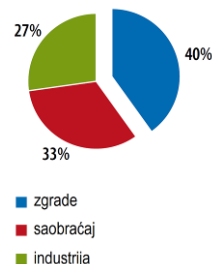
Rad se sastoji iz teorijsko-istraživačkog i praktičnog dela. Teorijski deo je na temu „ETICS evropski standard” gde se govori o evropskim standardima za „ETICS” sisteme. Praktični deo je vezan za procenu stanja i energetske sanaciju stambene zgrade u Ulici Alekse Šantića u Novom Sadu. U sklopu praktičnog dela je urađen elaborat energetske efikasnosti postojećeg stanja objekta. Nakon preduzetih mera za poboljšanje energetske svojstava urađen je i elaborat energetske efikasnosti saniranog objekta.

1. „ETICS” EVROPSKI STANDARD**1.1. Uvod**

Današnje zakonodavstvo se zauzima za bolju energetske budućnost. Cilj je izolovati svaki objekat novogradnje, ali i izvršiti toplotnu sanaciju svakog objekta starogradnje. Značaj toplotne izolacije je o neprocenjive vrednosti u doba klimatskih promena, rastuće potrošnje energije i ograničenih resursa. Efekti termoizolacionih sistema na okolinu i kvalitet života su dalekosežni [1].

Prema evropskim istraživanjima, čak 40% od ukupne energije odlazi na zgradarstvo. Upravo u tom sektoru bi se mogla postići i najveća ušteda energije. Iz tih razloga, Evropski Parlament i Savet, donose 2002. godine Direktivu o energetskim karakteristikama zgrada,

2002/91/EC koja u januaru 2003. godine postaje evropski zakon [2].

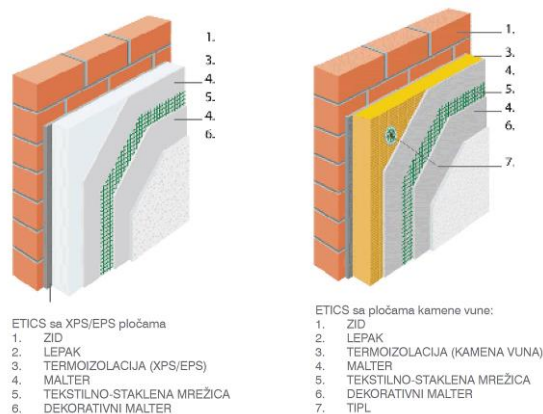


Slika 1. Raspodela potrošene energije [2]

1.2. „ETICS” sistemi

Naziv „ETICS” je skraćenica za složene sisteme spoljne termičke izolacije (External Thermal Insulation Composite Systems). Ovi sistemi su u upotrebi više od pet decenija, rasprostranjeni su širom evrope i mogu da se koriste za poboljšanje energetske efikasnosti novih i postojećih objekata. Spoljašnji toplotno izolacioni sistemi se sastoje od sledećih komponenti: lepak za lepljenje, termoizolacija (EPS ili kamena vuna), tiplovi (ako je potrebno), malter za armiranje, staklena mrežica, završni sloj maltera, profili (ugaoni sa staklenom mrežicom, početni, dilatacioni...).

Tehnologija izvođenja ovih sistema se odvija u tri faze, prvo se vrši lepljenje polistirolnih ploča ili kamene vune na zid odgovarajućim lepkom, pa postavljanje sloja armiranog staklenom mrežicom, preko polistirolnih ploča ili kamene vune, specijalnim fleksibilnim lepkom koji fasadi daje čvrstoću, elastičnost i na kraju postavljanje završnog sloja koji ima dekorativna svojstva, a istovremeno štiti kompletnu termoizolaciju od spoljnjih uticaja [3].



Slika 2. Sastav „ETICS” sistema [3]

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Mirjana Malešev, red.prof.

1.3. Evropski standard (EN 13500:2003) [4]

Pun naziv ovog standarda je: Proizvodi za toplotnu izolaciju zgrada – Spoljašnji toplotno izolacioni kompozitni sistemi (ETICS) na bazi mineralne vune. Ovaj evropski standard je odobren od strane „CEN“ 10. jula 2003. godine. „CEN“ je Evropski komitet za standardizaciju koji okuplja organe nacionalnih standardizacija iz 33 evropske zemlje i jedan od tri evropske organizacije za standardizaciju (pored „CENELEC“ i „ETSI“), koji su zvanično priznata od strane Evropske unije i od strane Evropskog udruženja slobodne trgovine („EFTA“- European Free Trade Association) kao odgovorne za razvoj i definisanje dobrovoljnih standarda na evropskom nivou. „CEN“ daje platformu za razvoj evropskih standarda i drugih tehničkih dokumenata u vezi sa različitim vrstama proizvoda, materijala, usluga i procesa.

EN 13500:2003 sadrži dva aneksa:

- Aneks A (normativ) kontrola fabričke proizvodnje
- Aneks B (informativni) dodatne informacije za klijente i projektante

Standard utvrđuje zahteve za fabrički izrađene proizvode za spoljašnje toplotno izolacione kompozitne sisteme „ETICS“ na bazi mineralne vune, koji se isporučuju u vidu višekomponentnih sistema i koriste za toplotnu izolaciju zgrada.

Oni ne doprinose stabilnosti zida i/ili tavanice na kojoj su postavljeni. Standard obuhvata sisteme kod kojih se zahteva toplotna izolacija materijala za prenošenje opterećenja na podlogu i obuhvata sisteme sa deklarisanom toplotnom otpornošću jednakom ili većom od $1 \text{ m}^2 \times \text{K/W}$. Zahtevi nacionalnih uredbi u pogledu mehaničke nosivosti i stabilnosti „ETICS“ treba da se uzmu u obzir. Ovaj standard ne obuhvata čvrstoću veze između „ETICS“ i površine zgrade za koju se mora pričvrstiti, takođe opisuje karakteristike proizvoda i sadrži postupke ispitivanja, označavanja i etiketiranja.

Po ovom standardu moraju se ispuniti zahtevi koji su sadržani pod tačkama od 1-9 (1.Toplotna otpornost, 2.Mehanička otpornost i stabilnost sistema, 3.Otpornost na požar, 4.Ploče od mineralne vune, 5.Čvrstoća pri zatezanju mreže (ojačanja), 6.Vodopropusnost površine sistema, 7.Otpornost na udar, 8.Otpornost na prodor, 9.Paropropusnost). U tabeli 1. dati su zahtevi za ploče od mineralne vune po ovom standardu.

Tabela 1. Zahtevi za ploče od mineralne vune

Karakteristike	Zahtevi	
	Vrednost	Klasa/nivo/granična vrednost
deklarisana toplotna otpornost	$R_D \geq 1,00 \text{ m}^2 \text{ K/W}$	granična vrednost
zatezna čvrstoća upravna na lice: ploča od MV fiksirana lepkom	$\geq 80 \text{ kPa}$	TR 80
ploča od MV fiksirana spojnicom ^a	$\geq 15 \text{ kPa}$	TR 15
ploča od MV fiksirana sidrom(tiplom) ^a	$\geq 7,5 \text{ kPa}$	TR 7.5
ploča od MV fiksirana sidrom kroz ojačanje ^a	$> 5 \text{ kPa}$	TR 5
dimenzionalna stabilnost	$\leq 1\%$	granična vrednost
tolerancija pravilnosti	$\leq 5 \text{ mm/m}$	granična vrednost
tolerancija ravnosti	$< 6 \text{ mm}$	granična vrednost
tolerancija dužine	$\pm 2\%$	granična vrednost
tolerancija širine	$\pm 1,5\%$	granična vrednost
tolerancija debljine	$+ 3 / - 1 \text{ mm}$	T 5
jačina pritiska	$\geq 10 \text{ kPa}$	CS (10/Y)10
dugoročno upijanje vode delimičnim potapanjem	$< 3 \text{ kg/m}^2$	granična vrednost

^a tretiran kao sistem bez lepka

1.4. Evropski standard (EN 13499: 2003) [5]

Ovaj standard je veoma sličan standardu EN 13500: 2003, sa razlikama u pojedinim vrednostima. Pun naziv ovog standarda je: Proizvodi za toplotnu izolaciju u građevinarstvu – Spoljašnji toplotno izolacioni kompozitni sistemi (ETICS) na osnovu ekspaniranog polistirena. Po ovom standardu se takođe moraju ispuniti zahtevi od 1-9. U tabeli 2. su dati zahtevi za ploče od ekspaniranog polistirena po ovom standardu.

Tabela 2. Zahtevi za ploče od ekspaniranog polistirena

Karakteristike	Zahtevi	
	Vrednost	Klasa/nivo/granična vrednost
deklarisana toplotna otpornost	$R_D \geq 1,00 \text{ m}^2 \text{ K/W}$	granična vrednost
zatezna čvrstoća upravna na lice: ploča od EP fiksirana lepkom	$\geq 100 \text{ kPa}$	TR 100
ploča od EP fiksirana sidrom(tiplom) ^a	$\geq 150 \text{ kPa}$	TR 150
dimenzionalna stabilnost	$\pm 0,2\%$	DS(N)2
tolerancija pravilnosti	$\pm 2 \text{ mm/m}$	S2
tolerancija ravnosti	$\pm 5 \text{ mm}$	P4
tolerancija dužine	$\pm 2 \text{ mm}$	L2
tolerancija širine	$\pm 2 \text{ mm}$	W2
tolerancija debljine	$\pm 1 \text{ mm}$	T2
dugoročno upijanje vode delimičnim potapanjem	$\leq 0,5 \text{ kg/m}^2$	granična vrednost

^a tretiran kao sistem bez lepka

1.5. Zaključak

Cilj evropskih standarda (EN 13500:2003 i EN 13499:2003) je da postave visok kvalitet „ETICS“ sistema uz pristupačne zahteve. Ovi evropski standardi podržavaju zaštitu životne sredine i time „ETICS“ sistemi dobijaju na značaju u savremenom građevinarstvu.

2. PROCENA STANJA STAMBENE ZGRADE U ULICI ALEKSE ŠANTIĆA U NOVOM SADU

2.1. Uvod

Objekat je građen po projektu „Ekonomični stanovi – kula“ tokom 1960-tih godina. Predmetni objekat se nalazi u naselju Grbavica, u Ulici Alekse Šantića br.1, u Novom Sadu. Objekat je stambena zgrada spratnosti Po+P+4 sa ravnim krovom namenjena za kolektivno stanovanje. Zgrada je pravougaone osnove, gabarita $14,72 \times 17,02 \text{ m}$. Bruto površina pod objektom je $250,53 \text{ m}^2$.



Slika 3. Izgled stambene zgrade

Stambena zgrada u podrumu ima 19 ostava, u prizemlju objekta postoje četiri stambene jedinice. Na I, II i III spratu projektovane su po četiri stambene jedinice. Na IV spratu postoje tri stambene jedinice, a pored stanova, ima perionica sa prostorijom za sušenje i velikom terasom. Svaki stan poseduje: predsoblje, kuhinju, spavaću sobu,

dnevnu sobu, ostavu, wc i kupatilo. Vertikalna komunikacija je ostvarena putem jednokrakog stepeništa koje povezuje sve etaže. U stepenišnom prostoru izveden je svetlarnik.

Radi se o zidanoj konstrukciji. Raspored nosećih zidova je simetričan u oba pravca, izvesne nesimetrije ima u podrumu i prizemlju zbog ulaza u objekat. Zidan je od opeke sa armirano-betonskim elementima u centralnom jezgru. Fundiranje objekta je izvršeno na armirano-betonskim trakastim temeljima. Zidovi prizemlja i svih spratova, debljine 38 cm su zidani opekam Mo-110 u krečnom malteru 1:3. Unutrašnji zidovi, debljine 25 cm su zidani šupljom opekam u produžnom cementnom malteru 1:3:9. Objekat ima stubove, horizontalne serklaže, nadvratne i nadprozorne grede. Stepenište, kao i međuspratna ploča u centralnom delu zgrade su monolitni, armiranobetonski, dok je iznad stanova međuspratna konstrukcija polumontažna, sitnorebrasta, tipa „Rapid“ sa armiranobetonskim horizontalnim serklažima, a ona se oslanja se na obimne zidove i zidove jezgra u koje je smešteno stepenište. Međuspratna konstrukcija ispod kupatila je monolitna armiranobetonska spuštana ploča.

2.2. Procena stanja konstrukcije

Vizuelnim pregledom obuhvaćeni su samo dostupni delovi objekta, a to su spoljašnje fasade. S obzirom na to da je objekat u upotrebi i da svaki element ima završnu obradu, vizuelnim pregledom je teško uočiti defekte kao posledicu grešaka prilikom izgradnje objekta. Na fasadi objekta uočena su sledeća oštećenja: vlažne mrlje, tragovi curenja vode, krečnjačke naslage, raslojavanje, pojava biološkog rastinja na dodiru zgrade i tla.



Slika 4. Vlažne mrlje usled slivanja vode



Slika 5. Tragovi curenja vode i raslojavanje završne obrade u gornjim delovima usled vlage i dejstva mraza

Analizom rezultata dobijenih vizuelnim pregledom konstrukcije, došlo se do zaključka da su navedena oštećenja uglavnom posledica prisustva vlage, slivanja vode, dejstva mraza i izloženosti drugim atmosferskim uticajima. Zaključeno je da zgrada nema značajnijih oštećenja, ona su lokalnog karaktera, ne utiču na nosivost, stabilnost, upotrebljivost i trajnost zgrade.

3. PRORAČUN ENERGETSKE EFIKASNOSTI

3.1. Građevinska fizika

Pri proračunu energetske efikasnosti urađen je kompletan proračun toplotne provodljivosti građevinskih elemenata koji čine termički omotač zgrade, proračun difuzije vodene pare, proračun gubitaka i dobitaka toplote, te proračun godišnje potrebne finalne energije za grejanje. Ovim proračunom je zaključeno da je postojeći objekat trenutno energetskog razreda E i da ne zadovoljava energetske zahteve za postojeće objekte prema Pravilniku o energetske efikasnosti zgrada.

3.2. Mere za unapređenje energetske efikasnosti objekta

U cilju poboljšanja energetskih svojstava zgrade uradiće se sledeće:

- postavljanje kamene mineralne vune „FKD-S Thermal“ debljine 10cm sa spoljašnje strane, na svim spoljnim zidovima.
- postavljanje „Ytong Multipor“ ploča debljine 7,5cm na svim zidovima ka negrejanom prostoru.
- postavljanje „Ytong Multipor“ ploča debljine 10cm sa spoljašnje strane (lepljene odozd), na svim međuspratnim konstrukcijama iznad negrejanog prostora.
- postavljanje kamene mineralne vune sa spoljašnje strane međuspratne konstrukcije iznad grejanog prostora, za ravan prohodan krov će se upotrebiti „Smart Roof NORM“, a za ravan neprohodan krov „Smart Roof THERMAL“. U oba slučaja debljina je 24cm.
- zamena postojećih prozora i vrata, šestokomornim PVC profilima sa dvoslojnim staklom ($d=4-15-4\text{mm}$) punjenim argonom.

Tabela 3. - Pregled koeficijenata prolaza toplote kroz termički omotač objekta pre i posle energetske sanacije

	U_p [W/m ² K]	U_s [W/m ² K]	U_{max} [W/m ² K]	p	s
SZ1	1,193	0,270	0,40	ne	da
SZ2	1,575	0,285	0,40	ne	da
SZ3	2,500	0,306	0,40	ne	da
SZ4	1,565	1,152	0,40	ne	ne
SZ5	2,404	0,304	0,40	ne	da
SZ6	1,078	0,383	0,55	ne	da
SZ7	1,815	1,307	0,55	ne	ne
SZ8	1,344	0,412	0,55	ne	da
MK1	1,451	0,343	0,40	ne	da
MK2	1,225	0,329	0,40	ne	da
MK3	1,660	0,354	0,40	ne	da
MK4	2,036	0,142	0,20	ne	da
MK5	0,784	0,138	0,20	ne	da
P1	3,500	1,470	1,50	ne	da
P2	3,500	1,465	1,50	ne	da
P3	3,500	1,467	1,50	ne	da
P4	3,500	1,492	1,50	ne	da
V1	3,500	1,453	1,50	ne	da
V2	5,350	1,426	1,50	ne	da
V3	3,000	3,000	1,60	ne	ne

Legenda:

- Up-koeficijent prolaza toplote pre sanacije
- Us-koeficijent prolaza toplote nakon sanacije
- U_{max} -max dozvoljeni koeficijent prolaza toplote
- p- zadovoljenje uslova pre sanacije
- s- zadovoljenje uslova nakon sanacije
- SZ-spoljašnji zid
- SZ6, SZ7, SZ8- zidovi ka negrejanom prostoru
- MK-međuspratna konstrukcija
- P-prozori
- V-vrata

4. ZAKLJUČAK

Na osnovu sprovedenih proračuna dobijene su vrednosti godišnje potrošnje finalne energije za grejanje, koja su pokazatelj energetskih svojstava zgrade.

Tabela 4. - Pregled potrebne energije za grejanje objekta

	GP [kWh/a]	SGPE [kWh/m ² a]	RGPE [%]	R
p	103740,11	116,47	166,386	E
s	40866,64	45,88	65,543	C

Legenda:

- GP- Godišnja potrebna energija za grejanje
- SGPE- Specifična godišnja potrebna energija
- RGPE- Relativna godišnja potrebna energija
- R- Razred
- p- objekat pre sanacije
- s- objekat nakon sanacije

Energetskom sanacijom je postignuto poboljšanje energetskih svojstava objekta i novim proračunom dobijeno je da zgrada spada u C energetski razred. Time je ispunjen uslov da energetski razred za postojeće zgrade sanacijom mora biti poboljšan za najmanje jedan razred.

5. SANACIONE MERE ZA POBOLJŠANJE TRAJNOSTI KONSTRUKCIJE

Trajnost konstrukcije je u velikoj meri povećana usled poboljšanja energetskih svojstava zgrade. S obzirom na to da je trajnost ugrožena velikim delom kao posledica prisustva vlage, izolacijom spoljnih delova konstrukcije je smanjena vlažnost, a samim tim produžena trajnost zgrade.

Oštećenja: vlažne mrlje, tragovi curenja vode, krečnjačke naslage, raslojavanje, su mala i lokalnog karaktera, ne utiču na nosivost, stabilnost, upotrebljivost i kao takva se saniraju pomoću reparaturnog maltera. Koristiće se neki od mnogobrojnih reparaturnih maltera koji se nalaze na tržištu, oni su mikroarmirani vlaknima ili obogaćeni aditivima.

Biološko rastinje u blagoj pojavi na kontaktu između tla i zgrade, sanira se sredstvom za uklanjanje mahovine, algi i lišajeva, a potom će se naneti vodoodbojan premaz. Mehanička oštećenja fasade će se sanirati pri postavljanju izolacije na spoljašnjim zidovima.

Pri saniranju mehačkog oštećenja trotoara ukloniće se i biološko rastinje. Potrebno je uraditi novi trotoar, osim na jednoj strani gde je već urađen.

6. LITERATURA

- [1] Baunit termoizolacioni fasadni sistemi www.baunit.com
- [2] Brošura „ETICS are amazing“ www.ea-etics.eu
- [3] Ravago Lambda Sr www.ravago.rs
- [4] Evropski standard (EN 13500:2003): *Proizvodi za toplotnu izolaciju zgrada – Spoljašnji toplotno izolacioni kompozitni sistemi (ETICS) na bazi mineralne vune.*
- [5] Evropski standard (EN 13499: 2003): *Proizvodi za toplotnu izolaciju u građevinarstvu – Spoljašnji toplotno izolacioni kompozitni sistemi (ETICS) na osnovu ekspaniranog polistirena.*
- [6] Inženjerska komora Srbije: *Pravilnik o energetskej efikasnosti zgrada („Sl.glasnik RS“ br.61/2011),* Beograd.
- [7] Inženjerska komora Srbije: *Predavanja za obuku o energetskej efikasnosti zgrada,* Beograd, www.ingkomora.org.rs
- [8] www.knaufinsulation.rs
- [9] www.ytong.rs

Kratka biografija:

Slavko Tomić rođen je u Subotici 1985. god. Master rad na Fakultetu tehničkih nauka iz oblasti Građevinarstvo – Procena stanja, održavanje i sanacija građevinskih objekata odbranio je 2016. godine.

**PROCENA STANJA, SANACIJA I POBOLJŠANJE ENERGETSKE EFIKASNOSTI
ZGRADE GIMNAZIJE „SVETOZAR MARKOVIĆ“ U NOVOM SADU****ASSESSMENT, REPAIR AND ENERGY EFFICIENCY IMPROVEMENT OF THE
BUILDING OF HIGH SCHOOL “SVETOZAR MARKOVIĆ” IN NOVI SAD**

Jelena Bačić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAĐEVINARSTVO

Kratak sadržaj – Radom je obuhvaćena detaljna procena stanja i energetske efikasnosti objekta. Nakon analize uočenih oštećenja i rezultata proračuna, dat je predlog mera sanacije koje bi dovele do unapređenja energetske efikasnosti objekta i povećanja trajnosti.

Abstract – This paper includes a detailed assessment of the current state of the building by visual inspection, as well as its calculation of energy efficiency. Based on analysis of observed damage and calculation of energy efficiency a description of the repair measures in order to increase the service life of structures and energy efficiency improvements is given.

Ključne reči: konzervacija, zaštita spomenika kulture, sanacija, energetska efikasnost

1. UVOD

Rad se sastoji od teorijsko-istraživačkog i praktičnog dela. Predmet istraživanja teorijskog dela je Zaštita spomenika kulture, dok je tema praktičnog dela procena stanja i sanacija Gimnazije "Svetozar Marković" u Novom Sadu. U okviru praktičnog dela, urađen je elaborat energetske efikasnosti postojećeg stanja, sa predlogom mera sanacije. Takođe, prikazan je i elaborat energetske efikasnosti saniranog objekta, kao i dalji predlozi za poboljšanje trajnosti i uslova za boravak u objektu.

2. ZAŠTITA SPOMENIKA KULTURE

Konzervacija (restauracija) je delatnost posvećena očuvanju kulturne baštine. Konzervatorske aktivnosti uključuju proveru stanja, izradu dokumentacije, rad na samom predmetu kulturne baštine, te preventivnu zaštitu. Uloga konzervatora je da predmet zaštite održi u stanju što bližem izvornom. Pored toga, konzervator se mora pridržavati jednostavnih etičkih odrednica kao što su: načelo minimalne intervencije, te minimalnog gubitka postojećeg tkiva, korišćenje metoda i materijala usmerenih ka reverzibilnosti (smanjuje probleme kod budućeg tretmana, istraživanja i korišćenja), minimalni gubitak autentičnosti i temeljno dokumentovanje svih preduzetih postupaka.

Prva faza konzervacije je svakako procena neke građevine i njenog istorijskog značaja, njene namene kroz istoriju i ostalih bitnih odlika za razumevanje konstrukcije.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Mirjana Malešev, red. prof.

Sledeća faza podrazumeva precizna merenja objekta, fotogrameriju i stereofotogrameriju, analizu stabilnosti same konstrukcije i mesta na kom se ona nalazi (uticaj tla i vetrova na njenu stabilnost).

Potom, ispituju se sve instalacije objekta, ukoliko će on i dalje biti korišćen.

Svi prikupljeni podaci se pažljivo analiziraju i nakon toga se odlučuje o planu konzervacije u skladu sa sredstvima koja su za to na raspolaganju i određuju se adekvatne tehničke mere zaštite.

Ciljevi tehničke zaštite su očuvanje spomenika kulture putem ojačanja konstrukcije i produženje veka materijala. U zavisnosti od ishoda istraživanja i stepena očuvanosti objekta određuje se i način zaštite:

- konzervacija
- anastiloza
- restauracija
- rekonstrukcija
- premeštanje celih spomenika ili delova
- konsolidacija

Gimnazija "Svetozar Marković" spada u prostornu kulturno-istorijsku celinu, odnosno ona je deo starog jezgra grada Novog Sada.

3. PROCENA STANJA

Od datuma izgradnje do danas, školski objekat u ulici Njegoševa 22 u Novom Sadu, prošao je kroz tri bitne faze u konstruktivnom razvoju. Izgrađen je 1895. godine kao parterna zgrade pravougaone osnove.

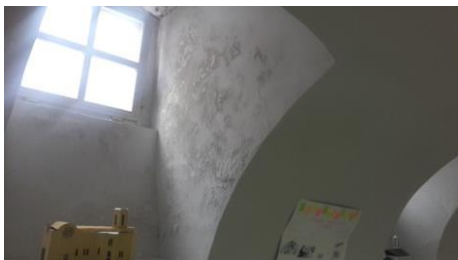
Deset godina kasnije, graditelj Bela Peklo izvršio je rekonstrukciju zgrade, nadograđen je još jedan sprat i dodato dvorišno krilo, čime je dobijena osnovna "na lakat". Konačno, 1993. godine GP "Neimar" iz Novog Sada je po zamisli arhitekta Milorada Milidragovića izvršilo nadogradnju drugog sprata preko cele osnove objekta i izgradilo malu fiskulturnu salu u prizemlju.

Danas imamo objekat koji se prostire na tri etaže, ispod jednog dela osnove nalazi se i suteran. Površina osnove objekta iznosi $1048\text{m}^2 + 114\text{m}^2$.

Objekat je izgrađen u masivnom sistemu. U suteranu su izvedeni masivni svodovi od opeke, dok se u prizemlju objekta nalazi tzv. pruski svod. Međuspratna konstrukcija iznad prvog sprata je sitnorebrasta AB konstrukcija olakšana TM glinenim blokovima, izvedena na već postojećoj tavanicu od poluobljnih greda.

Iznad drugog sprata se nalazi plafonska konstrukcija od gips-kartonskih ploča koja je privrščena za drvene noseće grede. Krovna konstrukcija je čelična, a krovni pokrivač crep.

Na objektu je izvršen detaljan vizuelni pregled sa procenom stanja i nisu uočena značajna oštećenja glavnih konstruktivnih elemenata. Oštećenje karakteristično za stare zidane objekte je svakako vlaga u zidovima, i to pre svega u suterenskoj etaži (Slika 1.), odnosno u prizemnom delu objekta ispod koga ne postoji suteran (Slika 2.).



Slika 1. Vлага u suterenskim prostorijama



Slika 2. Ljuskanje fasade usled prisustva vlage u zidovima

Osim vlage, na fasadi su uočena i druga oštećenja usled mehaničkih udara, atmosferilija i slivanja vode sa nepravilno rešenih konstruktivnih detalja, te su na pojedinim mestima oštećeni fasadni malter, boja kao i fasadni ukrasi (Slike 3 i 4.).



Slika 3. Oštećen fasadni venac

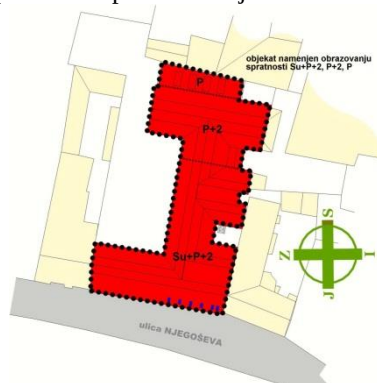


Slika 4. Otpadanje fasadne boje

Vizuelnim pregledom nisu uočeni defekti, kao ni ozbiljnija oštećenja koja mogu da ugroze bilo koji od četiri uslova nosivosti, stabilnosti, trajnosti i funkcionalnosti objekta.

4. PRORAČUN ENERGETSKE EFIKASNOSTI POSTOJEĆEG OBJEKTA

Proračun energetske efikasnosti zgrade izračunava se prema važećem pravilniku o energetske efikasnosti zgrada, Službeni glasnik RS: 061/2011 objavljen 19.08.2011. godine. Na Slici 5. prikazana je situacija zgrade sa prikazom spratnosti objekta.



Slika 5. Situacija zgrade

4.1 Termički omotač zgrade

Elemente termičkog omotača čine:

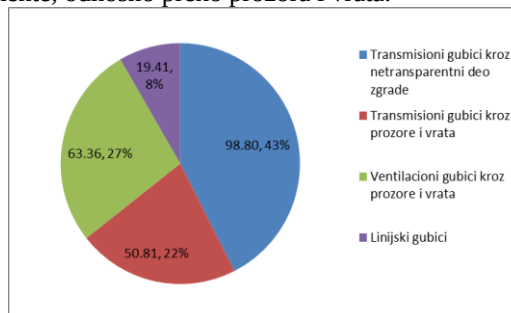
- fasadni zidovi od opeke različite debljine
- zidovi na dilataciji od opeke različite debljine
- ravan i kosi krov iznad grejanog prostora
- međuspratna konstrukcija ispod negrejanog prostora (tavana)
- međuspratna konstrukcija iznad otvorenog prolaza (iznad ajnforta)
- zid u tlu
- pod na tlu (podrum i prizemlje)
- prozori i vrata.

Nakon izvršenog proračuna građevinske fizike na osnovu sastava elemenata termičkog omotača, dobijene su vrednosti provodljivosti građevinskih elemenata i difuzije vodene pare. U Tabeli 1. prikazan je pregled koeficijenata prolaza toplote pre i nakon sanacije, kao i maksimalne dozvoljene vrednosti.

Potom je izvršen proračun gubitaka toplote celog objekta (transmisionih, ventilacionih i linijskih gubitaka), kao i dobitaka (od sunčevog zračenja i unutrašnjih izvora toplote) i proračunata je godišnja potrebna finalna energija za grejanje koja iznosi $Q_{H,nd, interm}=304476,4$ kWh.

Prema Pravilniku o uslovima, sadržaju i postupku izdavanja sertifikata o energetskim svojstvima zgrade, objekat je svrstan u energetski razred D.

Na grafikonu 1. prikazan je procentualni odnos gubitaka toplote postojećeg objekta, gde se može uočiti da se najveći deo gubitaka toplote ostvaruje kroz transparentne elemente, odnosno preko prozora i vrata.



Grafikon 1. Gubici toplote izraženi u [kW] i [%]

5. PRORAČUN ENERGETSKE EFIKASNOSTI SANIRANE ZGRADE

Radi poboljšanja energetske svojstava zgrade izvršena je zamena transparentnih elemenata objekta, odnosno zastakljenja prozora i vrata, potom sanacija međuspratne tavanice prema negrejanom tavanu, i spoljnih fasadnih zidova gde je to moguće.

Predloženo je da se kod prozora "krilo na krilo" izvrši zamena stakla spoljašnjeg krila u dvostruko niskoemisiono 4-12-4mm punjeno ksenonom. Kod jednostrukih drvenih i čeličnih prozora, kao i vrata vrši se zamena zastakljenja istih karakteristika.

Pretpostavljeno je poboljšanje termičkih karakteristika međuspratne konstrukcije prema tavanu KP2, povećavanjem debljine termičke izolacije na ukupnih 15cm, odnosno postavljanjem dodatnog sloja staklene vune od 10cm.

Takođe, predlaže se termoizolacija fasadnog zida drugog sprata (dvorišna fasada), oblaganjem zidova slojem kamene mineralne vune od 12cm.

Preduzete sanacione mere dovele su do željenih rezultata, odnosno popravile energetske razrede zgrade. Po ponovljenom proračunu godišnja potrošnja finalne energije za grejanje iznosi $Q_{H,nd, interm}=223808,4$ kWh, što prema Pravilniku o uslovima, sadržaju i postupku izdavanja sertifikata o energetskim svojstvima zgrade znači da zgrada namenjena obrazovanju i kulturi spada u energetske razrede C. U Tabeli 3. prikazan je pregled potrošnje finalne energije za grejanje objekta pre i posle sanacije, a u Tabeli 2. priložena je klasifikacija zgrada namenjenih obrazovanju i kulturi na energetske razrede.

Tabela 1. Pregled koeficijenata prolaza toplote

	Up [W/m ² K]	Us [W/m ² K]	U _{max} [W/m ² K]	Pre san	San
ZF1	1,235	1,235	0,4	NE	NE
ZF2	1,647	1,647	0,4	NE	NE
ZF3-1	0,965	0,23	0,4	NE	DA
ZF3-2	0,965	0,965	0,4	NE	NE
ZF4	0,69	0,69	0,4	NE	NE
ZF5	0,807	0,807	0,4	NE	NE
ZF6	1,061	1,061	0,4	NE	NE
ZP1	0,693	0,693	0,4	NE	NE
ZD1	1,177	1,177	0,5	NE	NE
ZD2	1,547	1,547	0,5	NE	NE
ZD3	0,930	0,930	0,5	NE	NE
ZD4	0,675	0,675	0,5	NE	NE
ZD5	0,782	0,782	0,5	NE	NE
KRF	0,355	0,355	0,2	NE	NE
KRS	0,36	0,36	0,2	NE	NE
KP1	0,64	0,64	0,55	NE	NE
KP2	0,637	0,25	0,4	NE	DA
TEA	0,287	0,287	0,3	DA	DA
PKP	2,72	1,37	1,5	NE	DA
P01	3,50	1,20	1,5	NE	DA
P02	3,50	1,16	1,5	NE	DA
P03	3,50	1,14	1,5	NE	DA
P04	3,50	1,21	1,5	NE	DA
P05	3,50	1,14	1,5	NE	DA
P06	5,00	1,35	1,5	NE	DA
P07	3,56	1,25	1,5	NE	DA
P08	3,48	1,16	1,5	NE	DA
P09	5,00	1,25	1,5	NE	DA
P10	5,00	1,35	1,5	NE	DA
P11	5,00	1,32	1,5	NE	DA
P12	5,00	1,22	1,5	NE	DA
V1	5,00	1,11	1,6	NE	DA

V2	5,00	1,11	1,6	NE	DA
V3	5,00	1,11	1,6	NE	DA
V4	5,00	1,11	1,6	NE	DA
V5	4,00	1,07	1,6	NE	DA

Legenda:

- Up-koeficijent prolaza toplote pre sanacije
- Us-koeficijent prolaza toplote nakon sanacije
- U_{max}-max dozvoljeni koeficijent prolaza toplote
- p- zadovoljenje uslova pre sanacije
- s- zadovoljenje uslova nakon sanacije

Tabela 2. Godišnja potrebna energija i energetske razrede zgrade

Zgrada namenjena obrazovanju i kulturi	Nove	postojeće
Energetski razred	$Q_{H,nd,rel}$ [%]	$Q_{H,nd}$ [kWh/(m ² a)]
A+	≤ 15	≤ 12
A	≤ 25	≤ 20
B	≤ 50	≤ 38
C	≤ 100	≤ 75
D	≤ 150	≤ 113
E	≤ 200	≤ 150
F	≤ 250	≤ 188
G	> 250	> 188

Tabela 3. Pregled godišnje potrošnje finalne energije za grejanje objekta pre i posle sanacije

	pre	san	
$Q_{H,nd}=$	304476	223808	kWh/a
$q_{H,nd}=$	101,73	74,8	kWh/m ² a
$Q_{H,nd,rel}=$	135,64	99,73	%
Razred:	D	C	

5.1 Dodatne mere sanacije za poboljšanje energetske svojstava objekta

Sanacija kosog krova

Kada je reč o kosom ventilisanom krovu, predlažemo rešenje izolacije sa unutrašnje strane koje nudi proizvođač izolacionih materijala i sistema Knauf.

LDS sistem je energetske efikasan sistem koji omogućava odličnu toplotnu zaštitu, zaptivenost i zaštitu od uticaja vetra, visok stepen otpornosti na požar i dobru zvučnu izolaciju.

Sanacija spoljnih zidova sa unutrašnje strane

Izolaciju radimo sa unutrašnje strane samo kada je to jedina opcija, odnosno kada je postavljanje izolacije sa spoljne strane zida nemoguće ili je izuzetno otežana ugradnja. Pri sanaciji i rekonstrukciji spomenika kulture kod kojih je zaštićena fasada objekta, odnosno kod kojih je fasada oblikovno zahtevna, često je jedino rešenje za izolaciju zidova, njihovo oblaganje sa unutrašnje strane.

Multipor termoizolacione ploče, proizvođača Ytong, su se pokazale kao veoma dobro rešenje kod svih vrsta masivnih zidova bez parne brane. Multipor je mineralna termoizolaciona ploča proizvedena sa ciljem bolje izolacije građevina, sastoji se od: kreča, peska, cementa i vode.

Kako samo ime govori, ovaj materijal sadrži veliki broj vazdušnih pora, koje obezbeđuju dobra termoizolaciona svojstva.

Toplotne pumpe za uštedu energije (voda-voda)

Solarne toplotne pumpe koriste prirodnu toplotnu energiju iz zemlje, podzemnih voda ili vazduha i doprinose značajnoj uštedi na troškovima grejanja. Toplotne pumpe ne zagađuju okolinu, efikasne su čak i na niskim temperaturama, mogu pokriti čak 80% energetske potrebe zgrade, koristeći obnovljivu energiju iz okoline. Tek 20% energije treba dodati u vidu električne energije za rad same pumpe. Leti se pumpa može koristiti za hlađenje.

Toplotne pumpe voda/voda koriste toplotu podzemne vode. Usisna pumpa dovodi podzemnu vodu do toplotne pumpe koja akumuliranu energiju iz podzemne vode pretvara u toplotnu energiju za sistem grejanja i pripremu potrošne tople vode.

Oхлаđena podzemna voda se potom vraća u apsorpcioni bunar i na taj način se vraća u prirodni podzemni tok. Dakle, kod ovog sistema su potrebna dva bunara – jedan iz kojeg se crpi voda, koja protiče preko izmenjivača toplote i pri tome joj se oduzima toplota, a drugi je ponorni bunar u koji se ohlađena voda vraća.

6. SANACIONE MERE ZA POBOLJŠANJE TRAJNOSTI KONSTRUKCIJE

Osim sanacije objekta radi poboljšanja energetske svojstava zgrade, potrebno je izvršiti i druge sanacione mere koje će popraviti oštećenja uočena na objektu.

6.1 Sanacija vlage u spoljašnjim zidovima

Horizontalna hidroizolacija presecanjem zidova

HIO-tehnologija je metoda trajne zaštite od kapilarne vlage, koja potpunim presecanjem zidova ne ugrožava njihovu stabilnost. Ova metoda nema ograničenja za debljinu zida u pogledu mogućnosti presecanja i dužine šina. Stabilnost zida nije ugrožena ni u jednoj fazi izvođenja radova niti postoji opasnost od sleganja objekta. Presecanjem zidova se formira trajni prekid kapilarnog vlaženja zida.

Vertikalna hidroizolacija sa spoljašnje strane

Izrada vertikalne hidroizolacije podrazumeva otkopavanje spoljašnjeg podrumskog zida širokim iskopom, lepljenje bitumenskih traka i obezbeđenje hidroizolacije podrumskog zida zidanjem dodatnog zida od opeke 12cm. Nakon toga preostaje nasipanje zemlje u široki iskop i ponovno popločavanje dvorišta.

Vertikalna hidroizolacija sa unutrašnje strane-penetracija

Poboljšanje vodonepropustljivosti zidova podrumskih delova zidova uz susedni objekat (zid na dilataciji) i ka uličnoj strani može se raditi penetratima i samoisušivim - parodifuznim malterima.

6.2 Sanacija fasade

Pre bojenja fasade, vrši se priprema podloge. Na mestima gde se fasadna boja ljušti i otpada, potrebno je ukloniti labave delove struganjem ili pranjem. Deonice na kojima je oštećen fasadni malter, najpre oštemati do pravilnog geometrijskog oblika i potom ispuniti produžnim malterom i na taj način izravnati površine fasadnih zidova. Nakupljenu prljavštinu i prašinu najpre treba ošmirglati, a potom i prebrisati vodom i pustiti da se osuši. Potrebno je uzeti otiske za sve malterske profile na fasadi. Zamena dekorativnih elemenata se poverava konzervatoru, odnosno vajar. Predlaže se bojenje obnavljajućim bojama za mineralne podloge i objekte kulturne baštine, i to 1x temeljni premaz silikatnog prajmera i 1-2 x silikatne fasadne boje.

7. ZAKLJUČAK

Na osnovi sprovedenih proračuna dobijene su vrednosti godišnje potrošnje finalne energije za grejanje, koja su pokazatelj energetske svojstava zgrade. Odgovarajućim sanacionim merama postignuto je poboljšanje energetske svojstava objekta, te ispunjen uslov da energetski razred za postojeće zgrade sanacijom mora biti poboljšan za najmanje jedan razred. Osim toga dati su i predlozi za dalje poboljšanje energetske svojstava zgrade, kao i sanacione mere za poboljšanje trajnosti konstrukcije.

8. LITERATURA

- [1] Svetislav Vučenić: *Urbana i arhitektonska konzervacija*, Beograd 2004.
- [2] Inženjerska komora Srbije: *Pravilnik o energetske efikasnosti zgrada*, "Sl. glasnik RS", br. 61/2011, Beograd.
- [3] Inženjerska komora Srbije: *Predavanja za obuku o energetske efikasnosti zgrada*, Beograd.
- [4] M. Radeka: *Materijal sa predavanja- Materijali u građevinarstvu I*, Departman za građevinarstvo, Novi Sad.
- [5] M. Malešev i V. Radonjanin: *Materijal sa predavanja- Materijali u građevinarstvu II*, Departman za građevinarstvo, Novi Sad.

Kratka biografija:



Jelena Bačić rođena je u Novom Sadu 1990. godine. Diplomski - master rad na Fakultetu tehničkih nauka iz oblasti Građevinarstvo – Procena stanja, održavanje i sanacija građevinskih objekata odbranila je 2016. godine.

HIDRAULIČKA ANALIZA I FUNKCIONISANJE VAKUUMSKE KANALIZACIJE NA PRIMERU NASELJA GOSPODINCI**HYDRAULIC ANALYSIS AND FUNCTIONING OF VACUUM SEWAGE IN THE EVENT OF THE VILLAGE GOSPODINCI**

Miloš Tišma, Matija Stipić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAĐEVINARSTVO

Kratak sadržaj – U master radu je urađena hidraulička analiza i mogućnost funkcionisanja vakuumske kanalizacije na primeru naselja Gospodinci. Prilikom razmatranja korišćen je Standard DWA-A 116-1E koji se upravo i odnosi na vakuumski kanizacioni sistem. Sve definicije, formule i pravila za proračun vakuumske kanalizacije u ovom master radu su usvojene iz navedenog Standarda. Vakuumski sistem kanisanja se u poslednjih nekoliko godina pojavljuje na području Vojvodine kao alternativa dosadašnjem gravitacionom sistemu. Razlozi za ovo su specifičnosti koje se ogledaju u poskupljenju izvođenja i eksplataisanja gravitacionog sistema.

Abstract – The master work is performed hydraulic analysis and the possibility of functioning of the vacuum sewerage in the case of the village Gospodinci. In considering is used the Standard DWA-A 116-1, which is precisely relates to a vacuum sewage system. All definitions, formulas and rules for the calculation of the vacuum sewer in this master work are adopted from the above Standard. The vacuum system of channeling in the last few years appears in Vojvodina as an alternative background gravitational system. The reasons for this are the specifics of which are reflected in increased prices of construction and exploitation gravitational system.

Ključne reči: Vakuum, kanalizacija, hidraulika, crpna stanica, Standard DWA-A 116-1E.

1. UVOD

Zahtevi za potpuno zatvorenim sistemom kanisanja, racionalizacijom količine ugrađenog materijala, smanjenjem potrebnog rada pri izvođenju kanizacione mreže, naponi za smanjivanje nepovoljnih uticaja na životnu sredinu pri građenju i pri eksploataciji, bili su glavni razlozi za primenjivanje novih kanizacionih sistema.

U okviru sistema odvođenja otpadnih voda, sistem vakuumske kanalizacije predstavlja jedno od mogućih tehničkih rešenja. Teritorija Vojvodine, zbog svojih geografskih karakteristika, pogodna je za primenu ove vrste odvodnje upotrebljene otpadne vode.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Matija Stipić, docent.

2. METODOLOGIJA

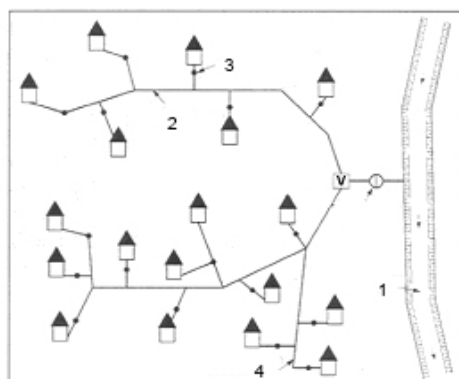
Gospodinci su naselje u Srbiji u opštini Žabalj. Nalaze se u Vojvodini, u južnoj Bačkoj. Od Žablja su udaljeni 8 km. Prema poslednjem popisu iz 2011. godine naselje ima 3715 stanovnika. U naselju ima 1293 domaćinstva, a prosečan broj članova po domaćinstvu je 2,8. Gospodinci su danas tipično ravničarsko, panonsko selo. Kuće su ušorene, okrenute prema ulici sa prostranim ekonomskim dvorištima i baštama. Povezano je sa asfaltnim putem kao i železničkom prugom sa svim većim privrednim i kulturnim centrima.

3. VAKUUMSKI KANALIZACIONI SISTEMI

Sistem se primenjuje u slučajevima kao što su:

- Potpuno ravan teren, naročito ako je visok nivo podzemnih voda,
- Nepovoljan uzdužni profil sa izraženim usponima i padovima,
- Nestabilno tlo,
- Male gustine naseljenosti i veliki koeficijenti neravnomernosti potrošnje vode, odnosno neravnomerno tečenje u kanalizaciji,
- Potreba zaštite terena od mogućeg zagađenja kanalskim sadržajem, potreba za visokim stepenom ekološke zaštite,
- Potreba za savlađivanjem prepreka,
- Oblasti zaštite voda.

Treba napomenuti da je vakuumski kanizacioni sistem, pod određenim uslovima znatno jeftiniji od gravitacionog sistema. Svi troškovi amortizacije, kamata, rad i održavanje moraju biti uključeni u troškove poređenja. Svi na navedeni razlozi zbog kojih je vakuumski kanizacioni sistem u prednosti u odnosu na ostale sisteme se moraju razmatrati prilikom izrade idejnog projekta što je u ovom master radu urađeno.

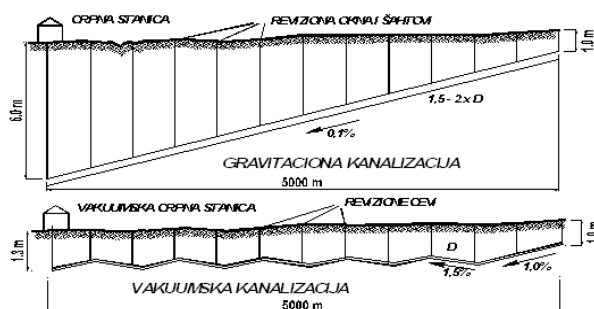


Slika 1. Šema vakuumske kanalizacije

Glavni delovi kanalizacionog sistema pod vakuumom su:

- Gravitacioni kućni priključci, rade kao klasična kanalizacija,
- Priključne jedinice sa interfejs ventilima, su posebna okna u koja se sliva voda iz pojedinih kuća, a odatle se, dejstvom interfejs ventila, voda prebacuje u sistem cevi pod vakuumom,
- Vakuumski glavni razvod (PVC ili PE cevi) funkcioniše pod vakuumom od približno 0,6 bara i služi za transport kanalskog sadržaja od sabirnih jedinica do sabirne stanice,

Sabirna ili pumpna stanica se nalazi na kraju vakuumskog kolektora i skuplja kanalski sadržaj iz celog sistema u sabirni rezervoar. Iz sabirnog rezervoara vodu dalje potiskuju pumpe za vodu. Važno je napomenuti da je samo u sabirnoj stanici potrebna električna energija.

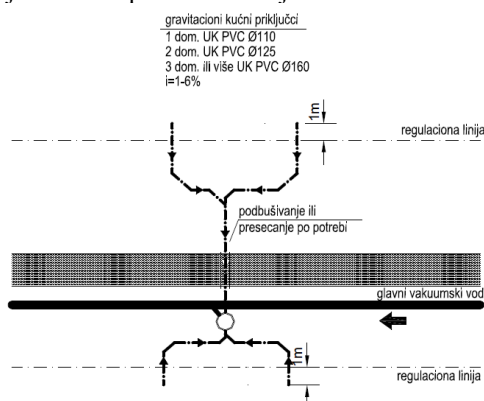


Slika 2. Podužni profil vakuumske i gravitacione kanalizacije

3.1 Princip funkcionisanja sistema

Po Standardu DWA-A 116-1E koji se primenjuje ovom prilikom, dužina vakuum cevovoda je do 4km.

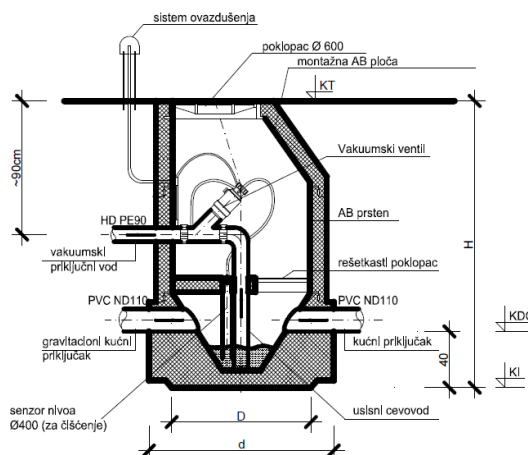
Da bi se ostvarilo kretanje fluida u sistema mi unosimo energiju kojom primoravamo taj isti fluid da menja svoj položaj i kreće se prema sabirnoj stanici.



Slika 3. Tipiko rešenje spajanja vakuumske kanalizacije sa gravitacionom

Vakuumski sistem kanalizacije se formira od potpuno zatvorenog sistema cevovoda, u kojem se konstantno održava podpritisak. Stvaranje vakuma se vrši u vakumskoj ili sabirnoj stanici. Korisnici vakuumske kanalizacije se priključuju u sabirnu šahtu. Ovo se ostvaruje gravitacionom kanalizacijom. Sabirne šahte moraju biti smeštene na javnoj površini kako bi se

obebedio pristup u slučaju potrebe čišćenja ili opsluživanja iste. U sabirnoj šahti se nalazi interfejs ventil koji se na osnovu povišenog nivoa upotrebljene vode u sabirnoj šahti otvara i propušta određenu količinu tečnosti zajedno sa vazduhom. Ventil se otvara na račun razlike pritiska u vakuumskom sistemu i sabirnoj šahti.



Slika 4. Tipiko rešenje sabirne šahte

Vakuumski kanalizacioni sistemi funkcionišu po sledećem principu:

- Upotrebljena voda od svakog korisnika otiče gravitaciono, kroz gravitacione priključne jedinice puneći rezervoar priključne jedinice do određenog nivoa.
- Porastom vode do određenog nivoa u sabirnoj šahti, vazduh potiskivan porastom nivoa vode sabija se u indikatorskoj cevi.
- Sledeći korak je otvaranje ventila sa tajmerom čiji jedan ciklus traje od 3 do 30s. Svrha podešavanja dužine ciklusa je omogućavanje ulaksa i vazduha u sistem zajedno sa vodom u potrebnom odnosu.
- Voda pomešana sa vazduhom se u cevima kanalizacionog sistema kreće tangencionalnom brzinom od 4,5 do 6,5 m/s.
- Prethodno opisan princip se ponavlja pri sledećem punjenju tečnosti u sabirnoj šahti.
- Kada fluid stigne u vakuumsku sabirnu stanicu on ulazi u sabirni rezervoar odakle se radom pumpe za vodu transportuje u novo postrojenje, postrojenje za prečišćavanje otpadne vode.
- U sabirnoj vakuumskoj stanici su smeštene vakuumske pumpe koje omogućavaju stvaranje vakuma u sistemu.

3.2 Formiranje mreže

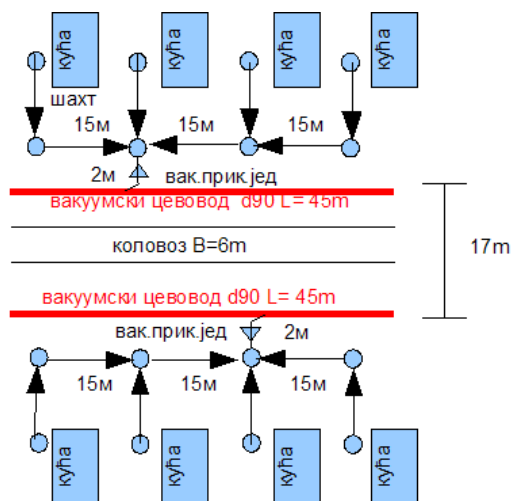
Sistem vakuumske kanalizacije naselja Gospodinci se sastoji od tri nezavisna vakuumska kolektora sa zajedničkom crpnom stanicom. Prilikom odabira vakuumskih kolektora obratila se pažnja da stanovništvo u naselju bude ravnomerno raspoređeno i tako smo dobili kolektor odnosno vakuumsku granu 1-13, vakuumsku granu I-IX i vakuumsku granu A-M. Grana 1-13 je ukupne dužine zajedno sa glavnim i priključnim vakuumskim vodovima 5147m.

Grana I-IX je ukupne dužine 5119m.

Grana A-M je ukupne dužine 6488m.

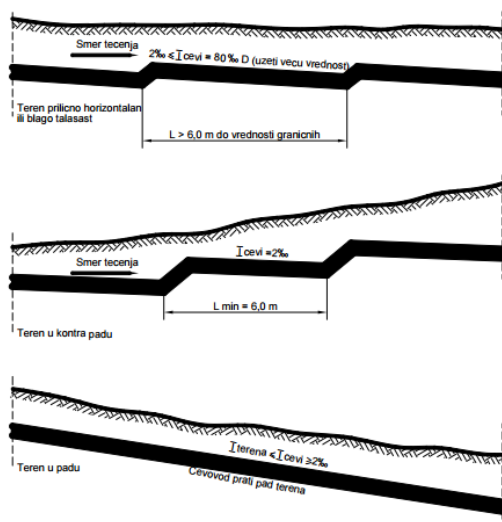
Stanovništvo je podeljeno po granama na sledeći način:

1. Grana 1-13 sa ukupno stanovništva 1128.
2. Grana I-IX sa ukupno stanovništva 1209.
3. Grana A-M sa ukupno stanovništva 1608.



Slika 5. Vakuumska kanalizacija sa obe strane ulice

Zbog cene vakuumske priključne jedinice I ukupnog broja od oko 1200 gravitacionih priključaka, prva varijanta ima manju cenu u odnosu na drugu. U praksi će rešenje uvek biti između ova dva s obzirom na određeni broj kuća duž svake ulice. Usvajeno je da se zbog veće pouzdanosti rada ide na minimalni priključak od dve kuće na jednu jedinicu koji se u pojedinim slučajevima može povećati do maksimalno 4 priključka na jednu jedinicu.



Slika 6. Šema podužnog profila u funkciji pada terena

Vakuumska mreža može da se izvede od cevi koje su izrađene od različiti materijala. Ovom prilikom su primenjene HDPE cevi koje su međusobno spojene lepljenjem odnosno varenjem cevi.

3.3 Vakuumska pumpna stanica

Najvažniji deo sistema vakuumske kanalizacije je vakuumska crpna stanica. Formiranje vakuuma, sakupljanje otpadne vode, dalji transport, upravljanje, kontrola rada

sistema, merenje kapaciteta se dešava u okviru ovog objekta.

Uloga vakuumske pumpne stanice jeste da radom vakuumskih pumpi održava potreban nivo vakuuma u sabirnom kanalizacionom sistemu. Skupljena upotrebljena voda iz naselja Gospodinci se vakuumskim kanalizacionim vodovima dovodi do vakuumskih sudova u koje se izliva. Iz vakuumskih sudova se sakupljena voda preko uronjenih kanalizacionih pumpi potiskuje ka planiranom postrojenju za prečišćavanje upotrebljenih voda.

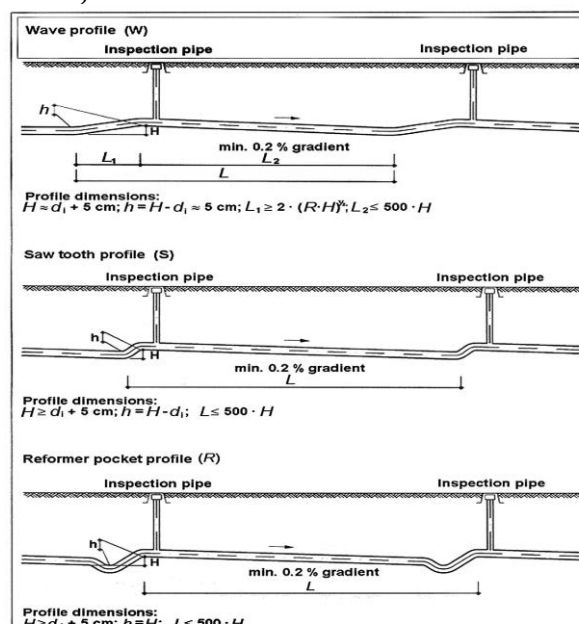
Po Standard DWA-A 116-1E vakuum pumpe održavaju u jednom ili više rezervoara vakuum stanice vakuum pritisak od 60 do 70 kPa (40 do 30 kPa apsolutnog pritiska), ili 0.6 do 0.7 bar, što je obično potrebno za rad vakuum sistema. Otpadne vode se dostavljaju od vakuum stanica do postrojenja za prečišćavanje preko pumpi ili pneumatskim transportom (sudovi pod pritiskom).

Oprema vakuumske sabirne stanice se sastoji od:

- Rezervoara,
- Vakuumske pumpe,
- Pumpe za vodu,
- Kontrolnih uređaja,
- Potrebne armature.

4. PROJEKTNII ZAHTEVI

Veličina nominalnog dijametra vakuum kanalizacije je rezultat hidrauličkog proračuna, Vakuum cevovodi od PVC moraju biti SDR 21. Za PVC-U cevi pod pritiskom DIN 8061, DIN 8062 i DVGW se primenjuje 320. Koeficijent toplotnog širenja od 0,08 mm/m*K se mora poštovati. Vakuum cevovodi od PE moraju biti SDR 11. Za HDPE cevi pod pritiskom DIN 8075, EN 12201 i DVGW se primenjuje. Toplotni koeficijent širenja 0,2mm/m*K mora da se poštuje. Elektrozaravane spojnice se mogu koristiti. Elektrozarivanje može da vrši samo obučena osoba. U odstupanju od EN 1091 sve cevi i armature u vakuum cevovodu moraju imati nominalni pritisak rangiran prema DIN 2401-1 od najmanje 1,0 MPa (10 bara).



Slika 7. Tipovi podužnog profila

5. HIDRAULIČKI PRORAČUN

Maksimalna visinska razlika se izračunava na osnovu konzervativne pretpostavke da su sva uzvišenja ispunjena tečnošću.

Maksimalna statička visinska razlika je visina H (visinska razlika između visoke tačke i prethodne niske tačke) smanjena za unutrašnji prečnik cevi.

Zbir maksimalnih statičkih visina duž vakuum glavne grane ne sme da prelazi 4 do 5 m.

Kod većih statičkih visina zahteva se montaža automatskih ventila za upuštanje vazduha u vakuum cevovod kada vakuum pritisak padne ispod minimuma.

Tabela1: Srednji odnos vazduh/voda

Dužina vakuum grane	Dužina-specifična gustina naseljenosti	
	0.05l/m	0.11/m
	0.21/m	0.51/m
	Srednji odnos vazduh/voda	
500m	3.5-7	3-6
	2.5-5	2-5
1000m	4-8	3.5-7
	3-6	2.5-5
1500m	5-9	4-8
	3.5-7	3-6
2000m	6-10	5-9
	4-8	3.5-7
3000m	7-12	6-10
	5-9	4-8
4000m	8-15	7-12
	6-10	(5-9)*
*Preporučuje se samo za specijalne slučajeve		

Obično se pretpostavlja maksimalan protok od 0,005 l/(I*s). Otpadne vode iz industrija su pretvorene u ekvivalentne stanovnike pri cemu ekvivalentna populacija odgovara protoku od 150l/d.

U posebnim slučajevima se projektovani maksimalni protok pretvara u 0,005 l/s.

Visinski profili i promene smera cevovoda se mogu raditi savijanjem ako se pažljivo drži dozvoljenih radijusa. Minimalni radijus savijanja PVC cevi je $R > 300 \cdot d_e$, a HDPE cevi je $R > 50 \cdot d_e$.

Uputstva proizvođača se moraju poštovati. Veze moraju biti prema uputstvu proizvođača.

Veze moraju biti urađene po preporuci proizvođača.

6. REZULTATI PRORAČUNA

6.1 Proračun profila kanalizacije

Usvojeni prečnici vakuumske kanalizacije se kreću od DN65 do DN250.

Maksimalna statička razlika pritiska za granu I-IX iznosi 1.40m, za granu I-IX 1.35m, a za granu A-M 1.43m.

Rezultati zadovoljavaju uslov hidrauličke uslove vakuumskog sistema.

6.2 Pumpna stanica

Izabrani broj vakum pumpi i pumpi za otpadnu vodu:

nA=	6	
QA,p,s=	400	m3/h
QA,p=	160	m3/h
nww=	4	
Qww,p=	10	l/s
Potrebna zapremina rezervoara:		
V=	$V_w + V_A - V_{ww}$	
V=	$0,75 + 5,55 - 0 =$	6,3 m3

$$V \geq 3 \cdot V_w$$

$$V \geq 3 \cdot 0,75 = 2,3 \text{ m3}$$

Usvaja se rezervoar zapremine 7m3

Potrošnja energije po broju stanovnika u godini iznosi:

$$W(PT) = 11.5 \text{ kWh}/(I \cdot a)$$

7. ZAKLJUČAK

Primena vakuumskog sistema ulične kanalizacije na području Vojvodine, a samim tim i u naselju Gospodinci je moguća.

Reljefni izgled naselja, vladajući uslovi terena i relativno visok nivo podzemne vode, dovode do kranjeg zaključka, da je primena ovog sistema na datom području ima smisla. Pored navedenog svakako treba zaključiti da je dobro rešavanje sistema kanalisanja moguće nakon analize više varijantnih rešenja, uključujući i razna raspoloživa tehnološka rešenja.

Područja u kom se primenjuju sistemi vakuumske kanalizacije karakteriše mala gustoća stanovništva, poprilično ravan teren i visoke podzemne vode. Sve nabrojane stavke su karakteristične i za naselje koje je obrađeno u ovom master radu.

Kratka biografija:



Miloš Tišma rođen je u Prijedoru 1990. god. Master rad na Fakultetu tehničkih nauka iz oblasti Građevinarstvo – Hidrotehnika odbranio je 2016.godine.



Matija Stipić rođen je u Somboru 1964. godine. Doktorirao je na Fakultetu tehničkih nauka 2009. god., a od 2011. ima zvanje docenta. Oblast interesovanja su mu hidraulika i komunalna hidrotehnika.

Aleksandar Dušanić, *Fakultet tehničkih nauka, Novi Sad*

2632

3. ANALIZA OPTEREĆENJA

Analiza opterećenja urađena je prema evropskim standardima [4], [5] i [7], a obuhvata stalno, korisno, opterećenje vetrom, opterećenje snegom i seizmičko dejstvo.

3.1 Stalno opterećenje

Stalna opterećenja su ona opterećenja koja deluju na konstrukciju tokom celog eksploatacionog veka konstrukcije. U stalno opterećenje ulazi sopstvena težina svih elemenata konstrukcije, slojeva podova, plafona i zidova. Stalno opterećenje je naneto kao površinsko opterećenje na međuspratne tavanice, dok je opterećenje od zidova naneto kao linijsko opterećenje. Intenzitet opterećenja tavanica jednak je i iznosi $1,50 \text{ kN/m}^2$ za sve prostorije osim lođa i zajedničke ostave gde je predviđeno opterećenje od $2,0 \text{ kN/m}^2$. Ravni neprohodni krov opterećen je sa $8,50 \text{ kN/m}^2$, dok je ravni prohodni krov opterećen sa $6,50 \text{ kN/m}^2$. Stepenišna ploča je opterećena sa $1,50 \text{ kN/m}^2$. Linijsko opterećenje od zidova iznosi $8,0 \text{ kN/m}^2$ za fasadni zid, $5,7 \text{ kN/m}^2$ za zidove između stanova i $7,4 \text{ kN/m}^2$ za unutrašnje pregradne zidove.

3.2 Korisno opterećenje

Prema preporukama iz standarda [5], korisna opterećenja za sve prostorije iznose $2,0 \text{ kN/m}^2$, osim za zajedničku ostavu kada iznosi $4,0 \text{ kN/m}^2$. Stepenišna ploča opterećena je korisnim opterećenjem od $3,0 \text{ kN/m}^2$, prohodni krov sa $2,0 \text{ kN/m}^2$, a neprohodni krov sa $0,50 \text{ kN/m}^2$.

3.3 Opterećenje snegom

Opterećenje snegom se usvaja intenziteta $1,0 \text{ kN/m}^2$ po površini krovne ploče.

3.4 Opterećenje vetrom

Opterećenje vjetrom je određeno prema standardu [5]. Modelirana su 4 slučaja opterećenja vetrom, svaki slučaj odgovara jednom od 4 pravca duvanja vetra koji su upravni na fasadu objekta. Prilikom modeliranja, određeno je kao površinsko pa je pretvoreno u linijsko i raspoređeno da deluje na grede i stubove ramova na fasadi.

3.5 Seizmičko opterećenje

Seizmičko dejstvo je određeno prema [5]. Objekat se nalazi u VIII seizmičkoj zoni, a projektovan je za srednju klasu duktilnosti – DCM. Usvojen je tip tla C. Gornja vrednost faktora ponašanja q koji služi za ocenu kapaciteta disipacije energije, mora se odrediti za svaki proračunski pravac prema izrazu (1):

$$q = q_0 * q_w \geq 1.5 \quad (1)$$

gde je:

q_0 – osnovna vrednost faktora ponašanja, zavisna od tipa konstrukcijskog sistema

q_w – faktor koji uzima u obzir preovlađujuću vrstu loma konstrukcijskih sistema sa zidovima

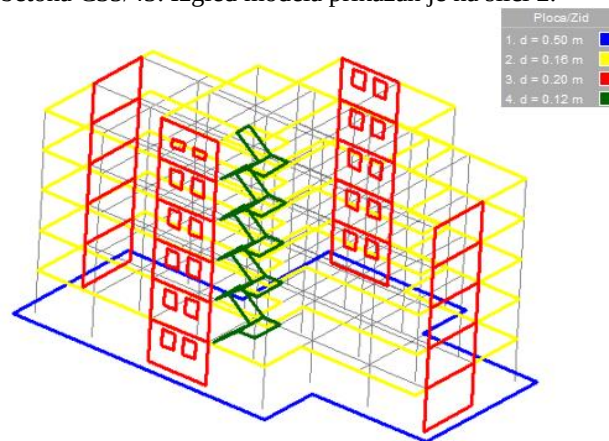
Korišćen je tip 1 elastičnog spektra odgovora, sa vrednostima parametara iz tabele 1.

Tabela 1. Vrednosti parametara koji opisuju preporučeni tip 1 elastičnog spektra odgovora

Kategorija tla	S	$T_B(s)$	$T_C(s)$	$T_D(s)$
A	1.0	0.15	0.4	2.0
B	1.2	0.15	0.5	2.0
C	1.15	0.20	0.6	2.0
D	1.35	0.20	0.8	2.0
E	1.4	0.15	0.5	2.0

4. MODELIRANJE KONSTRUKCIJE

Konstrukcija je modelirana u softveru Tower 6 baziranom na metodi konačnih elemenata. Svi uticaji u konstrukciji određeni su na prostornom proračunskom modelu. Grede i stubovi su modelirani kao linijski elementi, a međuspratne tavanice, krovna ploča i zidna platna kao površinski elementi. Mehaničke karakteristike kao što su računsa čvrstoća betona na pritisak, modul elastičnosti i Poasonov koeficijent određeni su klasom betona C35/45. Izgled modela prikazan je na slici 2.



Slika 2. Izgled proračunskog modela

5. DINAMIČKA ANALIZA

5.1 Modalna analiza

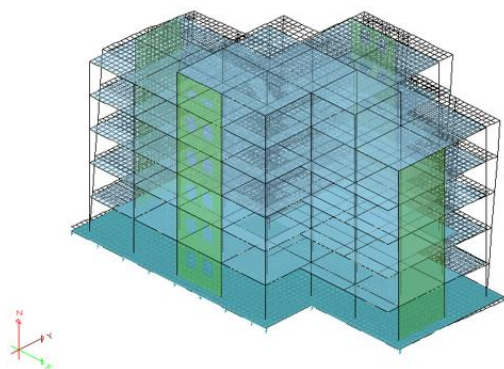
Modalna analiza služi za proračun svojstvenih vrednosti i oblika vibracija konstrukcije. Krutost tavanica u svojim ravnima obezbeđuje da mase i momenti inercije masa svakog sprata mogu biti koncentrisane u centrima masa tavanica.

Stalna opterećenja i opterećenje snegom uzimaju se u punom iznosu, dok se vrednost korisnog opterećenja redukuje odgovarajućim faktorom jer se smatra da neće biti u punom iznosu prisutno za vreme zemljotresa. Vrednosti prva tri perioda oscilovanja konstrukcije, dobijena modalnom analizom, prikazani su u tabeli 2.

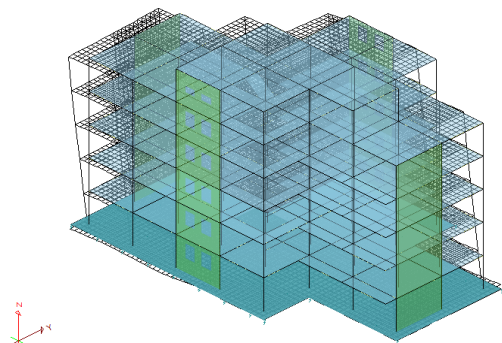
Tabela 2. Vrednosti perioda oscilovanja

	$T [s]$	$f [Hz]$
1	0.7563	1.3222
2	0.6957	1.4374
3	0.5166	1.9357

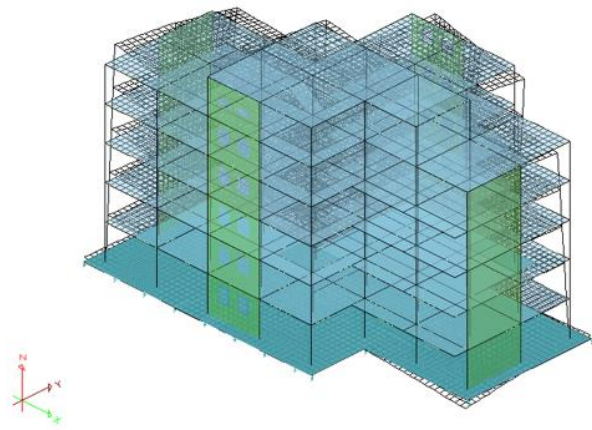
Na slikama 3, 4 i 5 prikazana su prva tri tona oscilovanja konstrukcije:



Slika 3. Forma oscilovanja u prvom tonu



Slika 4. Forma oscilovanja u drugom tonu



Slika 5. Forma oscilovanja u trećem tonu

6. DIMENZIONISANJE

Dimenzionisanje svih elemenata je izvršeno pomoću softverskog paketa Tower 6.0, saglasno propisima [6], [7], na osnovu graničnih uticaja merodavnih kombinacija opterećenja. Pri dimenzionisanju korišćena je klasa betona C35/45 i armatura S500H.

7. PRORAČUNSKE KONTROLE

Proračunske kontrole i rezultati su prikazani izrazima (2), (3), (4), (5) i (6).

Kontrola dozvoljenog napona u tlu:

$$\sigma_{tla} = 149.52 \text{ [MPa]} \leq \sigma_{a,dop} = 240 \text{ [MPa]} \quad (2)$$

Vrednost normalizovane aksijalne sile u stubovima:

$$\vartheta_1 = \frac{N_{ed}}{A_c * f_{cd}} = 0.426 \leq 0.65 \quad (3)$$

Vrednost normalizovanog aksijalnog opterećenja u zidovima:

$$\vartheta_1 = \frac{N_{ed}}{A_c * f_{cd}} = 0.02 \leq 0.40 \quad (4)$$

Kontrola pomerljivosti konstrukcije:

$$F_{V,Ed} = 42500 \text{ [kN]} \leq 158\,011 \text{ [kN]} \quad (5)$$

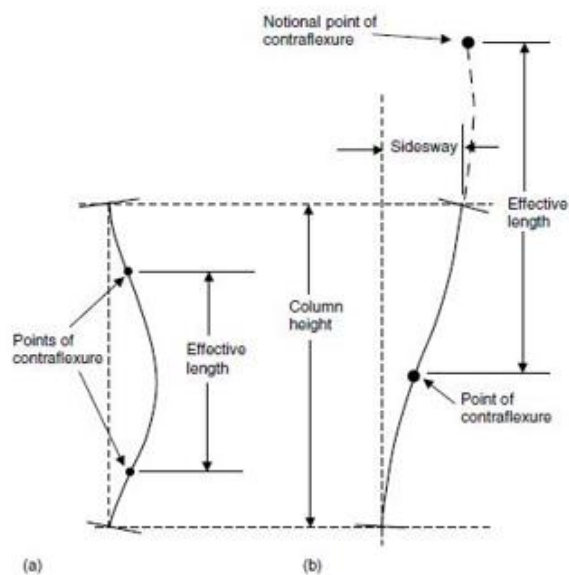
Kontrola relativnog spratnog pomeranja:

$$d_r * v = 8.43 \text{ [mm]} \leq 0.005h = 32 \text{ [mm]} \quad (6)$$

8. ANALIZA EFEKATA TEORIJE DRUGOG REDA

Izvijanje: lom usled nestabilnosti elementa ili konstrukcije pod dejstvom idealno aksijalnog pritiska i bez transversalnog opterećenja.

Efektivna dužina izvijanja [8] je dužina na kojoj se uzima u obzir oblik deformacione krive, slika 6.



Slika 6. Koncept efektivne dužine: a) izolovani stub, b) stub u pomerljivoj konstrukciji

Uticaji prvog reda su uticaji od dejstava koji su proračunati ne vodeći računa o uticaju deformacija konstrukcije, ali uzimaju u obzir geometrijske imperfekcije, dok uticaji efekata teorije drugog reda podrazumevaju dodatne uticaje od dejstava usled deformacija konstrukcije. Metode analize efekata teorije drugog reda su:

- Generalna metoda: nelinearna analiza, uključuje geometrijsku nelinearnost, tako da je potreban odgovarajući softver.
- Metoda nominalne krutosti: u ovoj metodi prvo se odredi nominalna krutost, a zatim se faktor uvećanja momenta kojim množimo momenat prvog reda da bismo dobili momenat drugog reda.
- Nominalna krivina: procenjuje se granično pomeranje, a zatim i momenat drugog reda koji se dodaje na momenat prvog reda.

Primenom ovih analiza, čiji su rezultati prikazani u tabelama 3 i 4, proističe da je za stub dimenzija 45x45 cm dužina izvijanja na svim etažama manja od kritične dužine izvijanja, pa nije potrebno uzeti u obzir efekte drugog reda.

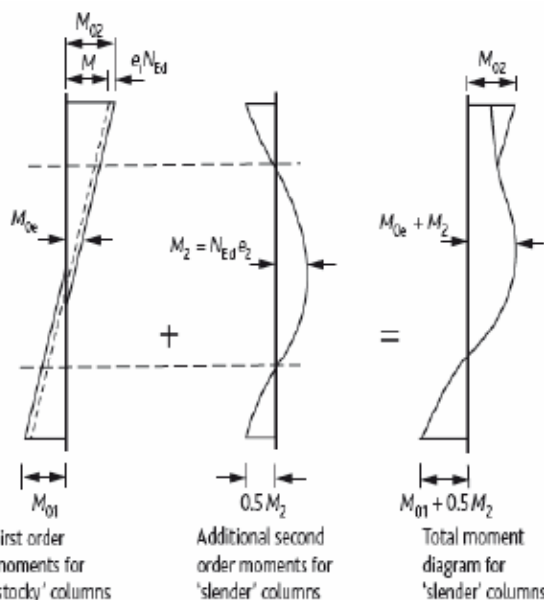
Tabela 3. Poređenje „ λ_{lim} “ i „ λ_x “ za stub 45x45 cm

$\lambda_{lim} = 20 \cdot A \cdot B \cdot C / \sqrt{n}$						
Etaža	A	B	C	n	λ_{lim}	λ_x
P	0.714286	1.1	1.846566	0.52086	40.20677	31.2592
1	0.714286	1.1	2.550681	0.413972	62.29671	29.67337
2	0.714286	1.1	2.666234	0.31635	74.49189	28.93295
3	0.714286	1.1	2.62166	0.221543	87.52704	28.93295
4	0.714286	1.1	2.299393	0.129883	100.2609	28.93295
Pk	0.714286	1.1	2.548773	0.06272	159.9273	30.96153

Tabela 4. Poređenje „ λ_{lim} “ i „ λ_x “ za stub 35x35 cm

$\lambda_{lim} = 20 \cdot A \cdot B \cdot C / \sqrt{n}$						
Etaža	A	B	C	n	λ_{lim}	λ_x
P	0.714286	1.1	2.000	0.888	33.357	40.912
1	0.714286	1.1	2.642	0.705	49.455	41.469
2	0.714286	1.1	2.696	0.536	57.861	41.469
3	0.714286	1.1	2.670	0.372	68.834	41.469
4	0.714286	1.1	2.517	0.212	85.907	41.469
Pk	0.714286	1.1	2.644	0.094	135.857	45.097

Proračunom po metodi nominalne krivine i nominalne krutosti dobijaju se vrednosti momenata drugog reda, kao i količina potrebne armature za pokrivanje tih momenata, kao i njeno procentualno uvećanje u odnosu na momenat prvog reda. Princip sabiranja momenata prvog i drugog reda prikazan je na slici 7.



Slika 7. Princip sabiranja momenata prvog i drugog reda

Rezultat proračuna dati su u tabeli 5.

Tabela 5. Poređenje količine armature stuba za uticaje prvog reda i za uticaje drugog reda

	M [kNm]	$A_{a,uk}$ [cm ²]	Promena $A_{a,uk}$
Uticaji prvog reda	113.99	23.94 cm ²	
Metoda nominalne krivine	149.14	32.07 cm ²	+33.96 %
Metoda nominalne krutosti	164.02	33.51 cm ²	+39.97 %

9. ZAKLJUČAK

Projektovanje objekta je jedan iterativan proces u kome je potrebno pronaći kompromis između arhitektonskih i konstruktivnih zahteva tako da i jedni i drugi budu zadovoljeni.

Kombinacije opterećenja koje uključuju gravitaciono opterećenje i seizmičko dejstvo se javljaju kao merodavne. Nakon završenog proračuna uticaja sprovedene su proračunske kontrole te su korigovane pojedine dimenzije elemenata i klasa betona.

Usvojena armatura pokriva granične uticaje, poštuje preporuke Evrokoda koje se tiču minimalnih i maksimalnih količina armature u pojedinim konstruktivnim elementima te rastojanju i oblikovanju podužne i poprečne armature.

Analiziran je uticaj efekata teorije drugog reda na primeru jednog rama i uticaj momenata drugog reda na povećanje količine armature u stubovima, gde je zabeleženo povećanje količine armature za oko 35 %, što nije zanemarljivo i potrebno je uzeti u obzir.

10. LITERATURA

- [1] Živorad Radosavljević, Dejan Bajić: „Armirani beton 3“, Građevinska knjiga, Beograd 2008.
- [2] Đorđe Lađinović: „Analiza konstrukcija zgrada na seizmičko dejstvo“, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad 1989.
- [3] Zoran Brujić: Materijal sa predavanja iz predmeta Betonske konstrukcije
- [4] EN 1990:2002 Evrokod 0; Osnove proračuna konstrukcija, Beograd, februar 2006.
- [5] EN 1991-1-1: 2002 Evrokod 1; Dejstva na konstrukcije. Deo 1-1: Zapremine težine, Sopstvena težina, korisna opterećenja za zgrade. Deo 1-3; Dejstva snega, Deo 1-4; Dejstva vetra, Beograd, novembar 2009.
- [6] EN 1992-1-1:2004 Evrokod 2; Proračun betonskih konstrukcija. Deo 1-1: Opšta pravila i pravila za zgrade. Beograd, februar 2006.
- [7] EN 1998-1:2004 Evrokod 8; Proračun seizmički otpornih konstrukcija. Deo 1: Opšta pravila, seizmička dejstva i pravila za zgrade. Beograd, novembar 2009.
- [8] A.W. Beeby, R.S. Narayanan: „Designers' guide to eurocode 2“: Design of concrete structures, Thomas Telford Publishing, London, 2005.
- [9] „Eurocode 2: Worked examples“, European concrete platform, Brussels, 2008.
- [10] <http://www.radimpex.rs> – Uputstvo za Tower 6
- [11] <http://www.radimpex.rs> – Uputstvo za Armcad

Kratka biografija:



Aleksandar Dušanić rođen je u Vukovaru 1989. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Seizmičke analize konstrukcija odbranio je 2016. godine.

UPRAVLJANJE TROŠKOVIMA GRAĐENJA ABC METODOM CONSTRUCTION COST MANAGEMENT CONTROL USING ABC METHOD

Suzana Juškić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAĐEVINARSTVO

Kratak sadržaj – U savremenim uslovima poslovanja, klasična metoda obračuna troškova ne obezbeđuje tačne i pouzdane informacije o troškovima i ceni koštanja. Ovaj rad je posvećen ABC metodi koja se nameće kao rešenje za nedostatke klasične metode i koja doprinosi efikasnijem upravljanju troškovima u skladu sa strategijom poslovanja građevinskog preduzeća. Rad ima za cilj da prikaže primenjivost pomenute metode u procesu građenja i da ukaže na viši nivo objektivnosti obuhvatanja indirektnih troškova.

Abstract – It is a fact that the classical approach to cost analysis does not provide sufficient and absolutely reliable results in today's business world. In this paper, we will talk about ABC method, which can be used as a means of overcoming these limitations. This method performs the cost analysis while taking particular business strategies of modern construction companies into account, thus providing more accurate results and more effective solutions. The goal of this paper is to show the application of this method in a process of building construction and to draw attention to higher levels of objectivity when performing indirect costs analysis.

Ključne reči: trošak, upravljanje troškovima, ABC metoda, građevinsko preduzeće, indirektni troškovi, alokacija troškova

UVOD

Značajno područje upravljanja građenjem je upravljanje troškovima, koje treba da obuhvati objektivno prenošenje nastalih troškova u cenu koštanja. Visina troškova građenja ima uticaj na visinu cena što ima takođe snažan uticaj na konkurentski položaj građevinskog preduzeća. Radi poboljšanja upravljanja troškovima razvija se metoda praćenja i raspoređivanja troškova zasnovana na aktivnostima, odnosno ABC metoda.

U ovom radu se istražuje u koliko meri i u kojim slučajevima je ABC (Activity-Based Costing) metoda pogodnija od klasične metode obračuna troškova, mogućnost implementacije ABC metode u procesu građenja, te koje su koristi od njenog uvođenja.

Uslov za izradu novog proizvoda, odnosno izgradnju određenog objekta upotrebnih kvaliteta, zahteva kontinuirano ulaganje u proizvodne faktore, te njihovo trošenje. Osnovni faktori proizvodnje (ljudski resursi, sredstva za rad i predmeti rada) različito učestvuju u formiranju vrednosti novog proizvoda.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bila dr Erika Malešević, red.prof.

1. ODREĐIVANJE TROŠKOVA

Celokupno poslovanje se između ostalih vezuje za ekonomisanje troškovima, odnosno za pronalaženje optimalnog odnosa između troškova i ekonomskih rezultata poslovanja.

Kvantitativni izraz trošenja proizvodnih faktora u naturalnim jedinicama predstavlja utrošak a vrednosno izraženi trošci predstavljaju troškove. Troškovi se mogu posmatrati u užem i širem smislu.

Troškovi u užem smislu predstavljaju novčani izraz utrošenih elemenata proizvodnog procesa - procesa građenja. Troškovi u širem smislu obuhvataju troškove u užem smislu i sve izdatke vezane za poslovanje preduzeća.

1.1 Podela troškova

Zavisno od ciljeva posmatranja, u poslovanju preduzeća nastaju različiti troškovi koji se grupišu na osnovu različitih kriterijuma [8].

1.1.1 Troškovi prema prirodnim vrstama troškova

Pod prirodnim vrstama troškova podrazumevaju se troškovi pojedinih proizvodnih faktora. Troškovi prema elementima proizvodnje dele se na:

- troškovi rada
- troškovi sredstava za rad
- troškovi predmeta rada – troškovi materijala

1.1.2 Troškovi prema mestu nastajanja

Grupisanje troškova prema mestu nastanka ima za cilj što realnije sagledavanje troškova po nosiocima, te dinamike troškova. Prema mestu nastanka troškovi se grupišu na:

- troškove izrade
- režijske troškove

1.1.3 Troškovi prema nosiocima

Klasifikacija troškova po nosiocima vrši se na osnovu vezivanja troškova za određeni proizvod koji ga je prouzrokovao. Po toj klasifikaciji troškovi se grupišu na:

- direktne ili pojedinačne troškove
- indirektni, zajednički ili opšti troškove

Gruba klasifikacija direktnih troškova:

- Eksploatacioni troškovi
- Troškovi osnovnih sredstva (mekhanizacije)
- Plate radnika

Indirektni, zajednički ili opšti troškovi obuhvataju sve ostale troškove koji se ne vezuju za određeni obim proizvoda. U osnovi se indirektni troškovi mogu podeliti na:

- Indirektni troškovi gradilišta:
 - troškove pripremno završnih radova
 - troškove režije gradilišta
 - ostali troškovi gradilišta

- Indirektni troškovi na nivou celog preduzeća:
- režijski troškovi uprave preduzeća
- troškovi svih zajedničkih službi (opslužuju više gradilišta)
- ostali indirektni troškovi na nivou preduzeća

1.1.4 Troškovi prema dinamici proizvodnje

Troškovi u zavisnosti od dinamike proizvodnje dele se na:

- fiksne troškove
- varijabilne troškove

2. ODREĐIVANJE VREDNOSTI GRAĐEVINSKIH RADOVA OBJEKTA

Pri proračunu troškova građenja uglavnom se primenjuje tzv. dodatna kalkulacija jediničnih cena građevinskih proizvoda, radova i usluga. Osnovni princip ove kalkulacije je da se indirektni troškovi građenja na neki način raspodeljuju ili dodaju direktnim troškovima građenja kako bi se dobila jedinična cena građevinskog proizvoda ili usluge.

Dodavanje ili raspodela indirektnih troškova vrši se u dodatnoj kalkulaciji putem tzv. ključa za raspodelu indirektnih troškova na direktne troškove koji se među građevinarima naziva faktor.

Raspodela indirektnih troškova u dodatnoj kalkulaciji, može biti putem [5]:

- faktora na radnu snagu - frs
- menadžer faktora - fm

U građevinskoj proizvodnji ili građenju gde se planira izvedba većina radova sa većim učešćem živog rada primenjuje se uglavnom ključ raspodele indirektnih troškova preko jediničnih direktnih troškova radne snage (bruto plata radnika u proizvodnji) u jediničnu cenu koštanja ili tzv. faktor na radnu snagu, frs.

$$ckt = (trst \cdot frs) + tmatt + tmeht \quad (1)$$

gde su:

trst - jedinični direktni troškovi radne snage

tmatt - jedinični direktni troškovi materijala

tmeht - jedinični direktni troškovi mehanizacije

Faktor se sastoji od dela koji pokriva opšte troškove preduzeća i na njega nadodanog promenljivog dela za svako određeno gradilište i uvek mora biti veći od 1.

Proračun jediničnih cena preko faktora na radnu snagu je vrlo grub način i treba ga izbegavati kad god je to moguće. Nepouzdan je pokazatelj, posebno za one pozicije koje u sebi imaju mali utrošak radne snage a velike materijalne troškove i troškove opreme.

Ako bismo rezimirali indirektno troškove i uporedili način formiranja troškova i njihov uticaj na krajnju prodajnu cenu proizvoda, onda se može zaključiti da uvek treba težiti za menadžer faktorom (ključem raspodele indirektnih troškova u jediničnu cenu koštanja preko ukupnih jediničnih direktnih troškova materijala (predmeta rada), radne snage (troškova živog rada) i troškova rada mašine), jer daje tačnije i preglednije rezultate od faktora na radnu snagu.

$$ckt = dtt \cdot fm \quad (2)$$

$$ckt = (tmatt + trst + tmeht) \cdot fm$$

$$fm = 1 + (itt / dtt) = 1 + fm^* \quad (3)$$

$$fm^* = itt / dtt$$

gde su:

ckt - jedinična cena koštanja po jedinici mere pozicije iz predmeta i predračuna radova

dtt - ukupni direktni troškovi po jedinici mere pozicije iz predmeta i predračuna radova

itt - dio indirektnih troškova koji se raspodeljuje na direktne troškove

tmatt - jedinični direktni troškovi materijala pozicije iz predmeta i predračuna radova

trst - jedinični direktni troškovi radne snage pozicije iz predmeta i predračuna radova

tmeht - jedinični direktni troškovi mehanizacije pozicije iz predmeta i predračuna radova

Jedinične indirektno ili posredne troškove teško je ili nikako racionalno planirati ili iskazati po pojedinačnim planiranim ili izvršenim proizvodnim učincima te se oni u celini kao masa ukupnih troškova uglavnom utvrđuju za ukupni obim proizvodnje. Stoga se ključ raspodele proračunava na neki drugi način. Tu se polazi onda od činjenice koja se da dokazati, koja pretpostavlja međusobnu analogiju ukupnih troškova i jediničnih troškova. Na osnovu ove analogije proizilazi da se ključ raspodele može izračunati na način da je:

$$fm = 1 + (IT/DT) \quad (4)$$

$$fm^* = (IT/DT)$$

$$fm = 1 + fm^*$$

gde je:

IT - ukupni indirektni troškovi izračunati u masi za potrebe građenja

DT - ukupni direktni troškovi građenja

Faktor menadžer je srazmerno manji jer je veća vrednost sa kojom se množi. Dobija se iz odnosa planiranih direktnih (svih za sve projekte i svih za određeno gradilište, odnosno projekat) i indirektnih troškova zajedničkih za celo preduzeće i za određeno gradilište.

Ova dva faktora daju različite cene kalkulativnih pozicija te različitu dinamiku naplate troškova ali ukupno iskalkulisana cena koštanja tj. prodajna cena celog projekta (građevinskog proizvoda) bi trebala biti ista jer se sa njima pokrivaju isti troškovi.

Postavlja se pitanje sa kojom objektivnošću faktor pokriva sve indirektno troškove građenja, odnosno da li se faktor predimenzioniše i ugrožava konkurentsku poziciju građevinskog preduzeća.

Često se u literaturi navodi stav da nove metode obračuna troškova treba da obezbede veću tačnost i efikasnost u alokaciji pojedinih vrsta troškova u odnosu na klasičnu metodu. Za naslovljeni rad odabran je prikaz ABC metode budući da je reč o upravljanju indirektnim, opštim troškovima.

3. ABC METODA OBRAČUNA TROŠKOVA

ABC metoda (Activity-Based Costing), metoda obračuna troškova po aktivnostima, razvijena je osamdesetih godina prošlog veka. Razvili su je američki profesori Robert Kaplan i Robin Kuper.

ABC metoda pruža detaljnije informacije na osnovu kojih će se efikasnije planirati i kontrolisati troškovi i ima tačniju metodologiju upravljanja troškovima.

3.1 Struktura ABC metode

Osnovna pretpostavka ABC metode je da se ne može upravljati troškovima nego aktivnostima koje ih uzrokuju. Sastoji se od pet osnovnih komponenti [9]:

- resursi
- indikator resursa
- aktivnosti
- indikator aktivnosti
- troškovni objekat



Slika 1. Obračun troškova na osnovu aktivnosti

Sistem dodeljivanja i praćenja indirektnih troškova po aktivnostima u osnovi deluje tako da se troškovi resursa dodeljuju aktivnostima primenom prvog stepena troškovnih indikatora, troškovi aktivnosti se primenom drugog stepena troškovnih indikatora dodeljuju troškovnim objektima dok se direktni troškovi, direktno dodeljuju troškovnim objektima (slika 1.) [1].

3.2 Poređenje klasične metode i ABC metode

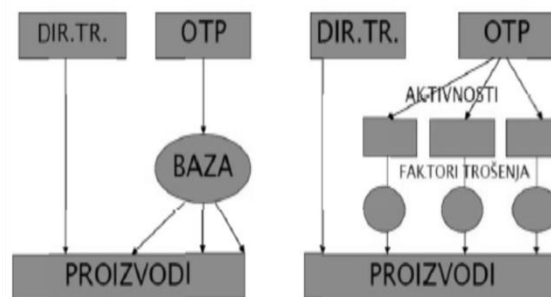
Razlika između klasičnog i savremenog pristupa javlja se u načinu dodavanja indirektnih troškova proizvodnje na nosioce troškova odnosno proizvode. Dok se kod klasičnog pristupa indirektni troškovi proizvodnje identifikuju i prate po mestima troškova, a zatim dodaju putem najčešće jedne stope dodatka na nosioce troškova, savremeni pristup identifikuje i prati indirektne troškove proizvodnje po aktivnostima koje su ih uzrokovale, a zatim ih dodaju putem više različitih faktora trošenja na nosioce troškova (slika 2.).

3.3 Prednosti i nedostaci ABC metode

ABC metoda je efikasnija i preciznija u raspoređivanju indirektnih troškova, ali uprkos tome, ima svoje prednosti i nedostatke.

Prednosti primene ABC metode su [2]:

- bolja identifikacija i kontrola troškova resursa
- mogućnost redukovanja velikog broja nepotrebnih aktivnosti
- bolje dodeljivanje indirektnih troškova na proizvode i usluge
- jasnije povezivanje troškova sa odgovornošću zaposlenih
- mogućnost izrade realnijeg plana i budžeta
- raspored troškova na aktivnosti obavlja se ciljem uspostavljanja uzajamnih veza između objekata kalkulacije i uzročnika troškova
- određuje interese svim funkcijama u preduzeću



Slika 2. Poređenje klasične metode i ABC metode

Pored navedenih koristi od primene ABC metode treba istaknuti i njene nedostatke [2]:

- zahteva mnogo radnog vremena na prikupljanju i obradi podataka
- traži novčana i intelektualna ulaganja
- zahteva primenu specijalizovanog softvera
- teškoće kod identifikovanja uzročnika aktivnosti
- raspored opštih troškova na osnovu uzročnika troškova delatnosti još ne znači da će kompanija, uz smanjenje broja uzročnika, zaista i ostvariti niže troškove
- izbor osnovice za izračunavanje ključeva po ABC metodi stvar je subjektivnog prosuđivanja
- kod klasičnih i manjih firmi moglo bi se dogoditi da troškovi uvođenja ABC metode budu veći od koristi

3.4 Dizajniranje ABC metode

Pri dizajniranju ABC metode postoje četiri koraka [3]:

- identifikacija aktivnosti potrebnih za proizvodnju proizvoda
- dodeljivanje troškova resursa aktivnostima
- identifikacija proizvoda
- povezivanje troškova aktivnosti sa proizvodima pomoću indikatora aktivnosti

U prvom koraku moramo identifikovati aktivnosti koje su neophodne za proizvodnju proizvoda. Nakon identifikacije aktivnosti sledi dodeljivanje troškova resursa aktivnostima putem indikatora resursa koji pokazuju koliko je koja aktivnost uzrokovala trošenje pojedinih resursa.

Troškovi resursa mogu biti dodeljeni aktivnostima na tri načina [4]:

- direktno zaračunavanje
- procena
- arbitražna alokacija

U trećem koraku se identifikuju proizvedeni proizvodi. Dizajneri ABC metode nastoje identifikovati sve outpute koji su rezultat upotrebe resursa čiji se troškovi razdeljuju. Outputi su troškovni objekti za koje se aktivnosti izvode.

četvrtom koraku se povezuju troškovi aktivnosti sa proizvodima preko indikatora troškova aktivnosti. Troškovi aktivnosti mogu se dodeliti proizvodima na ista tri načina kao što se troškovi resursa dodeljuju aktivnostima, tj. direktnim zaračunavanjem, procenom i arbitražnom alokacijom.

Dizajneri ABC metode koriste nekoliko različitih tipova indikatora troškova aktivnosti [7]:

- indikatori transakcije
- indikatori trajanja
- indikatori intenziteta
- indikatori procesa

3.5 Bitni faktori za primenu ABC metode

Za uvođenje ABC metode u poslovanje bitno je utvrditi nekoliko ključnih faktora o poslovnom subjektu:

- privredna delatnost unutar koje se subjekt nalazi
- broj zaposlenih
- organizacionu strukturu
- pravni oblik
- koje su to usluge koje subjekt pruža
- postojeću metodu upravljanja troškovima

Uvođenjem ABC metode mogu se značajno unaprediti troškovni učinak i profitabilnost, s obzirom da postojeća klasična metoda u današnjim uslovima proizvodnje ne daju pravu sliku o stvarnim troškovima pojedinih proizvoda.

Primena ABC metode usmerena je na troškovnu hijerarhiju, a to su četiri sledeće kategorije [6]:

- troškovi jedinice outputa
- troškovi serije proizvoda
- troškovi podrške proizvoda
- troškovi podrške pogona

Implementacija ABC metode u procesu građenja uslovljena je strukturom troškova građenja. Postoje znatne razlike u strukturi cena pojedinih objekata.

Ukoliko indirektni troškovi premašuju jednu trećinu u strukturi ukupnih troškova trebalo bi razmišljati o potrebi uvođenja ABC metode. Ukoliko u strukturi ukupnih troškova indirektni troškovi čine od toga manji deo, savremena ABC metoda ne bi značajno pridonela objektivnijem utvrđivanju jediničnih troškova proizvoda, pa tako ni kvalitetnijoj oceni profitabilnosti proizvoda. Troškovi njene primene bili bi veći od koristi koje bi se njenom primenom mogle ostvariti.

4. ZAKLJUČAK

U ovom radu je dat prikaz teme obračuna troškova na bazi aktivnosti koji je razvijen 80-ih godina prošlog veka u pokušaju da se koriguju nedostaci klasične metode obračunavanja troškova.

ABC metoda omogućuje veću preglednost indirektnih troškova a samim tim daje mogućnost za racionalizaciju odnosno njihovo smanjenje na pojedinim aktivnostima, gde je eventualno došlo do prekoračenja troškova. Implementacija ABC metode nije jednostavna, ali u praksi postoje primeri uspešne primene a i budući razvoj položaja građevinskih preduzeća zahtevaće između ostalih i promenu u načinu poslovanja ukoliko preduzeća žele ne samo da opstanu na tržištu već da u konkurentskom ambijentu obezbede rast i razvoj.

5. LITERATURA

- [1] Belak Vinko: "Računovodstvene podloge za upravljanje troškovima", RriF, br. 9/2005.
- [2] Cokins, G.: "Activity-Based Management - Making it Work", McGraw-Hill, 1996.
- [3] Cooper R., Kaplan R.S., Lawrence M.S., Eileen M., Oehem R.M., "Implementing Activity-Based Cost Management", Institute of Management Accountants 1999.
- [4] C'O Guin, M.: "The Complete Guide to Activity Based Costing", Prentice Hall, 1991.
- [5] Ćirović, G., Luković, O., (2004), "Finansijsko poslovanje i investicije u građevinarstvu", Beograd, VGGŠ
- [6] Horngren C., Datar S.M., Foster G.: "Cost Accounting", Prentice Hall, New Jersey, 2003.
- [7] Kaplan, R., Cooper, R.: "Cost & Effect", Harvard Business School Press, Boston, 1998.
- [8] Malešević E., "Ekonomika građevinarstva i osnovi menadžmenta", Građevinski fakultet Subotica, 1999.
- [9] Miller, A. J. (1996): "Implementing Activity Based Management in Daily Operations", J. Wiley & Sons

Kratka biografija:



Suzana Juškić rođena je u Doboju 1987. god. Master rad na Fakultetu tehničkih nauka u Novom Sadu, iz oblasti Građevinarstvo - Organizacija i tehnologija građenja odbranila je 2016.god.

NUMERIČKI MODEL ZA PROVERU SIGURNOSTI ODBRAMBENIH NASIPA NA SUFOZIJU I FLUIDIZACIJU**NUMERICAL MODEL FOR CHECKING FLOOD DIKES IN TERMS OF SUFFUSION AND FLUIDIZATION**

Aleksandra Lajko, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAĐEVINARSTVO

Kratak sadržaj – Pod modelom strujanja u oblasti podzemnih voda podrazumeva se sredstvo za opis strujanja u nekom hidrogeološkom kolektoru. U poslednje tri decenije nasipi za odbranu od poplava dimenzionišu se po analitičkom modelu nasipa na slabo propusnom pokrovnom sloju. U ovom radu prikazan je numerički model u istu svrhu, za simulaciju strujanja u podlozi i u telu odbrambenog nasipa pri uslovima visoke voda za vreme odbrane od poplave. Uspostavljenim modelom proverena je osetljivost podloge nasipa na sufoziju i fluidizaciju u zavisnosti od dispozicije nasipa i karakteristika podloge nasipa. Rezultati su upoređeni sa rezultatima analitičkog modela i formulisani su zaključci u pogledu efikasnosti numeričkog modela i dobijenih rezultata.

Glavne reči: Odbrambeni nasip, podzemne vode, ravansko strujanje, gradijent izviranja, numerički model

Abstract – In the groundwater domain, flow model refers to a tool capable for describing flow in a particular aquifer. During the last three decades design of flood dikes utilized the analytical model of dike built on a confined aquifer. This work presents a numerical model for the same purpose, i.e. simulation of flow in the aquifer and through the dike in case of high water during flood defence. The established model was used to check the sensitivity of the confining soil layer to suffusion and fluidization depending on dike configuration and soil characteristics. The results of numerical simulations are compared to the results of the analytical model. Conclusions regarding efficiency of the numerical model and the results obtained are derived.

Key words: Flood dike, ground water, 2D flow, flow gradient, numerical model

1. UVOD

Cilj odbrane od poplava je da se smanje ili da se spreče štete na plavnom području. Odbrana od poplava može biti pasivna ili aktivna. Aktivnom odbranom od poplava utiče se na promenu proticaja poplavnih voda tako da se u celosti ili delimično zaštiti ugroženo područje ili područja. Ovo se postiže izgradnjom retencija kao i uređenjem korita reke.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Đula Fabian, vanr. prof. Građevinskog fakulteta Subotica.

Pasivnom odbranom od poplava direktno se štiti plavno područje. Ovo se postiže izgradnjom nasipa za zaštitu od poplava na odbrambenoj liniji oko plavnog područja i odbrambenim radovima u toku prolaska poplavnih voda [1].

Ovim radom prikazan je numerički model u svrhu provere hidrauličke stabilnosti (na sufoziju i fluidizaciju) podloge nasipa za odbranu od poplava. Rezultati numeričkih opita upoređeni su sa rezultatima klasičnog analitičkog postupka i izvedeni su zaključci u pogledu efikasnosti numeričkog modela.

1.1. Kritični vidovi stabilnosti odbrambenih nasipa

Pri projektovanju nasipa proveravaju se sledeći vidovi ugrožavanja sigurnosti [2]:

1. ispiranje tla (sufozija) u branjenom terenu,
2. lom tla (fluidizacija) u branjenom terenu,
3. ispiranje stope nasipa (usled kontaktne filtracije),
4. rušenje obale,
5. sufozija tela nasipa,
6. klizanje tela nasipa po podlozi (kao i klizanja branjene i nebranjene kosine),
7. erozija usled talasa i
8. prelivanje.

Iskustvo u projektovanju novih nasipa, sanacije i nadvišenja postojećih, ukazuje na to da je najteže postići odgovarajuću sigurnost u pogledu hidrauličke stabilnosti podloge nasipa, pošto zavisi od prirodom nametnutih uslova (na primer, od geološke slojevitosti tla, od njegovih geomehničkih i hidrogeoloških karakteristika), koji se inženjerskim postupcima ne mogu promeniti, eventualno se njihov nepovoljan uticaj može ublažiti. Prvenstveno se misli na prolom (tj. fluidizaciju) i na ispiranje (sufoziju) pokrovnog sloja.

To su vidovi sigurnosti pod rednim brojem 1. i 2. sa prethodnog spiska.

Sufozija je ispiranje tj. „iznošenje” malih zrna porozne sredine usled prevelikih brzina strujanja. Time se povećava propusnost tla, tj. koeficijent filtracije K , što ubrzava započeti proces. Prema mestu nastanka može biti unutrašnja i spoljna sufozija. Sufozija je dugotrajni, spor proces.

Suprotan proces sufoziji jeste kolmiranje – popunjavanje šupljina u poroznoj sredini sitnijim česticama.

Fluidizacija je podizanje tla sa svim svojim komponentama usled velike razlike pijezo kote, ΔH , odnosno pritiska, p , ispod i iznad pokrovnog sloja podloge nasipa. To je kratkotrajni proces koji se manifestuje naglim izbacivanjem tla u vidu mutnog izvora.

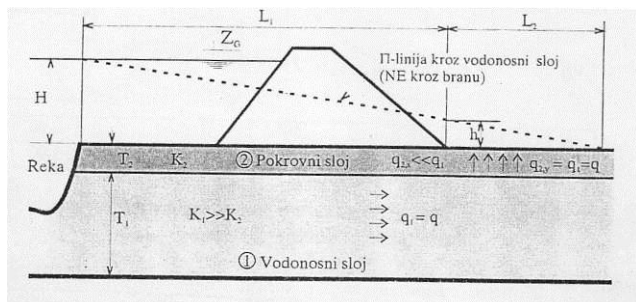
2. PROVERA HIDRAULIČKE STABILNOSTI NASIPA

Za proveru hidrauličke stabilnosti odbrambenih nasipa do pojave personalnih računara bili su korišćeni pojednostavljeni analitički modeli, a u skorašnje vreme i numerički modeli.

2.1. Analitički modeli za proveru hidrauličke stabilnosti podloge nasipa

U hidrotehničkoj praksi za proračun nasipa, uglavnom, koriste se dve šeme:

1. Šema nasipa sa pokrovnim i vodonosnim slojem, po kojoj je uspostavljen najčešće korišćen analitički model, data je slikom 1 [3, str. 141].



Slika 1. Poprečni presek kroz nasip i podlogu sa pokrovnim slojem

Pretpostavke izvođenja analitičkog modela su:

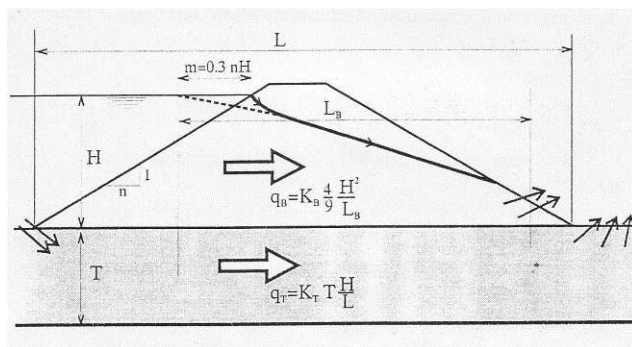
- a) na analiziranom području podloga je dvoslojna, sastoji se od pokrovnog sloja preko vodonosnog sloja, ispod čega je nepropusna podloga,
- b) pokrovni sloj je znatno slabije propustan od vodonosnog,
- c) u pokrovnom sloju horizontalni transport vode je zanemarljiv, $q_{2x} \ll q_1$, pa se ukupan horizontalni protok ostvaruje u vodonosnom sloju, $q \approx q_1$,
- d) na inundaciji se zanemaruje infiltracija kroz pokrovni u vodonosni sloj,
- e) strujanje u vodonosnom sloju računski se tretira kao horizontalno,
- f) u vodonosnom sloju pijezo linija pada linearno u smeru branjenog terena,
- g) pijezo kota u branjenom terenu poklapa se sa kotom terena,
- h) sva voda dotekla iz reke u vodonosni sloj izliva se na teren kroz pokrovni sloj, $q = q_{2y}$, i
- i) telo nasipa je nepropusno.

Po ovoj šemi utvrđuju se veličine h i L_2 , a na osnovu njih prosečni gradijent pijezometra kroz pokrovni sloj, I , koji se upoređuje sa kritičnim gradijentom na sufoziju i fluidizaciju, pa se donosi zaključak o hidrauličkoj stabilnosti pokrovnog sloga.

2. Šema nasipa na vodonosnoj podlozi je druga analitička šema koja se u praksi koristi, znatno ređe od šeme 1 [3, str. 143]. Po ovoj šemi izlazni gradijent kod branjene nožice nasipa ne može se utvrditi, pa je iz tog razloga izlazni gradijent često aproksimiran prosečnim gradijentom u stopi nasipa, tj. u temeljnoj spojnici:

$$I = H/L \quad (1)$$

Ta aproksimacija je, međutim, pregruba i nije na strani sigurnosti.



Slika 2. Poprečni presek kroz nasip i podlogu bez pokrovnog sloja

2.2. Numerički model za proveru hidrauličke stabilnosti podloge nasipa

Numerički model ravanskog strujanja u vertikalnoj ravni zasniva se na rešavanju diskretizovane diferencijalne jednačine strujanja u mnoštvu tačaka računске mreže u oblasti strujanja, uključivši vodonosnu sredinu, pokrovni sloj i telo nasipa. Kao rezultat, neposredno se dobije vrednost pijezo kote u svim računskim tačkama, na osnovu čega mogu se odrediti brzine strujanja i gradijenti pijezo kote, kako prosečni, tako i lokalni. Numerički model se zasniva na realnijim pretpostavkama od pretpostavki analitičkog postupka (na primer, pokazateljima geoloških slojeva podloge i tela nasipa mogu se zadati proizvoljne vrednosti), pored toga, numerički model daje detaljan uvid u raspored pijezo kota i brzina u strujnom polju.

U pogledu hidrauličke stabilnosti podloge nasipa, numerički model daje kvalitetniji rezultat od analitičkih modela, naime pored prosečnog gradijenta utvrđuje i lokalne gradijente pijezometra.

3. PRIMENJEN NUMERIČKI MODEL

3.1. Jednačina strujanja i diskretizacija strujnog polja

Diferencijalna jednačina strujanja koja čini osnovu numeričkog modela ustaljenog, ravanskog strujanja u vertikalnoj ravni (x_1, x_3) u telu nasipa i u izotropnim slojevima podloge nasipa je Laplasova jednačina:

$$\frac{\partial^2 \Pi}{\partial x_1^2} + \frac{\partial^2 \Pi}{\partial x_3^2} = 0. \quad (2)$$

Imajući u vidu da je strujno polje znatno razučeno u horizontalnom pravcu upravno na reku, korišćena je ekvidistantna mreža sa različitim korakom u horizontalnom i vertikalnom pravcu:

$$\Delta x_1 / \Delta x_3 = 6. \quad (3)$$

Sa ovim, diskretizovani oblik jednačine strujanja je:

$$\frac{\Pi_{i+1,k} - 2\Pi_{i,k} + \Pi_{i-1,k}}{\Delta x_1^2} + \frac{\Pi_{i,k+1} - 2\Pi_{i,k} + \Pi_{i,k-1}}{\Delta x_3^2} = 0, \quad (4)$$

odakle je izražena nepoznata zavisna promenljiva u pojedinoj računskoj tački i,k :

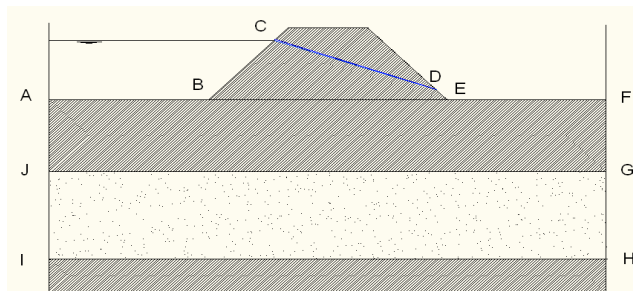
$$\Pi_{i,k} = \frac{\Delta x_3^2 (\Pi_{i+1,k} + \Pi_{i-1,k}) + \Delta x_1^2 (\Pi_{i,k+1} + \Pi_{i,k-1})}{2\Delta x_1^2 + 2\Delta x_3^2}. \quad (5)$$

Računska mreža u vertikalnom pravcu pokriva telo nasipa i slojeve ispod nasipa od kote terena sve do nepropusne

geološke podloge, a u horizontalnom pravcu od reke do vertikalnog preseka daleko u branjenom terenu, u kome je uticaj reke na nivo podzemne vode već zanemarljiv. U vertikalnom pravcu svaki geološki sloj je diskretizovan sa najmanje pet računskih tačaka, što omogućuje određenje kako prosečnog, tako i lokalnog gradijenta vertikalnog izviranja, od interesa za procenu hidrauličke stabilnosti podloge nasipa.

2.5. Granični uslovi

Granični uslovi mogu biti spoljni, po granicama strujne oblasti (duž linije A-B-C-D-E-F-G-H-I-J-A), i unutrašnji, duž linija razgraničenja geoloških slojeva različitih karakteristika (duž linije G-J).



Slika 3. Prikaz granica strujne oblasti

Spoljni granični uslovi:

- Po vertikali kod reke (A-J-I), po terenu inundacije (A-B) i po kvašenoj kosini nasipa (B-C):

$$\Pi = \Pi_{REKE}. \quad (6)$$

- Duž provirne linije u telu nasipa (C-D) i po nepropusnoj podlozi (I-H):

$$q = -k \frac{\partial \Pi}{\partial s} = 0 \Rightarrow \frac{\partial \Pi}{\partial s} = 0. \quad (7)$$

- Duž procedne linije u branjenoj nožici nasipa (D-E), po horizontalnoj površini branjenog terena (E-F) i po vertikali u branjenom terenu (F-G-H):

$$\Pi = x_3. \quad (8)$$

4. ANALIZA OSETLJIVOSTI

Prilagodljivost numeričkog modela raznovrsnim graničnim uslovima daje mogućnost za sprovođenje analize osetljivosti strujne slike od dispozicije nasipa, slojevitosti i hidrogeoloških karakteristika podloge. U tu svrhu formulisani su numerički opiti primenom oznaka na slici 1.

4.1. Razmatrani numerički opiti

Da bi se smanjio broj neophodnih opita za sprovođenje analize, slični slučajevi su grupisani normiranjem promenljivih karakterističnom visinom naleganja vode na nasip, H , i koeficijentom filtracije vodonosnog sloja, k_1 . Karakteristike sprovedenih numeričkih opita sažete su tabelom 1. Tako definisan minimalni skup numeričkih opita može dati jasan uvid u uticaj debljine pokrovnog sloja, T_2 , debljine vodonosnog sloja, T_1 , širine predterena, L_1 , i propusnosti pokrovnog sloja, K_2 , na strujnu sliku, samim tim na gradijent izviranja vode kod nožice nasipa u branjenom terenu:

$$I = I \left(\frac{T_1}{H}, \frac{T_2}{H}, \frac{L_1}{H}, \frac{K_2}{K_1} \right), \quad (9)$$

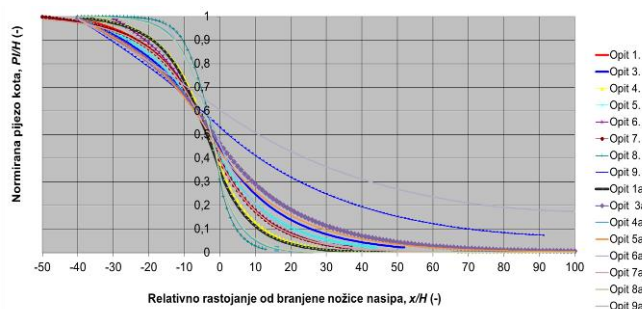
tj. na pojavu sufozije i fluidizacije.

Broj opita	$\frac{T_2}{H}$	$\frac{T_1}{H}$	$\frac{L_1}{H}$	$\frac{k_2}{k_1}$
1a	0,4167	2,25	40	0,01
1	0,75	2,25	40	0,01
2	0	2,25	40	0,01
3	1,4167	2,25	40	0,01
3a	2,083	2,25	40	0,01
4a	0,75	0,416	40	0,01
4	0,75	1,25	40	0,01
5	0,75	3,25	40	0,01
5a	0,75	5,75	40	0,01
6a	0,75	2,25	15	0,01
6	0,75	2,25	30	0,01
7	0,75	2,25	50	0,01
7a	0,75	2,25	80	0,01
8	0,75	2,25	40	0,1
8a	0,75	2,25	40	0,05
9	0,75	2,25	40	0,001
9a	0,75	2,25	40	0,0005

Tabela 1. Pregled normiranih pokazatelja sprovedenih numeričkih opita.

4.2. Prikaz i analiza rezultata

Prvo, slikom 4 su za sve numeričke opite prikazane pijezo linije po granici pokrovnog i vodonosnog sloja od reke (sa leve strane grafikona) u smeru branjenog terena (tj. ka desnoj ivici grafikona). Najintenzivniji pad pijezo linije je ispod samog nasipa, što se tumači povećanim brzinama i gubicima strujanja u tom uskom grlu. Pijezo linija ne opada linearno kako je to pretpostavljeno u analitičkom modelu, već se asimptotski približava branjenom terenu. Sa jedne strane, to ukazuje na činjenicu da se šema 1 analitičkog modela može opravdano koristiti samo kada je pokrovni sloj znatno - recimo 1000 puta - manje propustan od vodonosnog sloja. Sa druge strane, širina raskvašenog pojasa u branjenom terenu je znatno veća nego što bi se očekivalo na osnovu analitičkog modela. Slika 4. potvrđuje da je branjena nožica nasipa najugroženije mesto u pogledu sufozije i fluidizacije, pošto je na tom lokalitetu ($x/H = 0$) pijezo kota najvišlja u celom branjenom terenu ($x/H \geq 0$).



Slika 4. Normirana pijezo linija po granici pokrovnog i vodonosnog sloja

Dodatno, primera radi, u nastavku je prikazan uticaj relativne propusnosti pokrovnog sloja, K_2/K_1 , na gradijent izviranja, I , i na relativnu širinu raskvašenog pojasa u branjenom terenu, L_2/H . Iz tabele 1 sledi da su u ovu svrhu korišćeni rezultati opita 8, 8a, 9, i 9a.

Sprovođenjem numeričke simulacije strujanja po navedenim numeričkim modelima došlo se do raspodele vrednosti pijezo kote u strujnoj oblasti, na osnovu čega su za lokalitet branjene nožice nasipa sračinati gradijenti pijezo kote, i to:

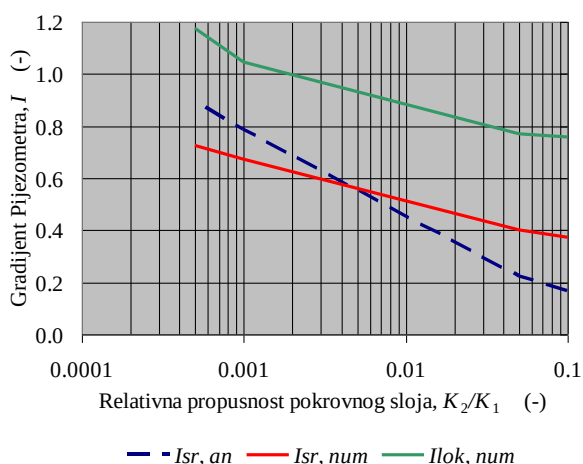
- prosečni (srednji) gradijent izviranja kroz pokrovni sloj ($I_{sr, num}$),
- lokalni rezultujući gradijent izviranja u nožici nasipa ($I_{lok, num}$).

Na istom grafikonu, upoređenja radi, prikazana je i vrednost prosečnog (srednjeg) gradijenta izviranja kroz pokrovni sloj ($I_{sr, an}$) utvrđena analitičkim postupkom.

Na osnovu rezultata prikazanih slikom 5 može se zaključiti da je vrednost prosečnog gradijenta kroz pokrovni sloj određen analitičkim postupkom veća od vrednosti utvrđene numeričkim postupkom samo u oblasti $K_2/K_1 < 0.004$, tj. kada je vodonosni sloj bar 250 puta propusniji od pokrovnog sloja. To znači da – ukoliko se hidraulička stabilnost podloge nasipa proverava na osnovu prosečnih gradijenata – analitički postupak je na strani sigurnosti u odnosu na numerički postupak samo za slučajeve $K_2/K_1 < 0.004$.

Dalje, lokalni rezultujući gradijenti izviranja u podlozi kod nožice nasipa su 1.3 do 2 puta veći od prosečnih gradijenata, što opravdava primenu visoke vrednosti koeficijenta sigurnosti, $S=2$, pri korišćenju analitičkog postupka za procenu sigurnosti podloge nasipa na sufoziju i fluidizaciju.

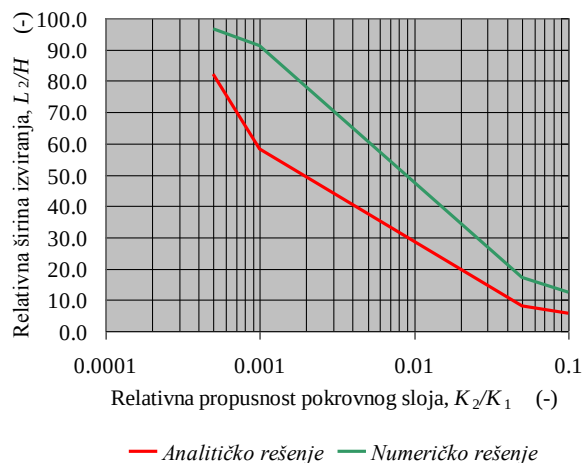
Gornja analiza jasno ukazuje na potencijalno najkritičnije mesto samog nasipa i njegove okoline koje neprekidno treba osmatrati tokom odbrane, a to je branjena nožica nasipa. Lokalni izlazni gradijenti na toj lokaciji jako često prevazilaze kritične, pa se najpre u njenoj okolini može očekivati pojava mutnih izvora, što je indikator nastupanja sufozije ili fluidizacije. To zahteva hitnu intervenciju, što daje značaj operativnoj odbrani od poplava.



Slika 5. Uticaj relativne propusnosti pokrovnog sloja na gradijent izviranja

Pored Izlaznih gradijenata, pomenutim numeričkim opitima utvrđena je i širina zone izviranja podzemnih voda u branjenom terenu, rezultat je prikazan slikom 6.

Slika 6 potvrđuje očekivanu tendenciju da se širina zone izviranja povećava sa smanjenjem propusnosti pokrovnog sloja. Tačniji, numerički postupak daje 1.2 do 2.2 puta veću širinu izviranja od analitičkog postupka. Prema tome, u ovom pogledu analitički postupak potcenjuje potencijalnu širinu raskvašavanja branjenog terena, što nije na strani sigurnosti.



Slika 6. Relativna širina izviranja u zavisnosti od relativne propusnosti pokrovnog sloja

5. ZAKLJUČAK

Primena analitičkog postupka provere hidrauličke stabilnosti odbrambenih nasipa kod nas je odomaćena zahvaljujući prof. Karlu Rezničeku, i koristi se već skoro četiri decenije. Međutim, pojava personalnih računara omogućila je uvođenje numeričkih modela u analizu stabilnosti odbrambenih nasipa. U ovom radu je u kratkim crtama prikazan jedan numerički model i ispitane su njegove mogućnosti u upoređenju sa klasičnim analitičkim postupkom.

Pokazalo se da numerički model daje detaljniji uvid u strujnu sliku i realniju procenu pokazatelja od kojih zavisi sigurnost odbrambenih nasipa.

Lokalni gradijent izviranja kroz pokrovni sloj – koji se analitičkim postupkom ne može utvrditi, već samo numeričkim – u svakom slučaju prevazilazi računске prosečne vrednosti, a često i kritične vrednosti sa aspekta sigurnosti.

Time je opravdano uvođenje numeričkog modela – danas već dostupnog i inženjerima praktičarima – u oblast odbrane od poplava.

4. LITERATURA

- [1] Karlo Rezniček, "Analiza tipskih profila nasipa uz Dunav i Tisu, Zaseban otisak iz godišnjaka Samoupravne interesne zajednice za osnovno uređenje voda Vojvodine", Vode Vojvodine, 1997.
- [2] Karlo Rezniček, "Odbrana od poplava u ravničarskim predelima", Izvod iz doktorske disertacije autora, Novi Sad, 1978.
- [3] Ljubodrag Savić, "Uvod u hidrotehničke građevine", Građevinski fakultet Beograd, 2003.

Kratka biografija:



Aleksandra Lajko rođena je u Požarevcu 1988. god. Master rad na Fakultetu tehničkih nauka iz oblasti Građevinarstva – Hidrotehničke konstrukcije, odbranila je u oktobru 2016. god.

HIDRAULIČKA ANALIZA PRIMENE RACIONALNE METODE I SWMM-a NA ODVOĐENJE ATMOSFERSKIH VODA NASELJA ĐURĐEVO

HYDRAULIC ANALYSIS OF APPLICATION OF RATIONAL METHODS AND SWMM's STORM WATER DRAINAGE ON SETTLEMENTS ĐURĐEVO

Biljana Bugarinović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAĐEVINARSTVO

Kratak sadržaj – U radu je obrađen proračun atmosferske kanalizacije za naselje Đurđevo, opština Žabalj. Proračun je vršen dvema metodama; racionalnom metodom i korišćenjem softvera SWMM. Prilikom proračuna je korišćena kiša 2-godišnjeg povratnog perioda, u trajanju od 20min. Uporedna analiza je vršena da bi se došlo do zaključaka da li se rezultati obe metode mogu poistovetiti ili je za projektovanje nophodno uraditi proračun obema metodama i izvršiti dimenzionisanje sa optimalnim vrednostima proticaja.

Abstract – The paper describes the calculation of storm sewer for settlement Đurđevo, municipalities Žabalj. The calculation is done by two research methods; rational method and the use of software SWMM. The calculation used rain of 2-year return period, a period of 20min. Comparative analysis was done in order to come to conclusions if the results of both methods can identify or is it in design necessary to done calculation by both methods and make dimensioning with optimum flow values.

Cljučne reči: Atmosferska kanalizacija, Swmm, racionalna metoda, Đurđevo

1. UVOD

U radu je izvršen proračun atmosferske kanalizacije za kaselje Đurđevo korišćenjem dve metode; racionalne metode i softverskog paketa SWMM. Cilj rada jeste da se naprave paralele između dve metode i da se pored njihovi rezultati. Za proračun je korišćena kiša 2-godišnjeg povratnog perioda, sa dužinom trajanja od 20 min. Pored ovoga, u radu je potrebno izvršiti modelovanje atmosferske kanalizacije za kišu 20-godišnjeg povratnog perioda, kako bi se videlo da li i kiša koja ima ovaj povratni period može da bude prihvaćena za usvojenu mrežu. Kako je reč o ruralnom području, odabir mreže čini kombinacija otvorenih kanala i zatvorenih kolektora.

2. TEORIJSKE OSNOVE

U literaturi je poznato više različitih metoda na osnovu kojih se može doći do proračuna količine padavina za posmatrano područje. Rad će ispratiti dve metode: racionalnu metodu i metodu nastalu primenom računara - SWMM metodu.

NAPOMENA:

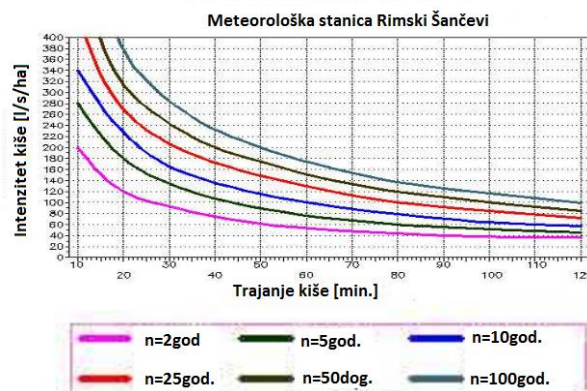
Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Matija Stipić.

2.1. Racionalna metoda

Ova metoda se najčešće koristi na urabanimizovanim slivovima, faktorom ograničenja na 15km². Ukupna količina kiše Q koja u jedinici vremena može da se slije sa površine F , iznosi [1]:

$$Q = c * i * F \quad (1)$$

gde je i intenzitet kiše a c koeficijent koji predstavlja proizvod koeficijenta oticaja Ψ i srednjeg intenziteta kiše. Kod proračuna racionalne metode, neophodno je poznavanje intenziteta padavina. Ovaj faktor najčešće zavisi od odabranog vremena koncentracije sliva, odnosno od vremena koje je potrebno da se najudaljenija tačka koja padne na posmatranom slivu slije do kontrolnog profila. Za racionalnu metodu neophodno je poznavanje koeficijenta oticaja na posmatranom slivu, kao i količina padavina sa merne stanice. Sa merne stanice se najčešće traže podaci u vidu ITP krive, za različite povratne periode (Slika 1.).



Slika 1. ITP kriva Rimski Šančevi

2.2. SWMM metoda

Softverski paket SWMM (EPA Storm Management Model) je dinamički, fizički baziran model koji simulira proces transformacije padavina u oticaj [2].

Osnovu matematičkog modelovanja u mreži čini jednačina održanja mase i jednačina količine kretanja za neustaljeno tečenje u otvorenim kanalima (Sent-Venanove jednačine):

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0 \quad (2)$$

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x} \left(\frac{Q^2}{A} \right) + g * A * \frac{\partial H}{\partial x} + g * A * S_f = 0 \quad (3)$$

gde je Q proticaj, x rastojanje duž toka, A površina poprečnog preseka, H pijezometrijska kota i S_f nagib linije energije.

Da bi se odredio proticaj u svakoj vezi, kao i piježometarska kota u svakom čvoru tokom vremenskog koraka Δt , dinamička jednačina i jednačina kontinuiteta se moraju istovremeno rešavati jednom od numeričkih metoda. U tu svrhu, softver raspolaže sa tri varijante i to model ustaljenog tečenja, model kinematskog talasa i model dinamičkog talasa. Najtačnije rešenje daje model dinamičkog talasa, dok pomenuta dva preostala daju menje precizne rezultate ali opet tačne.

2.3. Dimenzionisanje atmosferske kanalizacije

Nakon što se odredi količina padavina, prelazi se na dimenzionisanje mreže. Mreža atmosferske kanalizacije se može dimenzionisati kao sistem otvorenih kanala, zatvorenih kolektora ili kombinacijom prethodna dva. U ovom radu je korišćena kombinacija otvorenih kanala i zacevljenih deonica. Opšti kriterijum prilikom projektovanja otvorene kanalske mreže jeste da je minimalna dubina u otvorenom kanalu 0.50m, dok maksimalna dubina ukopavanja iznosi 1.20m. Sem ovog kriterijuma, važnu ulogu igra i brzina tečenja jer se ne sme dozvoliti da ona bude prevelika, kako ne bi došlo do spiranja čestica i narušavanja stabilnosti kosina. Pored toga, ne sme se dozvoliti ni preblagi pad, kako ne bi došlo do zadržavanja vode u kanalima što pogoduje razvoju mikroorganizama i bržem rastu trave [3]. Prilikom proračuna otvorene kanalske mreže kreće se od pretpostavke da je tečenje u kanalu ustaljeno, dakle nema promene brzine u profilu tokom vremena. Za proračun otvorenih kanala se koristi Šezi-Maningova jednačina:

$$Q = A * \frac{1}{n} * R^{2/3} * \sqrt{I} \quad (4)$$

gde od parametara figurišu proticaj; Q , nagib dna kanala; I , hidraulički radijus; R , površina poprečnog preseka; A i Maningov koeficijent; n .

Kada je reč o zacevljenim deonicama, tu se koristi nešto drugačiji pristup. Profili cevi se biraju na osnovu dijagrama Q/Q_{pp} i V/V_{pp} [1]. Ovde se dozvoljava veća brzina tečenja; minimalno 0.60 m/s, kako ne bi došlo do lepljenja čestica za zidove cevi. Takođe nisu dozvoljene ni prevelike brzine, kako ne bi došlo do odnosešna čestica sa zidova cevi, što kasnije za posledicu ima narušavanje stabilnosti zidova cevi.

Međutim, iz praktičnih razloga sistem ne može da se projektuje tako da prilikom padavina obezbedi potpunu zaštitu od plavljenja ili prezasićena tla vodom [1]. Zato se na pojedinim mestima dozvoljava da dođe do plavljenja ukoliko padne velika količina kiše, što se u literaturi definiše kao dozvoljena učestalost padavina.

3. STUDIJA SLUČAJA

Da bi se mogao izvršiti proračun atmosferske kanalizacije, nophodno je poznavanje područja na kome se proračun vrši. Zato je potrebno poznavanje koeficijenta oticaja, koje je različito za svako područje jer direktno zavisi od urbanizacije posmatranog područja. Sem ovoga, važni faktori su reljef područja i recipijent atmosferske kanalizacije.

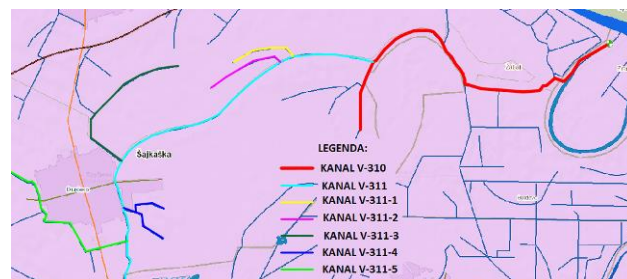
3.1. Karakteristike naselja Đurđevo

Naselje Đurđevo pripada opštini Žabalj, Južno-bački okrug. Površina naselja iznosi 341.32ha. Pripada ruralnom tipu naselja, gde se stanovništvo pretežno bavi

poljoprivredom. Na osnovu [4] došlo se do podataka o kakvom geološkom profilu naselja je reč. Naime, područje je ravničarsko sa vrlo blagim usponima i padovima. Područje je veoma plodno za poljoprivredu proizvodnju sa dobrom moći infiltracije.

3.2. Recipijent atmosferske kanalizacije

Na posmatranom području postoji mreža odvodnih kanala (Slika 2).



Slika 2. Mreža odvodnih kanala naselja Đurđevo

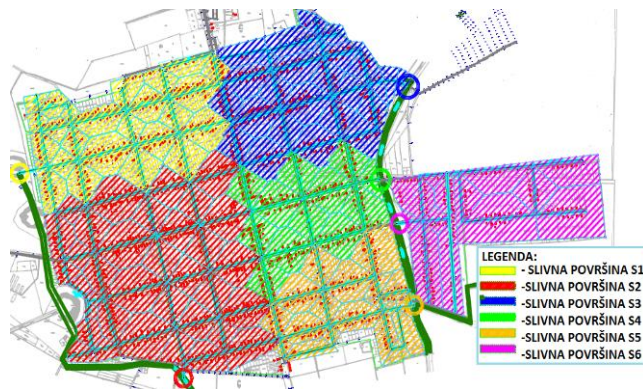
Svi oni su mogli da se iskoriste u odvođenju padavina, jer su u relativno dobrom stanju. Za proračun se, međutim iskoristio samo odvodni kanal V-311 i V-311-5.

3.3. Koeficijent oticaja

Koeficijent oticaja predstavlja odnos svih tipova slivnih površina; p_i i njihovih koeficijenta oticaja; ψ_i sa ukupnom površinom sliva; $\sum p_i$, odnosno:

$$\psi_{sr} = \frac{\sum p_i * \psi_i}{\sum p_i} \quad (5)$$

Već prethodno je rečeno da naselje Đurđevo pripada ruralnom tipu naselja. Ovo znači da je koeficijent oticaja manji nego kod urbanizovanih područja, budući da je većina terena pod livadom, što znači da se pretpostavlja da će se sva kiša koja padne na livadu upiti ili ispariti. Veća urbanizacija naselja zahteva i veći koeficijent oticaja pa se zato proračun izvršio sa dve varijante. Prva da je usvojen uniforman koeficijent oticaja, u vrednosti od $\psi=0.137$ i drugi, različit za svako slivno područje (Slika 3.).



Slika 3. Slivne površine naselja

4. REZULTATI PRORAČUNA

Korišćenje uniformnog koeficijenta i kod jedne i kod druge metode je dalo uvećane vrednosti proticaja. Ovo je prvenstveno proisteklo iz razloga što se kod naselja tek na najurbanizovanijem delu može očekivati ova vrednost

koeficijenta oticaja, dok se za sva ostala područja može očekivati koeficijent sa nižim vrednostima. Zato se za dimenzionisanje ali i za poređenje rezultata mreže usvojio proračun sa različitim koeficijentom oticaja za svaku slivnu površinu. Za poređenje rezultata će se odabrati po jedna karakteristična deonica iz svakog slivnog područja.

4.1. Upporedni rezultati proračuna

Rezultati dobijeni proračunom po obema metoda će se staviti u sledeću tabelu (Tabela 1).

Tabela 1.-Dobijeni rezultati po racionalnoj i SWMM metodi

Slivna površina	Koeficijent oticaja	Koeficijent nepropusnosti	Proticaj na izlivu	
			Racionalna metoda	SWMM metoda
S1	0.072	7.20	252.03	223.44
S2	0.079	7.90	562.45	422.23
S3	0.067	6.70	301.90	249.74
S4	0.094	9.40	328.55	356.17
S5	0.096	9.60	355.91	232.08
S6	0.055	5.50	218.77	228.73

Iz slivne površine S1 će se posmatrati rezultati za izlivni čvor I1. Za pomenutu deonicu, proticaj na izlivu dobijen racionalnom metodom iznosi 252.03 l/s, dok u SWMM metodi iznosi 223.44 l/s. Kada je u pitanju slivna površina S2, posmatraće izliv I2. Ova slivna površina je ujedno i najveća na celom naselju te su se očekivale najveće vrednosti proticaja za obe metode. Kada je u pitanju racionalna metoda, proticaj iznosi 562.45 l/s, dok je vrednost proticaja u SWMM metodi maksimalna u 15.20 i iznosi 422.23 l/s.

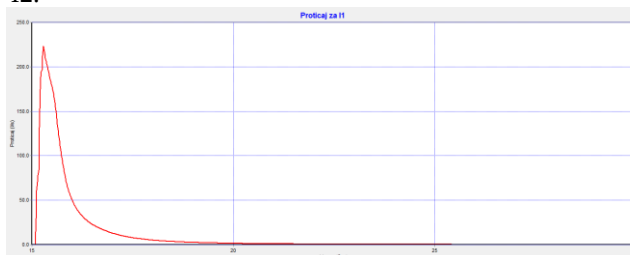
Za slivnu površinu S3 se vezuje manja urbanizacija sliva što za posledicu ima manju vrednost koeficijenta oticaja. Zato je bilo za očekivati manje vrednosti proticaja, a on u SWMM metodi iznosi 249.74 l/s a u racionalnoj 301.90 l/s. Slivna površina S4 ima veću urbanizaciju od prethodnih i proticaj dobijen racionalnom metodom iznosi 328.55 l/s a SWMM metodom 356.17 l/s. Slivna površina S5 po površini odgovara prethodnoj, sa skoro istim koeficijento oticaja. Proticaj u SWMM metodi iznosi 232.08 l/s a u racionalnoj 355.91 l/s. Slivna površina S6 ima najmanji stepen urbanizacije. Vrednost proticaja na ovom slivnom području u SWMM metodi 228.73 l/s dok je u racionalnoj metodi 218.77 l/s.

4.2. Diskusija rezultata

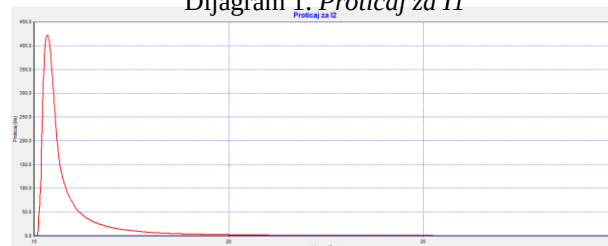
Za slivnu površinu S1, kao što se moglo videti u Tabeli 1., dobijena je očekivana vrednost rezultata, pri čemu je proticaj u racionalnoj metodi uvećan za oko 11%. Proticaj na izlivu u SWMM metodi je dobro raspodeljen (Dijagram 1.)

Uvećana vrednost proticaja na slivnoj površini S2 za racionalnu metodu je još izražajnija i iznosi oko 25%. Kada je u pitanju softverska obrada podataka, ona je

dobro izvedena, što ilustruje sledeći dijagram (Dijagram 2.). Takođe, kako je ova slivna površina veća od S1, primetna je i ravnomernija raspodela padavina na izlivu I2.



Dijagram 1. Proticaj za I1



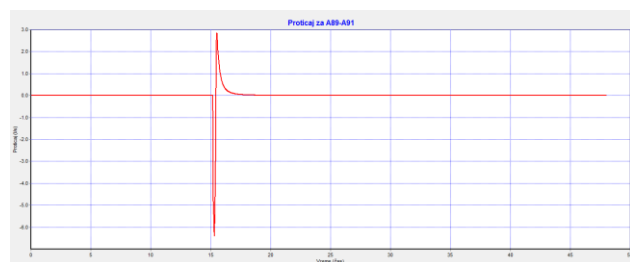
Dijagram 2. Proticaj za I2

Slivna površina S3 je po površini manja od prethodne dve, pa je očekivano da se dobiju određene nestabilnosti u pogledu rezultata. Razlika proticaja iznosi oko 18% u korist racionalne metode. Softver je pokazao odlične rezultate i vremenska raspodela je dobro aproksimirana. Softverski prikaz rezultata prikazan je grafički (Dijagram 3.)



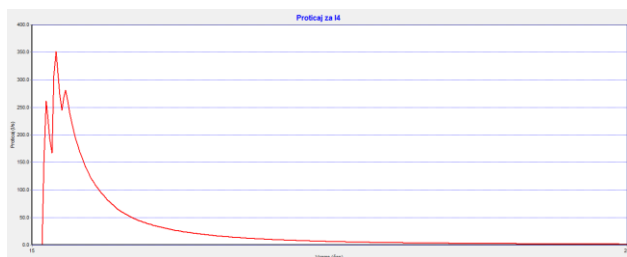
Dijagram 3. Proticaj za I3

Slivna površina S4 ima veći koeficijent oticaja i manju slivnu površinu u odnosu na prethodne tri. Ovde je došlo do odstupanja u očekivanom rezultatu te su vrednosti u racionalnoj metodi nešto manje u odnosu na SWMM metodu. Međutim, ovo se može tumačiti kao veliki broj početnih deonica koje su mogle izazvati grešku u proračunu. Sem tog, ovde dolazi do slučaja da se kiša u deonici A89-A91, umesto da se ocedi u niže deonice, prvo vrati u ovu deonicu iz deonice A91-A95 a zatim ocedi (Dijagram 4.)



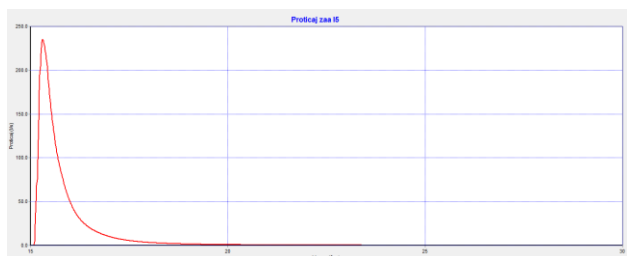
Dijagram 4. Proticaj za deonicu A89-A91

Zbog ove nestabilnosti, kasnije se javlja poremećaj u proticaju na izlivu I4 (Dijagram 5.) što dovodi do uvećanih vrednosti proticaja u odnosu na racionalnu metodu.



Dijagram 5. Proticaj za I4

Posmatrana slivna površina S5 ima vrlo sličan koeficijent oticaja kao i S4 ali ovde nema poremećaja u vidu tečenja u deonicama. Zato je ovde došlo do očekivanih rezultata u korist SWMM metode. Izraženo u procentima, ova vrednost iznosi oko 35%. Pravilno tečenje ilustruje i naredni dijagram (Dijagram 6.).



Dijagram 6. Proticaj za I5

Slivna površina S6, koja ima i najmanju vrednost koeficijenta oticaja dala je i najmanju vrednost proticaja racionanom metodom. Ovo je u neku ruku bila očekivana vrednost, budući da su proticaj i koeficijent oticaja u direktnoj vezi. Međutim, zbog velikog broja početnih deonica kao i slučaja poremećaja proticaja u nekoliko deonica, dolazi do toga da proticaj dobijen racionalnom metodom biva manji. Ovaj poremećaj se prenosi na izliv, što ilustruje naredni dijagram (Dijagram 7.).



Dijagram 7. Proticaj za I6

5. ZAKLJUČAK

U radu je obrađen proračun atmosferske kanalizacije na naselju Đurđevo korišćenjem racionalne metode i SWMM metode. Tema rada je nastala kao želja da se uporede rezultati i da se donese zaključak da li je moguće dobiti iste vrednosti korišćenjem tradicionalne metode i softverske metode. Za očekivati je bilo da će vrednosti korišćenjem tradicionalne metode biti veći, zato što se prilikom proračuna koristi samo maksimum pale kiše, dok se kod SWMM metode uvodi vremenska raspodela kiše. Prilikom proračuna je korišćena ITP kriva sa Rimskih

Šančeva, sa povratnim periodom od 2 godine. Najveći problem kod rezultata jeste postojanje velikog broja početnih deonica što za posledicu u SWMM metodi daje neočekivane rezultate. Naime, u slivnim površinama S4 i S6 postoji oko 50% početnih deonica kod kojih je visina padavina toliko mala da prilikom padavina daju za rezultat 0.00 l/s. Ovo ne znači da je softver pogrešio, nego da su to toliko male vrednosti da ne mogu da budu ispisane jer softver ispisuje vrednosti na dve decimale. Rezultati dobijeni proračunom dovode do zaključka da racionalna metoda daje očekivane, nešto veće rezultate proticaja, sem u slučajevima gde je prisutna smetnja u vidu većeg broja početnih deonica ili kada dođe do vraćanja kiše iz narednih deonica, pre nego što se kiša slije. Za potrebe projektovanja može se reći da obe metode daju dobre rezultate. Ukoliko se želi imati veći stepen sigurnosti, onda je opravdano koristiti racionalnu metodu a u suprotnom se mogu rezultati dobijeni SWMM metodom uzeti kao dovoljno tačni.

Kada je reč o kiši sa 20-godišnjim povratnim periodom, SWMM metoda ne pokazuje tečenje pod pritiskom što je pokazatelj dobro odabrane mreže. Takođe, samo dva čvora u celokupnom sistemu bivaju poplavljena, što je dobar rezultat. Treba imati na umu da se dimenzionisanje zadržalo sa kiše 2-godišnjeg povratnog perioda, što je dobar rezultat.

6. LITERATURA

- [1] "Pisana predavanja iz predmeta Komunalna hidrotehnika za studente IV godine odseka za hidrotehniku" - doc dr Matija Stipić, dipl. inž. građ., Fakultet tehničkih nauka u Novom Sadu, Departman za građevinarstvo i geodeziju, Katedra za hidrotehniku, Novi Sad 2016 god.
- [2] "Storm Water Management Model User's Manual Vrsion 5.1." – Lewis A. Rossman, Environmental Scientist, Emeritus U.S. Environmental Protection Agency, National Risk Management Laboratory Office of Research and Development U.S. Environmental Protection Agency 26 Martin Luther King Drive Cincinnati, OH 45268, September 2015.
- [3] „Open Channel Flows“, Prof. Dr. Atıl BULU, Istanbul Technical University College of Civil Engineering Civil Engineering Department Hydraulics Division
- [4] "Sistem za odvodnjavanje Vrbica-Đurđevo-Mošorin - Elaborat o zemljištu", Hidrozavod, Novi Sad, avgust 1972 god.

Kratka biografija:

Biljana Bugarinović rođena je u Tesliću 1988. god. Diplomski bechelor rad iz predmeta "Hidrotehnički objekti i sistemi" na Fakultetu tehničkih nauka odbranila je 2014 god. na temu "Hidraulička analiza vodosnabdevanja prve visinske zone opštine Teslić". Master rad na temu "Hidraulička analiza preimene racionalne metode i SWMM-a na odvođenju atmosferskih voda naselja Đurđevo" obranila je 2016 god.

Matija Stipić rođen je u Somboru 1964. godine. Doktorirao je na Fakultetu tehničkih nauka 2009. godine, a od 2011. godine ima zvanje docenta. Oblast interesovanja su hidraulika i komunalna hidrotehnika.

HIDRAULIČKA ANALIZA RAZVOJA KANALIZACIONE MREŽE NASELJA VRBAS I SLIVA CENTRALNOG POSTROJENJA ZA PRERADU OTPADNIH VODA**HYDRAULIC ANALYSIS OF THE SEWAGE SYSTEM DEVELOPMENT FOR VRBAS TOWN AND CATCHMENT OF THE CENTRAL WASTEWATER TREATMENT PLANT**

Viktor Keresteš, Matija Stipić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAĐEVINARSTVO

Kratak sadržaj –U okviru rada data je hidraulička analiza razvoja kanalizacione mreže otpadnih voda naselja Vrbas. Za modelovanje kanalizacionog sistema korišćen je programski paket Epa SWMM koji omogućava brz, iterativan postupak projektovanja. Projektovani kanalizacioni sistem optimalno rešava odvođenje upotrebljenih voda naselja Vrbas u cilju poboljšanja životnog standarda i zaštite životne sredine. Takođe, modeliran je i sliv CPPOV (centralno postrojenje za preradu otpadnih voda) – izvršen je hidraulički proračun otpadnih voda naselja Vrbas zajedno sa selima vrbaske opštine, kao i naseljima Crvenka i Kula, i dobijeno je ukupno hidrauličko i organsko opterećenje CPPOV. Na kraju rada, dat je opis odabranog rešenja rekonstruisanog stanja sa hidrauličkim proračunom i odgovarajućim brojem grafičkim priloga.

Abstract –The paper presents the hydraulic analysis of the wastewater sewage system development for Vrbas town. Software package EPA SWMM was used for modeling the sewage system which enables fast, iterative process of designing. The designed sewage system optimally solves the drainage of wastewater from Vrbas aiming to improve the living standards and environmental protection. In addition, wastewater treatment plant catchment was modeled and the hydraulic wastewater was calculated for Vrbas and villages of municipality Vrbas with the towns Kula and Crvenka. A total hydraulic and organic loading of CPPOV was also obtained. At the end, the description of the selected reconstruction solution was given with a complete hydraulic calculation and an adequate number of graphical attachments.

Ključne reči: otpadne vode, kanalizaciona mreža, hidraulička analiza, rekonstrukcija

1. UVOD

Ubrzanom industrijalizacijom i nebrigom za sakupljanje, odvođenje i prečišćavanje industrijskih i komunalnih otpadnih voda basena Vrbas-Kula-Crvenka tokom druge polovine XX veka, Veliki bački kanal je postao najzagađeniji vodotok u Srbiji, pa i u Evropi.

Devedesetih godina prošlog veka su pokrenute aktivnosti u vezi sa sagledavanjem problematike zagađenja i koncipiranja rešenja za prečišćavanje otpadnih voda industrije i stanovništva. Ove aktivnosti se nisu realizovale zbog ekonomskih sankcija i ratova na našim prostorima.

Prethodna studija razmatrala je deset varijacija rešenja za prečišćavanje otpadnih voda, a kao ekonomski i tehnički najracionalnije odabrano je sledeće: na centralnom postrojenju za preradu otpadnih voda (CPPOV) se prečišćavaju primarno tretirane otpadne vode industrija "Sloboda", "Eterna", "Istra", "Carnex" i "Vital" i stanovništva Crvenke, Kule i Vrbasa [1].

Početak radova na CPPOV je započet 2010. godine i, posle nekoliko odlaganja radova, nalazi se u završnoj fazi izgradnje [2].

U Vrbasu preovlađuje separatan kanalizacioni sistem (odvojeni kanali za sakupljanje upotrebljenih-fekalnih i kišnih voda). [1] Manji deo sistema u gradskom centru je opšteg tipa, odnosno istim sistemom kanala se sakupljaju upotrebljene i kišne vode. U ovom trenutku, odvođenje atmosferskih voda nije rešeno zaokruženim (potpuno formiranim) sistemom [2].

Postojeća kanalizacija je na kraju svog eksploatacionog veka. U planu je rekonstrukcija fekalne kanalizacione mreže, rehabilitacija postojeće atmosferske mreže, kao i njen završetak u delovima naselja u kojima nije izgrađena [2]. Planira se separatan kanalizacioni sistem. Današnje stanje ne može se smatrati zadovoljavajućim: veliki broj havarija često uzrokuje deformacije kolovoza, dok nedovoljni kapaciteti imaju za posledicu učestalo izlivanje atmosferskih i otpadnih voda [3].

2. OPŠTE KARAKTERISTIKE PREDMETNOG PODRUČJA

Opština Vrbas se nalazi u Srbiji, na teritoriji AP Vojvodine u Južnobačkom okrugu, i zauzima površinu od 376 km². Vrbas se graniči sa opštinama: Kula, Mali Idoš, Srbobran, Temerin, Novi Sad, Bački Petrovac, Bačka Palanka i Odžaci. Opština Vrbas se prostire u vidu pravougaone trake, pored Velikog bačkog kanala, u podnožju Telečke zaravni, a delimično na njenim padinama.

Opštinu čini osam katastarskih opština: Vrbas, Vrbas grad, Bačko Dobro Polje, Zmajevo, Ravno Selo, Kucura, Savino Selo i Kosančić.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Matija Stipić, docent.

Prema popisu stanovništva RZS iz 2011. godine, u Vrbasu ima 42.092 stanovnika, što je za 3.760 stanovnika manje u odnosu na popis iz 2002. godine (-8,2%).

Vrbaš je značajan industrijski centar Bačke, a najintenzivnija je prehrambena industrija.

Prema geografskom položaju, Naselje Vrbaš pripada oblasti umereno-kontinentalne klime. Njene odlike su umereno topla leta i umereno hladne, snežne zime, sa izraženim prelaznim godišnjim dobima. Padavina ima u svim godišnjim dobima, ali je njihova ukupna mesečna i godišnja količina veoma kolebljiva.

Teritorija Opštine Vrbaš prostire se na lesnoj zaravni i lesnoj terasi, na nadmorskoj visini od 81 do 104 metara. U celini posmatrano, reljef pokazuje naglašenu nagnutost od severozapada prema jugoistoku, kao i od severa prema jugu.

3. MODELIRANJE TEČENJA U KANALIZACIONIM SISTEMIMA

U radu se prikazuje jedna mogućnost matematičkog modeliranja, odnosno računarske simulacije tečenja vode u kanalizacionim sistemima. Navedene su osnovne jednačine tečenja vode i analizirani pojedini problemi koji se javljaju pri modeliranju, kao i uobičajeni postupci njihovog prevazilaženja.

Kod kanalizacionih sistema za upotrebljene (otpadne) vode, promena dotoka je po pravilu spora. Stoga se oni sa uspehom mogu tretirati bez uzimanja u obzir neustaljenih pojava. Nasuprot ovome, u sistemima za odvođenje kišnih voda (uključujući mešovite sisteme), neustaljenost tečenja je izrazita, pa je za korektnu analizu njihovog funkcionisanja većinu elemenata neophodno i tretirati u skladu sa tim.

Kretanje vode u kanalizacionim cevima obično se opisuje jednačinama održanja mase i količine kretanja za linijsko tečenje vode u otvorenom kanalu, tzv. Sent-Venantove jednačine, koje se mogu definisati u obliku [4]:

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0 \quad (1)$$

$$\frac{\partial Q}{\partial t} + \frac{\partial avQ}{\partial x} + gA \frac{\partial h}{\partial x} + gA(I_e - I_{dna}) = 0 \quad (2)$$

U mirnom tečenju potrebni su po jedan granični uslov (nivo, proticaj ili krivina protoka) na uzvodnom i nizvodnom kraju, a u burnom režimu zadaju se dva granična uslova na uzvodnom kraju i nijedan na nizvodnom [4].

Jednačine 1 i 2 nazivaju se i model dinamičkog talasa ili kompletni dinamički talas. Izostavljanje pojedinih članova omogućava jednostavnije i efikasnije numeričko rešavanje, ali se sa time gubi na tačnosti dobijenog rešenja.

4. PROGRAMSKI PAKET EPA SWMM

Određivanje merodavnih proticaja za dimenzionisanje sistema fekalne i kišne kanalizacije se sve češće vrši upotrebom programskih paketa, pomoću kojih se simulira proces transformacije padavina u oticaj. Za obavljanje proračuna neophodna je izrada matematičkog modela proučavanjaog sliva. To je složeni proces koji obuhvata

definisanje elemenata sliva i kanalizacionog sistema, zadavanje njihovih karakteristika, odnosno vrednosti parametara modela, i izbor metoda proračuna.

Jedan od najčešće korišćenih simulacionih modela je programski paket Epa SWMM (Storm Water Management Model). Pomoću SWMM softvera se može vršiti hidrauličko i hidrološko modeliranje. Tečenje u mreži se programom može simulirati pomoću modela ustaljenog tečenja, kinematskog i dinamičkog talasa, dok za proračun infiltracije postoje tri modela.

4.1. Mogućnosti programa

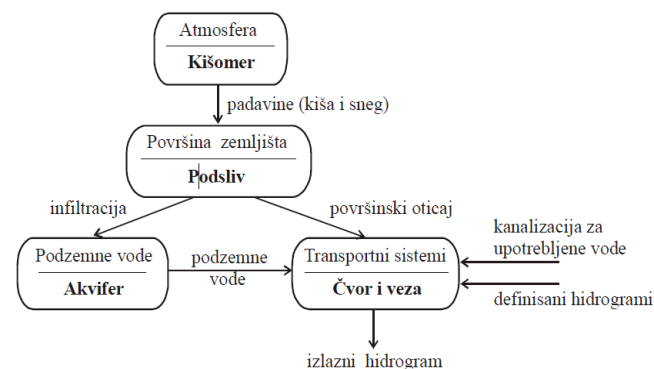
Program EPA SWMM koristi se za hidrauličko modeliranje koje se primenjuje za proračun tečenja u cevima i kanalima, kroz rezervoare ili objekte za raspodelu proticaja.

Neke od mogućnosti programa su:

- kolektorska mreža neograničene veličine,
- upotreba velikog broja standardnih oblika poprečnog preseka otvorenih i zatvorenih kolektora, kao i prirodnih kanala,
- modeliranje specijalnih elemenata kao što su rezervoari, objekti za tretman vode, objekti za raspodelu proticaja ili pumpe,
- primena zadatih spoljašnjih doticaja i elemenata kvaliteta vode koji se mogu dodati površinskom oticaju, podzemnim vodama, infiltraciji od padavina i proticaju za vreme sušnog perioda,
- primena metode kinematskog ili dinamičkog talasa za proračun tečenja u cevima i
- modeliranje različitih režima tečenja kao što su efekat uspora, povratno tečenje, tečenje pod pritiskom i površinsko zadržavanje vode.

Simulacioni program SWMM svoj koncept zasniva na interakciji nekoliko glavnih komponenti životne sredine, koje se modeliraju takozvanim objektima (slika 1).

SWMM kreira dinamički, fizički baziran model koji se može koristiti za simulaciju jednog događaja ili za kontinualnu simulaciju količine i kvaliteta oticaja.



Slika 1. Koncept simulacionog modela EPA SWMM

5. HIDRAULIČKI PRORAČUN

Hidraulički proračun kanalizacije upotrebljenih voda izvršen je primenom programskog paketa EPA SWMM koji omogućava hidraulički proračun tečenja u sistemu cevovoda. Ovaj način proračuna pokazuje svoju punu snagu u današnjim uslovima, kada su personalni računari dostigli visok nivo potrebnih performansi za komplikovane matematičke proračune primenjene u ovoj metodi.

Korišćeni pristup se ogleda u sledećem:

- predstavljanje kolektora realnim oblikom i položajem;
- proračun tečenja upotrebljenih voda u mreži kolektora simultanim rešavanjem kompletnih jednačina neustaljenog tečenja, uključujući i eventualno tečenje pod pritiskom;
- analiza raličitih početnih uslova;
- analiza različitih graničnih uslova;
- analiza rada crpnih stanica i
- posmatranje sistema kao celine.

5.1. Obrada podataka

Hidrauličkom proračunu prethodila je obrada podloga, odnosno priprema ulaznih podataka za simulaciju tečenja u sistemu kolektora. Ova priprema je urađena korišćenjem savremenih postupaka, pri čemu se imalo u vidu da tačnost pri unošenju podataka direktno utiče na dobijene rezultate. To podrazumeva da su se na rastersku podlogu - situaciju naselja, u programskom paketu SWMM nanosili podaci potrebni za pripremu datoteke digitalne predstave terena.

5.2. Mreža projektovanog cevovoda

Nakon što je početna varijanta kanala kreirana digitalizacijom na ekranu preko rasterske podloge naselja, uz istovremeno unošenje pretpostavljenih dimenzija preseka svih kolektora, softverski su testirani ovi podaci u mreži. Osnova za proračun je kasnije proveravana po kapacitetu, odnosno dimenzionisana na osnovu rezultata upotrebljenih voda.

Usvojen Maningov koeficijent hrapavosti, s obzirom na izabrani cevni materijal, iznosi $n=0,015m^{-1/3}s$.

5.3. Podela na pripadajuće dužine

Pod pripadajućom dužinom se podrazumeva deo mreže sa koga voda otiče u jednu određenu deonicu kolektora. U smislu hidrauličkog proračuna kanalizacije upotrebljenih voda, pripadajuće dužine će svakoj deonici generisati tačan broj ekvivalentnih stanovnika koji gravitiraju i opterećuju određenu deonicu kolektora. Tačan broj ekvivalentnih stanovnika se izračunava za kolektor i po formuli:

$$ESLi = \frac{N \times l}{\Sigma l}, \quad (3)$$

gde je l – pripadajuća dužina kolektora (m), $ESLi$ – broj stanovnika koji opterećuju određenu deonicu kolektora, a Σl – ukupna dužina fekalne kanalizacione mreže.

Generisani broj stanovnika koji opterećuje pojedinu deonicu sistema se množi sa parametrom koji predstavlja specifičnu produkciju otpadnih voda i koeficijentom dnevne neravnomerenosti kako bi se dobila količina srednjeg dnevnog protoka upotrebljenih voda za svaku deonicu:

$$Qsrdn = ESLi \times q_{spec} \times K_{dn}^{max} \quad (4)$$

Pošto korišćeni softver omogućava proračun nestacionarnog tečenja u cevovodima i simulaciju dnevnog protoka (za 24 časa), izračunate količine srednjeg dnevnog protoka deonice su se “upuštale” u kolektore umnožene sa koeficijentima časovnih neravnomernosti.

6. PROJEKCIJA UKUPNIH KOLIČINA OTPADNIH VODA KOJA ĆE SE PREČIĆAVATI NA CPPOV

U narednim tabelama (Tabele 1-3) dat je pregled ukupnih količina otpadnih voda koja će dolaziti na CPPOV, a Slika 2 prikazuje hidrogram dotoka na CPPOV.

Tabela 1. Vrbas – stanovništvo, ustanove i mala industrija

Broj stanovnika	28000
q_{spec} (l/st.dan)	160
Qsr^{dn} (m ³ /dan)	4480
Qsr^{dn} (l/s)	51.85
K_{max}^{dn}	1.5
Q_{max}^{dn} (m ³ /dan)	6720
Q_{max}^{dn} (l/s)	77.78
K_{max}^h	1.19
Q_{max}^h (l/s)	92.47

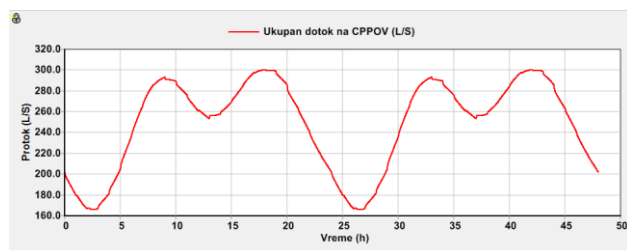
Tabela 2. Maksimalno časovno opterećenje CPPOV

Maksimalno časovno opterećenje	(m ³ /dan)	(l/s)
Vrbas stanovništvo	7989.41	92.47
Vrbas industrija	514.94	5.96
Savino Selo I Kucura st.	2430.43	28.13
Ravno Selo, Zmajevo I BDP stanovništvo	3326.4	38.50
Crvenka stanovništvo	3100.9	35.89
Kula stanovništvo	5619.46	65.04
Strane vode	3469.82	40.16
Ukupno	25 945.1	300.29

Tabela 3. Rekapitulacija organskog opterećanja CPPOV

Zagađivač		BPK ₅ (kg/dan)	Br. ES
Vrbas stanovništvo		1 680	28 000
Vrbas industrija		600	10 000
Sela vrbaske opštine - stanovništvo	Savino Selo + Kucura	510	8 500
	Ravno Selo + Zmajevu + Bačko Dobro Polje	738	12 300
Crvenka stanovništvo		630	10 500
Kula stanovništvo		1 248	20 800
Ukupno		5 406	90 100

*Napomena: 1 ES = 60 mg/l BPK₅



Slika 2. Hidrogram dotoka na CPPOV

7. ZAKLJUČAK

U diplomskom radu je prezentovano idejno rešenje kanalizacionog sistema za upotrebljene vode naselja Vrbas. Urađeno je rešenje kanalisanja sistema za projektni period od 20 godina sa ciljem poboljšanja životnog standarda, kvaliteta podzemnih voda i životne sredine. Njegovom izgradnjom se štiti životna sredina, smanjuje potencijalna opasnost od zaraze i poboljšava stanje ekosistema naselja.

Projektovana je mreža ukupne dužine oko 72 km, koja ima 10 crpnih stanica. Usvojena je početna dubina ukopavanja od 1.4 m od kote terena do kote vrha cevi kako bi bili ispunjeni svi uslovi vezani za minimalnu dubinu ukopavanja cevi. Uzimajući u obzir karakteristike otpadne vode (sa aspekta transporta materijala određene krupnoće) i činjenicu da se kanalizacija gradi za duži vremenski period, usvojen je minimalni prečnik cevi Ø250 mm. [5]

Minimalni prečnik cevi na početnim deonicama nije hidraulički opravdan, ali na tim deonicama kriterijum održavanja ima prioritet.

Da bi se izbegli veliki investicioni troškovi izgradnje kanalizacije, kao i da bi se uštedelo na zemljanim radovima, prilikom projektovanja fekalne kanalizacije naselja Vrbas, u većim deonicama su primenjeni mali padovi, koji uslovljavaju male brzine tečenja otpadne vode. Usled malih brzina tečenja vode, koje su manje od kritičnih, dolazi do pojave taloženja materija u cevovodu. Radi sprečavanja taloženja na deonicama se preporučuje češće ispiranje mreže tj. najbolje par puta godišnje.

Za kanalizacionu mrežu naselja Vrbas usvojeno je optimalno punjenje do 2/3 cevi, jer je tečenje u kanalizacionim kolektorima naselja Vrbas gravitaciono.

Za cevni materijal izabrane su cevi od tvrdog polivinil hlrida (PVC), zbog niza prednosti i povoljnih karakteristika u odnosu na druge materijale, a za potisne deonice usvojene su cevi od polietilena (PE).

Od nastanka ideje o rešavanju problema otpadnih voda naselja Vrbas, Kula i Crvenka do njene realizacije prošlo je više od 30 godina. U međuvremenu, nakon privatizacije dosta fabrika je prestalo sa radom, a one koje su ostale da rade, rade smanjenim kapacitetom. Oslabljenja industrija, kao i sve izraženija migracija stanovništva, dovele su do toga da je prečistač predimenzionisan.

8. LITERATURA

- [1] Građevinski fakultet, Institut za hidrotehniku, "Uređaj za prečišćavanje otpadnih voda grada i industrije Titovog Vrbasa i Kule – Idejni projekat – Tekstualni deo", Subotica, 1989.
- [2] "Strategija održivog razvoja Opštine Vrbas 2014-2020", Vrbas, 2014.
- [3] "Plan generalne regulacije za naseljeno mesto Vrbas", Vrbas, 2010.
- [4] "M. Stipić, "Pisana predavanja iz predmeta Komunalna hidrotehnika za studente IV godine odseka za hidrotehniku", Fakultet tehničkih nauka, Novi Sad, 2016.
- [5] "M. Milojević, "Snabdevanje vodom i kanalisanje naselja". Građevinski fakultet, Beograd, 1987.

Kratka biografija:

Viktor Keresteš rođen je u Vrbasu 1986. godine. Master rad na Fakultetu tehničkih nauka iz oblasti Građevinarstvo – Hidrotehnika odbranio je 2016.godine.

Matija Stipić rođen je u Somboru 1964. godine. Doktorirao je na Fakultetu tehničkih nauka 2009. godine, a od 2011. godine ima zvanje docenta. Oblasti interesovanja su hidraulika i komunalna hidrotehnika.

НЕЛИНЕАРНА СТАТИЧКА СЕИЗМИЧКА АНАЛИЗА ВИШЕСПРАТНИХ АБ ОКВИРА**NONLINEAR STATIC SEISMIC ANALYSIS OF MULTISTOREY RC FRAMES**Никола Рајић, Ђорђе Лађиновић, *Факултет техничких наука Нови Сад***Област – ГРАЂЕВИНАРСТВО**

Кратак садржај – Рад се бави истраживањем понашања постојеће армиранобетонске просторне рамове конструкције, нерегуларне у основи, изложене сеизмичким дејствима, применом нелинеарне статичке сеизмичке „pushover“ анализе, при чему је нелинеарно понашање моделирано применом модела са концентрисаном пластичношћу, односно применом пластичних зглобова.

Abstract – This thesis researches behaviour of existing spatial reinforced concrete frame building, not regular in plan, subjected to seismic action, using nonlinear static seismic „pushover“ analysis, while nonlinearities are being modeled using concentrated plasticity model, the so called plastic hinges model.

Кључне речи: Грађевинарство, бетон, нелинеарност, сеизмика, рамови

1. УВОД

У подручјима где је ниво сеизмичког хазарда довољно велик да утицај земљотреса може да угрози објекте и животе људи у објектима, као једно од релевантних оптерећења за које треба одредити утицаје и димензионисати конструкцију су тзв. сеизмички утицаји, односно утицаји који потичу од деловања земљотреса на објекат. У зависности од типа конструктивног система, значаја самог објекта и његове локације, одређује се сеизмичко оптерећење. Правилним димензионисањем објекта и конструисањем детаља може се постићи задовољавајући ниво сеизмичке заштите. Формирање модела на којем се врше одговарајуће анализе игра значајну улогу у добијању резултата адекватне тачности за разумно време трајања прорачуна. Сложеност модела је функција методе (линеарна или нелинеарна, статичка или динамичка) којом ће се анализирати сеизмичко дејство, броја степени слободе, модела нелинеарног понашања материјала, интеракције конструкција-тло, итд.

Предмет овог рада биће нелинеарна статичка сеизмичка анализа, популарно названа pushover анализа. У овој методи се примењује статички нумерички модел и нелинеарно понашање материјала.

Она допушта анализу понашања конструкција у нелинеарном домену развојем како материјалне, тако и геометријске нелинеарности. Материјалну нелинеарност је могуће моделирати применом пластичних зглобова у зонама у којима се очекује појава пластификације попречних пресека и применом влакнастих модела којима је могуће узети у обзир развој нелинеарних деформација дуж штапова. Статички нумерички модел је модел са више степени слободе, док се за одређивање циљног померања користи систем са једним степеном слободе.

Како у домаћим прописима није имплементирана нелинеарна статичка сеизмичка анализа, у овом раду је коришћен пропис EN1998 у којем је имплементирана N2 метода.

Анализа се спроводи подвргавањем конструкције монотонно растућој шеми бочних сила, које представљају инерцијалне силе које би деловале на конструкцију приликом померања основе. Шема бочних сила се узима као равномерно расподељена, троугаона или се користи модална расподела генерисањем из одређеног својственог облика. Гравитациона оптерећења се одржавају константним. Како се бочно оптерећење повећава у инкрементима, конструкцијски елементи почињу да достижу границу развлачења. Са сваким новим елементом који достиже ову границу следи губитак крутости. На овај начин долази се до нелинеарне релације сила-померање (pushover крива) за MDOF систем. За случај зграда то су смичућа сила у основи и померање врха зграда као релевантне величине за силу и померање. Померање до којег желимо да пратимо понашање конструкције се зове циљно померање.

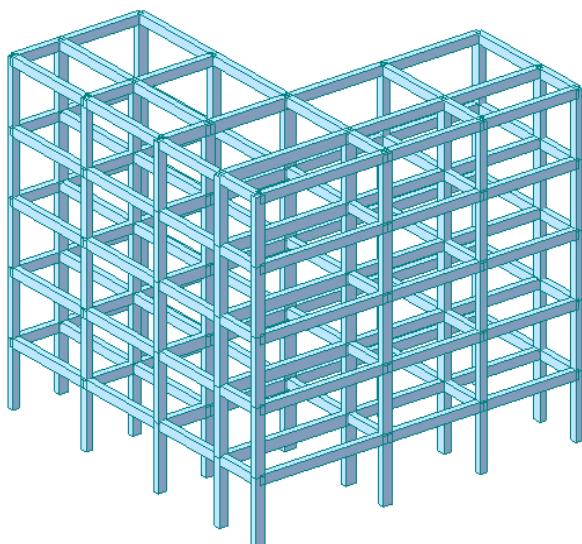
2. НУМЕРИЧКА АНАЛИЗА - 3D PUSHOVER АНАЛИЗА ПОСТОЈЕЋЕ ОКВИРНЕ АБ КОНСТРУКЦИЈЕ

Нумерички пример подразумева тродимензионалну (3D) нелинеарну статичку сеизмичку анализу постојеће (просторне) оквирне АБ конструкције. Моделирање и анализа конструкције са пластичним зглобовима је извршена у софтверу Midas GEN. На слици 1 дат је изометријски приказ анализираног објекта.

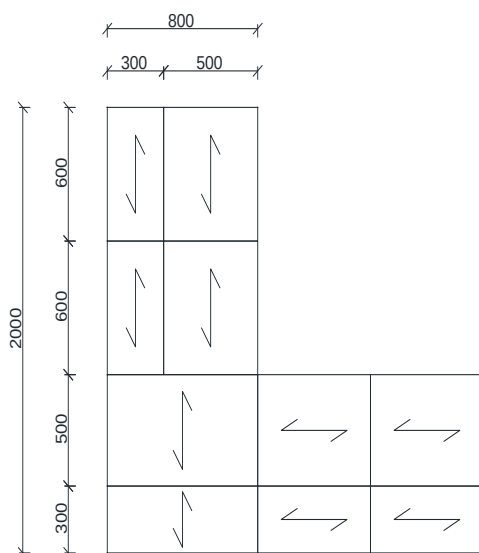
Конструкција је петоспратна АБ просторна оквирна конструкција основе облика слова Г. Са димензијама у основи према слици 2:

НАПОМЕНА:

Овај рад је произтекао из мастер рада чији ментор је био проф. др Ђорђе Лађиновић.



Слика 1: Модел за нумеричку анализу



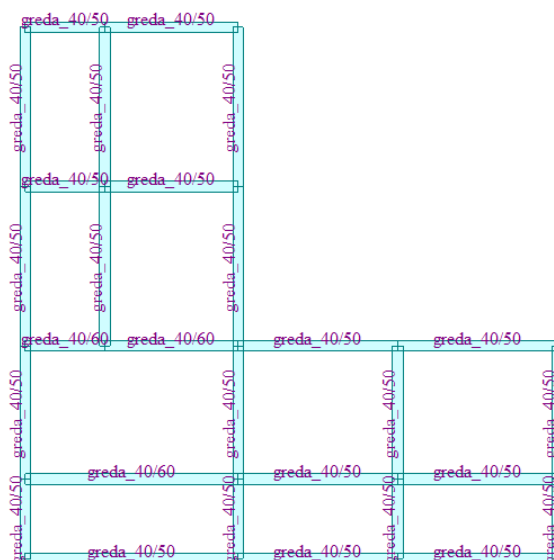
Слика 2: Основа објекта

Попречни пресеци стубова су 40/40, 40/55 и 40/70, а попречни пресеци греда 40/50 и 40/60. Међуспратне таванице су ситноребрасте таванице, при чему је стрелицама на слици 2 назначен правац у ком оне преносе оптерећење. Међуспратна конструкција није моделирана, али је узета у обзир њена сопствена тежина и пренета у правцу ношења, а обухваћена је и крутост таванице у сопственој равни, преко опције floor diaphragm.

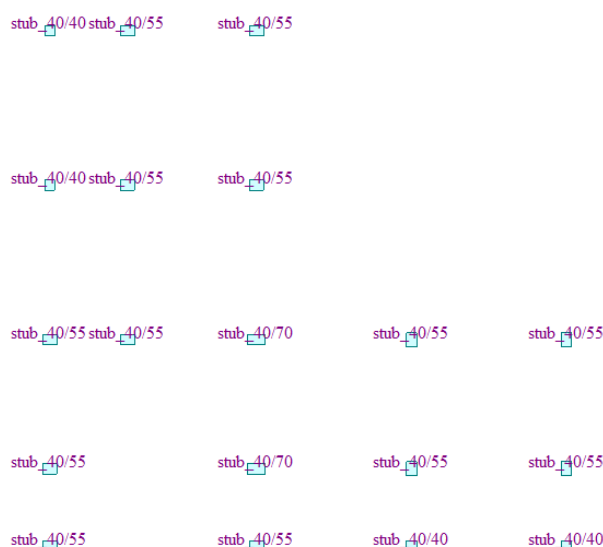
Као оптерећење на међуспратне таванице аплицирано је стално оптерећење од 6 kN/m^2 , корисно оптерећење од 2 kN/m^2 , а по ободу оптерећење од спољашњих зидова од 10 kN/m .

Примењени материјали на објекту су бетон класе C20/25, као и арматура класе А, са границом развлачења од 400 МПа.

У наставку на сликама 3 и 4 дат је приказ попречних пресека стубова и греда:



Слика 3: Попречни пресеци греда



Слика 4: Попречни пресеци стубова

Стубови попречног пресека 40/40 армирани су са подужном арматуром $12\text{Ø}16$ и узенгијама $\text{Ø}10/7.5$, стубови пресека 40/55 са подужном арматуром $16\text{Ø}16$ и узенгијама $\text{Ø}10/7.5$ и стубови пресека 40/70 са подужном арматуром $20\text{Ø}16$ и узенгијама $\text{Ø}10/7.5$.

Греде су армиране према три шеме армирања за греде попречног пресека 40/50, као и према две шеме за греде попречног пресека 40/60, са шипкама пречника 16 mm и узенгијама уз чворове $\text{Ø}8/5$.

2.1. Опис нумеричког модела

Модел за прелиминарну анализу је формиран коришћењем једнодимензионих гредних коначних елемената. Подрум није моделиран, јер као крута кутија није од интереса у овој анализи. Сви стубови су непокретно ослоњени и укљештени. Интеракција између оквира и испуне није узета у обзир. Нису моделиране круте зоне на месту пресека стубова и греда (у чворовима). Таванице су моделиране као круте у својој равни. Моделирање нелинеарног понашања је реализовано преко пластичних зглобова (концентрисана пластичност). Нису обухваћени $P - \Delta$ ефекти.

Формулисање пластичних зглобова према EN1998-3 се врши на основу кориснички дефинисане арматуре и карактеристика попречних пресека. Корисник задаје тзв. макро тип пластичног зглоба и присваја га на моделу одређеним чворовима, односно крајевима линијских елемената. За дефинисан макро тип пластичног зглоба, софтвер генерише одговарајуће карактеристике које има одговарајући пресек линијских елемената.

Овде су дефинисана два макро типа пластичних зглобова. Један за греде, а други за стубове. Макро тип пластичног зглоба за греде који не узима у обзир интеракцију аксијалних сила и момената савијања, а макро тип пластичног зглоба за стубове узима.

Формирано је 8 случајева оптерећења који се односе на нелинеарну статичку сеизмичку анализу (Pushover load case). За све случајеве оптерећења инкрементирање је контролисано померањима. $P - \Delta$ ефекти нису укључени у анализу. Узете су у обзир две шеме бочног оптерећења, равномерна и модална шема бочних сила, па тако имамо 8 случајева нелинеарне статичке сеизмичке анализе:

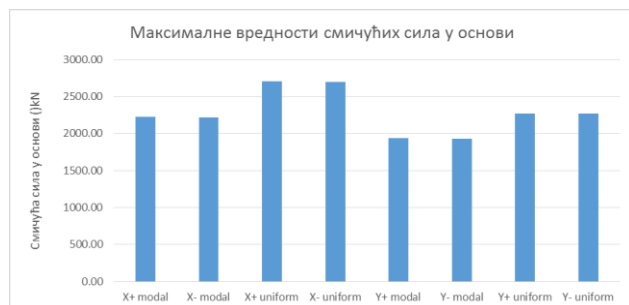
- $\pm X$ правац, равномерна шема бочних сила,
- $\pm Y$ правац, равномерна шема бочних сила,
- $\pm X$ правац, модална шема бочних сила
- $\pm Y$ правац, модална шема бочних сила

3. РЕЗУЛТАТИ АНАЛИЗЕ

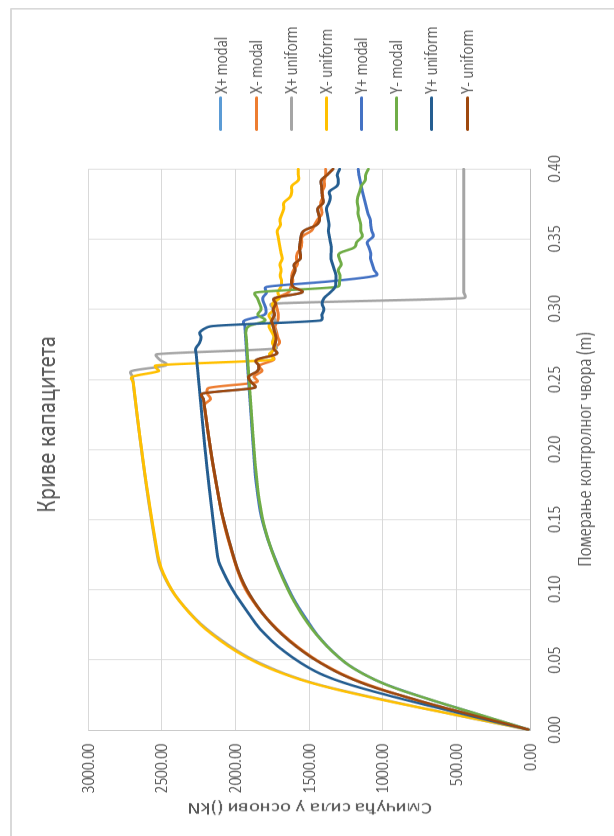
Након што су подешени сви параметри потребни за покретање анализе, она је извршена. Криве капацитета, као резултат ових анализа, за свих 8 случајева нелинеарних анализа су приказани на слици 6 како би се могли уочити односи између кривих.

Највећа смичућа сила у основи са вредношћу 2705.76 kN, за померање 0.26 m, се јавља за случај X+ uniform, а најмања смичућа сила у основи од 1930.95 kN, за померање 0.28 m, се јавља за случај Y- modal. Разлика између ове две вредности је око 40%.

На слици 5 јасно се уочава разлика између максимално достигнутих смичућих сила у основи за свих 8 случајева нелинеарних статичких сеизмичких анализа:

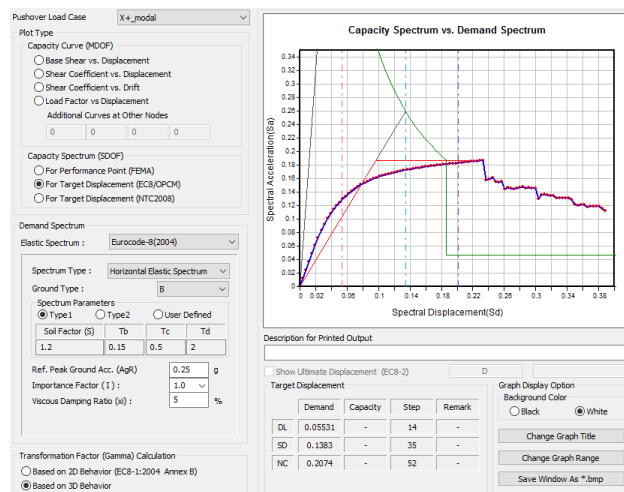


Слика 5: Максималне вредности смичућих сила у основи



Слика 6: Криве капацитета

Софтвер аутоматски срачунава циљна померања према EN1998, а на основу задатих параметара. У наставку се даје, на слици 7, прозор са унетим параметрима за један од случајева оптерећења (X+ modal), док је за остале случајеве задавање параметара учињено аналогно приказаном.



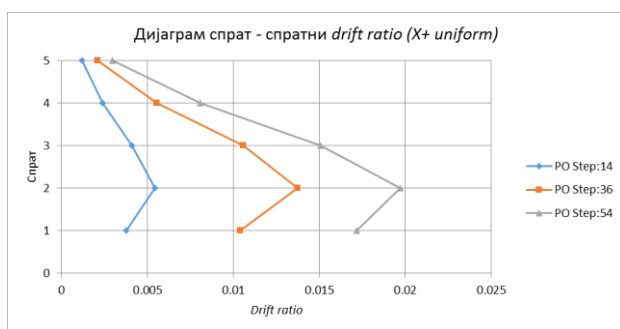
Слика 7: Прорачун циљног померања

Након што су срачуната циљна померања за све случајеве оптерећења, извршене су прорачунске контроле.

У програму Midas GEN постоји алат за аутоматску релаизацију прорачунских контрола. Прорачунске контроле се односе на поређење капацитета и захтева на локалном нивоу. Провере је потребно извршити за одговарајуће комбинације случајева оптерећења. Како постоји 8 случајева нелинеарне статичке сеизмичке анализе, при том није дозвољено међусобно комбиновати случајеве у којима су анализирани исти правци,

постоји 24 комбинације оптерећења. За сваки елемент потребно је извршити 4-8 провера, па је укупан број прорачунских провера које треба спровести једнак умношку броја комбинација (24), броја елемената (95 стубова и 140 греда) и броја провера (4-8), у зависности од тога да ли се проверавају и дуктилни и крти механизми (M_y, M_z и/или T_y, T_z), што укупно даје преко 23000 провера. Овај број провера превазилази могућности мануелног решавања, па је алат који је имплементиран у овом програму неопходан за решавање овог задатка. За проверу дуктилних механизма програм пореди обезбеђене капацитете са сеизмичким захтевима произашлих из pushover анализе за капацитет ротације, а за проверу кртих механизма се пореди капацитет трансверзалне силе коју пресек може да прими у еластичној фази рада са сеизмичким захтевом.

На слици 8 даје се дијаграм спратних дрефтова за случај оптерећења +X правац, равномерна шема бочних сила, за инкременте који одговарају стањима редом за, DL, SD и NC стање.



Слика 8: Спратни дрефтови

4. ЗАКЉУЧАК

Европске норме дозвољавају одређивање сеизмичког дејства на основу линеарно еластичног понашања конструкције применом мултимодалне анализе са спектрима одговора. Овакав прорачун конципиран је на еластичној анализи уз редукцију сеизмичких сила на основу процене захтеваног дуктилног одговора конструкције на глобалном нивоу. Према Еврокоду 8 дозвољава се примена и нелинеарне статичке и нелинеарне динамичке анализе. Нелинеарна статичка анализа се спроводи под константним гравитационим оптерећењем применом монотонно растућег хоризонталног оптерећења. За одређивање циљног померања користи се метода N2. Циљно померање представља сеизмички захтев. Процена сеизмичког одговора се врши на глобалном, спратном и локалном нивоу. Поређењем захтева и капацитета за одговарајуће гранично стање ротација тетива стубова и греда врше се провере на локалном нивоу.

На једном примеру просторне АБ оквирне конструкције је примењена нелинеарна статичка сеизмичка pushover анализа. Резултати анализа подразумевају криве капацитета, прорачунате захтеве и анализу понашање конструкције за одговарајућа стања.

Анализом су добијени и распореди пластичних зглобова, као и њихов редослед отварања. Такође добијени су и подаци о локалним захтевима на нивоу

елемената. Сви ови прорачунати капацитети су упоређени са захтевима аутоматском реализацијом провера. Анализе су урађене за две расподеле бочних сила, равномерну и модалну. Веће вредности смичућих сила у основи су добијене за равномерну расподелу, док су за модалну добијене мање. Такође, већа померања за максималне смичуће силе су добијене за модалну анализу.

Анализом пластичних механизма може се закључити да су механизми настали подвргавањем конструкције модалној шеми бочних сила склонији отварању пластичних зглобова прво у гредама, као и отказа одређеног броја греда, а затим у стубовима, да би колапс објекта подразумевао отказ свих стубова у њиховом дну. Док је за равномерну шему бочних сила карактеристично да је у почетку сеизмички захтев равномерно распоређен по свим стубовима, а да као први елемент отказује одређени број стубова, а да је за колапс карактеристично да отказују сви стубови у њиховом дну.

Иако има одређена ограничења, ова метода даје доста корисних података и даје увид у нелинеарно понашање конструкције за дејство земљотреса. На анализираном моделу би се у наредним радовима могла укључити и геометријски нелинеарна анализа, као и применити влакнасти уместо модела са пластичним зглобовима, како би се повећала поузданост резултата.

5. ЛИТЕРАТУРА

- [1] Chopra A. K.: Dynamics of structures, Theory and applications to earthquake engineering, Prentice Hall, New Jersey, 1995.
- [2] Fajfar P.: A nonlinear analysis method for performance based seismic design, Earthquake Spectra, Vol.16, No.3, pp.573-592, 2000.
- [3] Fardis N. M., Carvalho C. E., Fajfar P., Pecker A.: Seismic design of concrete buildings to Eurocode 8, CRC Press, 2015.

Кратка биографија:



Никола Рајић рођен је у Зрењанину 1992. год. Дипломирао је на Факултету техничких наука у Новом Саду 2015. године. Мастер рад на Факултету техничких наука из области Сеизмичка анализа конструкција брани октобра 2016. године.



проф. др Ђорђе Лађиновић рођен је у Шиду 1956. Дипломирао (1980.), магистрирао (1995.) и докторирао (2003.) на Факултету техничких наука у Новом Саду, а од 2012. је редовни професор на Департману за грађевинарство. Област интересовања му је Теорија конструкција.

KUĆNA AUTOMATIZACIJA ZASNOVANA NA RASPBERRY PI HOME AUTOMATION BASED ON RASPBERRY PI

Ivan Sušić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – MEHATRONIKA

Kratak sadržaj – Zadatak rada jeste realizacija programske podrške za Raspberry Pi i Atmega16 kao centralne procesorske uređaje kućne automatizacije. Raspberry Pi je predviđen za rad u bežičnoj mreži i da na taj način komunicira sa korisničkim interfejsom, a samim tim i sa korisnikom. Osim bežične komunikacije, Raspberry Pi je u sprezi (serijskom komunikacijom povezan) sa mikrokontrolerom ATmega16 kojem prosleđuje poruke koje je korisnik zadao, a zatim na osnovu tih poruka ATmega16 upravlja perifernim uređajima. Pored toga što korisnik može da zadaje komande perifernim uređajima, isto tako i periferni uređaji šalju informacije konroleru ATmega16 o trenutnim stanjima u kući, a on te informacije zatim dalje prosleđuje preko Raspberry Pi-a do korisnika kako bi korisnik imao informaciju o stanjima u objektu. Opisani je način i opisani su programski alati pomoću kojih se došlo do željenog rešenja.

Abstract – This paper is related to the issue of realization program support for Raspberry Pi and Atmega16 as central processor units. Raspberry Pi works in wireless network and in that way it communicate with user interface as well as with user. Except wireless communication, Raspberry Pi is connected (by serial communication) with microcontroller Atmega16 and transfer all commands given by user, so Atmega16 can control other peripheral devices. Besides, peripheral devices also sends information to Atmega16 so it forward them to the Raspberry Pi, so user can have clear picture about condition in his home environment. This paper describes how this project is done as well as program tools that were used.

Ključne reči: Kućna automatizacija, Raspberry Pi, Atmega16

1. UVOD

„Kućna automatizacija“ ili „pametna kuća“ je izraz koji se uobičajeno koristi da definiše stambeni prostor u kom se nalaze uređaji kao što su: osvetljenje, klima i TV uređaji, grejni sistemi, računari, sistem za nadzor itd. koji mogu međusobno da komuniciraju, a kojima se može upravljati vremenski, iz bilo koje prostorije u objektu, kao i putem interneta sa bilo koje lokacije u svetu.

Pametna kuća omogućava vlasniku stambenog prostora sistem kontrole i nadgledanja sa ciljem povećanja komfora, sigurnosti i smanjenja potrošnje energije, bilo da je vlasnik kod kuće ili je odsutan [1].

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Stevan Stankovski, red.prof.

Tokom razvoja ljudskog društva, čovek se trudio da sebi olakša upotrebu objekata u svom životnom prostoru. Posle otkrića i uvođenja električne energije u stambeni prostor, čovek je počeo i sa upotrebom kućnih aparata koji mu olakšavaju svakodnevni život. U današnje vreme sa napretkom tehnologije, čovek je uspeo da prilagodi upotrebu objekata u kući sopstvenim potrebama, a sve u cilju komfornijeg života.

Razvojem „pametnih uređaja“ kao što su računari i pametni telefoni, čovek je uspeo da stvori komunikaciju sa objektima u svom okruženju. Kao veoma dobra, pouzdana i za implementaciju najlakša tehnologija za komunikaciju između korisnika i perifernih uređaja (PU) kućne automatizacije je Wi-Fi tehnologija. Prednosti ove tehnologije su relativno mala veličina prijemnika i predajnika i postojanje gotovih softverskih modula što uvelike štedi vreme za implementaciju ovakve tehnologije.

Za spregu između korisnika i PU, koristi se RPi na kojem je pomoću web servera omogućeno pokretanje upravljačke aplikacije, slanje komandi i prikaz stanja i podataka sa PU i mikrokontroler Atmega16 (ATM16) koji prihvata komande od RPi-a, direktno upravlja PU i šalje podatke od PU ka RPi-u [2].

Ovo rešenje je upotrebljeno pre svega zbog brojnih mogućnosti ATM16 mikrokontrolera, koji pored analognih i digitalnih ulaza/izlaza poseduje i analognu/digitalnu konvertore, upravljanje širinskim modulacijama (PWM), kao i serijsku komunikaciju [3]. ATM16 je takođe iskorišćen kao čvorište, tako da PU mogu lakše međusobno komunicirati. Između kontrolera ATM16 i PU je ostvarena serijska komunikacija RS-485 [4] zbog svoje pouzdanosti i dalekog dometa.

Rad je podeljen u pet delova:

1. Teorijske osnove – Opisuje se konkretan primer kućne automatizacije, opisuju se RPi i ATM16 kao centralne jedinice sistema (CJS), data je njihova specifikacija, predstavljene su prednosti i mane, kao i razlozi korišćenja upravo ovih uređaja za sistem kućne automatizacije (SKA).
2. Koncept rešenja – Daje uvid u ideju pravljenja kućne automatizacije pomoću Raspberry Pi-a i ATM16.
3. Rešenje – Pruža detalje vezane za implementaciju predloženog rešenja. Daje uvid u hardversku i softversku implementaciju komponenti, kao i njihovu međusobnu komunikaciju.
4. Rezultati – Ispitivanje SKA, prikaz rada.
5. Zaključak – Iznete su prednosti ovakvog rešenja i data je ideja za poboljšanje i nadogradnju ovakvog rešenja.

2. TEORIJSKE OSNOVE

U ovom poglavlju detaljnije je opisan SKA, opisano je šta čini jedan takav sistem i zašto je kućna automatizacija pogodna za upotrebu. U ovom primeru akcenat je stavljen na RPi i ATM16 kao CJS, koje su odabrane kako zbog svojih dobrih osobina i mogućnosti, tako i zbog cene i lakoće implementacije. Sledeći su opisani PU koji su korišćeni u ovom SKA tj. opisano je upravljanje svetlima, zavesama i prozorima. Opisani su i senzori temperature, vlažnosti vazduha, detekcije prisustva, detekcije dima, RFID kao veličine čije se stanje menja, prati i treba da se na adekvatan način prikaže samom korisniku. Dati su konkretni primeri senzora koji po svojim osobinama zadovoljavaju potrebe jednog ovakvog sistema. Bežični komunikacioni protokol (Wi-Fi) je opisan sledeći i pomoću njega RPi komunicira sa korisnikom. Kao što je već navedeno, prednosti upotrebe ovakvog protokola su u pouzdanosti, dimenzijama prijemnika i predajnika, lakoći implementacije, kao i ceni. Odmah zatim je opisana serijska komunikacija kojom RPi komunicira sa ATM16 i kojom ATM16 komunicira sa PU, osobine ovakve komunikacije i razlog upotrebe baš nje. Poslednji je opisan web server, kao jedna od mogućnosti RPi-a, koji povezuje korisnika sa pametnim uređajima kućne automatizacije pomoću kojeg je omogućeno lakše upravljanje, kao i prikaz promene stanja.

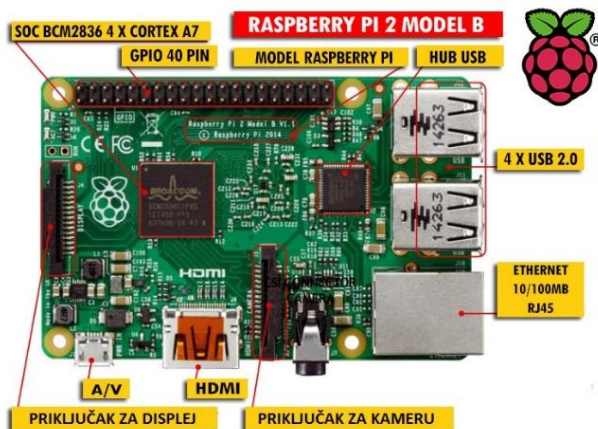
2.1 Sistem kućne automaizacije

Sistem kućne automatizacije se sastoji od različitih uređaja kojima korisnik može da upravlja i dobija informacije od njih, a to su:

1. Prekidači – omogućava uključivanje i isključivanje rasvete,
2. Senzori – temperature, vlažnosti vazduha, pokreta i dima,,
3. Uređaji za nadzor – kontrolisanje ulaska i izlaska iz objekta, sigurnosne kamere,
4. Aktuatori – kontrola zavesa i prozora.

2.2 Raspberry Pi

U ovom radu upotrebljen je RPi 2 model B, a njegov izgled prikazan je na Slici 3. Konkretno, on poseduje 900MHz ARM Cortex-A7 centralnu procesorsku jedinicu sa četiri jezgra i 1GB RAM memorije. Ovaj model je izabran kao pogodan za ovu upotrebu zbog njegove cene, brzine rada i broja priključaka kojima raspolaže. Raspberry Pi je prikzn na Slici 1.



Slika 1: Raspberry Pi 2 model B

2.3 Upravljanje svetlima

Razvojem pametnih uređaja kao i potrebe za kućnom automatizacijom, ljudi su došli na ideju da upravo pomoću ovih pametnih uređaja kontrolišu nivo osvetljenja u svom stambenom prostoru. Najveća prednost ovakvog upravljanja u odnosu na klasično prekidačko (ručno) upravljanje je to što se jednom ili grupom rasvetnih tela može upravljati jednim dodirnom (klikom) na aplikaciji na pametnom uređaju. Ideja upravljanja rasvetom u ovom radu je povezivanje elektronske ploče sa relejom na ATM16, a samim tim i na RPi. Upravljanje rasvetom se vrši preko web pretraživača na kom je interaktivna aplikacija. Pored upravljanja postoji i povratna informacija o promeni stanja ukoliko više korisnika upravlja svetlom u isto vreme. Na ovaj način svaki korisnik zna trenutno stanje objekta kojim upravlja.

2.4 Upravljanje svetlima

Kao bitan deo kućne automatizacije je i upravljanje prozorima i zavesama. Ovi PU imaju zadatak da korisnika koji je u objektu štite od spoljašnjih vremenskih prilika. Razvojem tehnologije, omogućeno je da i ovi predmeti budu automatizovani, jer je puno jednostavnije rukovati njima pomoću jednog klika na pametnom uređaju, nego ručno. Njihova automatizacija se sastoji u tome da se pomoću dodatne elektronike prvenstveno omogući pokretanje prozora i zavesa. Oni će biti povezani na servo motor koji će da omogući kretanje i da imaju više unapred definisanih položaja. Kako ima više mogućih stanja, motor mora da bude povezan na kontroler koji će kako da zadaje željeno stanje, tako i da očitava trenutno stanje u kom se ovi predmeti nalaze. Posle uspešnog izvršavanja komandi, korisnik dobija povratnu informaciju o tome kao i o trenutnoj poziciji u kojoj se prozori i/ili zavesa nalaze.

2.5 Upravljanje prozorima i zavesama

Kao bitan deo kućne automatizacije je i upravljanje prozorima i zavesama. Ovi PU imaju zadatak da korisnika koji je u objektu štite od spoljašnjih vremenskih prilika. Razvojem tehnologije, omogućeno je da i ovi predmeti budu automatizovani, jer je puno jednostavnije rukovati njima pomoću jednog klika na pametnom uređaju, nego ručno. Njihova automatizacija se sastoji u tome da se pomoću dodatne elektronike prvenstveno omogući pokretanje prozora i zavesa. Oni će biti povezani na servo motor koji će da omogući kretanje i da imaju više unapred definisanih položaja. Kako ima više mogućih stanja, motor mora da bude povezan na kontroler koji će kako da zadaje željeno stanje, tako i da očitava trenutno stanje u kom se ovi predmeti nalaze. Posle uspešnog izvršavanja komandi, korisnik dobija povratnu informaciju o tome kao i o trenutnoj poziciji u kojoj se prozori i/ili zavesa nalaze.

2.6 Senzor temperature, vlažnosti vazduha, dima i detekcije prisustva

Pošto su mikrokontroleri digitalne komponente javila se potreba za senzorima temperature koji će umesto analognog davati digitalan signal. Iako može da daje digitalni signal, bolju praktičnu primenu u ovom SKA će da ima analogni senzor koji će da bude povezan na neki od analognih ulaza ATM16 (A0-A7) koji u sebi sadrže A/D konvertor na svakom od ovih ulaza. Na ovaj način će se analogne vrednost očitavanja sa temperaturnog senzora automatski konvertovati i biće pogodnija za dalju obradu kao i za slanje i prikazivanje vrednosti temperature korisniku.

Kontrolisanje i praćenje vlažnosti vazduha je od izuzetne važnosti kako za industrijske tako i za kućne prilike. Ovaj senzor će da dopunjuje kućnu automatizaciju, a kao i senzor temperature i on će da bude povezan na neki od analognih ulaza pinova kontrolera ATM16 (A0-A7).

Senzori za detekciju pokreta su relativno mali, jeftini, male potrošnje i jednostavni za upotrebu. Zbog ovih razloga oni su našli primenu u aplikacijama u kućnim ili poslovnim okruženjima. PIR senzori su u osnovi napravljeni od piroelektričnih senzora koji mogu da detektuju različite nivoe infracrvenog zračenja.

Detektori gasova su uređaji koji detektuju prisustvo različitih gasova u nekom prostoru, najčešće funkcionišu kao deo nekog sigurnosnog sistema. Ovakvi uređaji treba da detektuju curenje gasa i o tome obaveste upravljački sistem i/ili da aktiviraju alarm, jer postoje mnogi gasovi koji su zapaljivi, eksplozivni ili otrovni i opasni su kako za čoveka tako i za njegovu okolinu.

2.7 RFID

RFID je sistem daljinskog slanja i prijema podataka pomoću RFID tag-a. RFID tag je relativno mali objekat koji se može zalepiti ili ugraditi u željeni proizvod. RFID tag sadrži u sebi antenu koja omogućava prijem i slanje radio-talasa od RFID primopredajnika [5]. U SKA, RFID će se koristiti za kontrolu ulaska/izlaska iz objekta. Svaki korisnik će da ima posebno svoj lični tag pomoću kojeg će da se identifikuje. Na ovaj način, u svakom trenutku će se znati ko je poslednji ušao/izašao iz objekta i ujedno obezbediti veća sigurnost samog objekta.

2.8 Komunikacioni protokoli

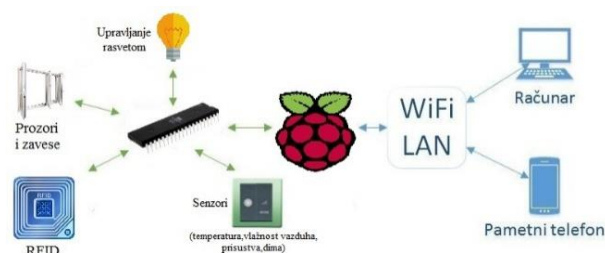
Komunikacioni protokol predstavljaju formalne opise digitalnih poruka i njihovih pravila [6]. Širok spektar komunikacionih mreža sa svojim raznovrsnim primenama predstavljaju veliki potencijal, većina njih je veoma jednostavna za instalaciju u raznim uslovima, a upravo ta karakteristika ih čini naročito pogodnim za samostalne (Embedded) sisteme gde predstavljaju interfejs između čoveka i njegove okoline.

U industriji kao i u svakodnevnom životu, postoji ogromna potreba za umrežavanje raznih komponenti sistema i njihovom komunikacijom.

Zato su se vremenom pojavili različiti komunikacioni protokoli. U nastavku će biti prikazani protokoli koji se koriste u ovom primeru kućne automatizacije, a to su Wi-Fi komunikacija i serijska komunikacija. Ovi protokoli su se kao što je već navedeno izdvojili po tome što su laki za implementaciju, što ne zahtevaju preveliki dodatni hardver i zbog svoje cene.

3. KONCEPT REŠENJA

Arhitektura rešenja sastoji se od upotrebe RPi-a i ATM16 kao CUI, serijske komunikacije između njih, a zatim i spajanja PU na ATM16 serijskom komunikacijom. Cilj je uz minimalnu cenu proširiti spektar mogućnosti samog hardvera bez narušavanja postojećih funkcionalnosti RPi-a kao i celokupnog sistema kućne automatizacije. Posle zadavanja komandi PU korisnik automatski dobija odgovor o uspešnosti zadate komande, kao i o prikazu novog stanja u kom se PU nalazi.



Slika 2: Koncept rešenja kućne automatizacije

4. PROGRAMSKO REŠENJA

U ovom poglavlju prikazani su programski alati i razvojno okruženje koja su korištena u daljem radu i ostvarenju idejnog rešenja.

4.1 Implementacija perifernih uređaja sa kontrolerom

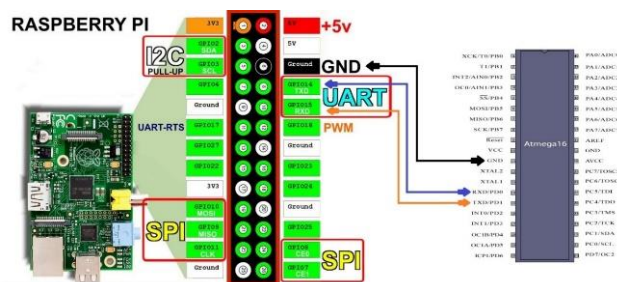
Za izradu zadatka korišteno je razvojno okruženje Easy AVR6 firme Mikroelektronika.

U razvojnom okruženju CodeVisionAVR je potrebno pokrenuti CodeWizardAV, tu je potrebno odabrati tip kontrolera gde od ponuđenog je neophodno selektovati AT90, ATtiny, Atmega, FPSLIC. Nakon toga otvara se prozor sa tabom Chip gde treba izabrati kontroler Atmega16, zatim podesiti Clock odnosno frekvenciju eksternog oscilatora koji se nalazi na razvojnom okruženju.

Jednostavnost programiranja mikrokontrolera u CodeVision i veliki broj unapred definisanih biblioteka bili su ključni u odabiru baš ovog programa za pisanje koda kako za serijsku komunikaciju, tako i za komunikaciju ATM16 sa PU.

4.2 Implementacija komunikacije RPi-a i ATM16

Kako bi ova dva uređaja u sistemu kućne automatizacije mogla da komuniciraju i razmenjuju podatke, neophodno je između njih uspostaviti vezu, kako hardversku, tako i softversku.



Slika 3: Hardverska komunikacija između RPi-a i ATM16

Druga neophodna stvar koja se mora uraditi na RPi-ju je softversko omogućavanje serijske komunikacije. Da bi se serijska komunikacija omogućila, neophodno je da logovanje bude isključeno, što se radi pozivom komande `sudo raspi-config`. Zatim se odabere opcija *Advance Options*, zatim na *Serial*, zatim kliknuti na *Yes* i posle toga na *Finish*. Kada se na samom RPi-ju omogući serijska komunikacija, potrebno je restartovati RPi, a zatim proveriti da li je serijska komunikacija zaista omogućena ponovnim pisanjem komande `dmesg | grep tty`.

4.3 Pokretanje Raspberry Pi u željenom režimu rada

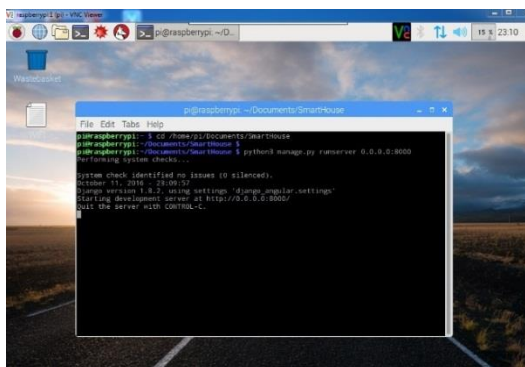
Kako bi korisnik mogao da komunicira sa svim PU i isto tako svi periferni uređaji mogli da komuniciraju sa korisnikom, neophodno je na RPi-ju instalirati web server.

U ovom slučaju, instaliran je Nginx web server koji kao takav može da podrži TCP, UDP, HTTP kao i mnoge druge protokole [7]. Nginx se instalira jednostavno, na način da se u komandnom prozoru ukucaju sledeće naredbe: `sudo apt-get update`, a posle nje i `sudo apt-get install nginx`.

U cilju ostvarivanja samo komunikacije sa ATM16, u programskom jeziku Python, koji je kao što je rečeno sastavni deo Raspbian Jessie verzije Linux operativnog sistema, je napisan program koji ovo upravo omogućuje. U isto vreme je Python upotrebljen i za pravljenje takozvanog backend dela aplikacije, i to je takođe jedna od prednosti upotrebe ovog programskog jezika.

Pored ovog programskog jezika kao osnovne platforme, u ovom rešenju upotrebljen je i Django. Django je Python web framework koji se koristi za web aplikacije i pogodan za rad sa Python aplikacijama. Kako bi se pokrenula aplikacija, neophodno je u komandnom prozoru navesti prvo direktorijum u kom se program nalazi, a zatim ukucati sledeću naredbu:

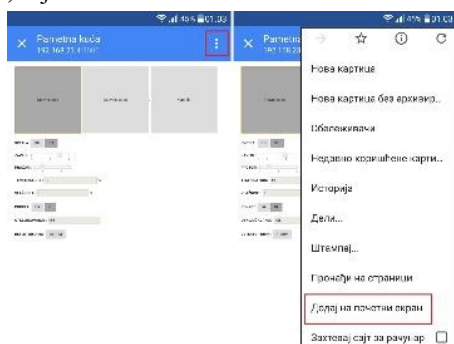
```
python3 manage.py runserver 0.0.0.0:8000
```



Slika 4: Pokretanje web servera

4.4 Implementacija korisničkog interfejsa

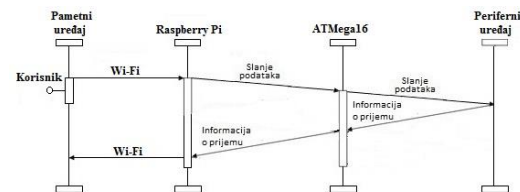
Da bi korisnik mogao da ima komunikaciju sa PU, neophodno je da ima odgovarajući korisnički interfejs koji će da bude prilagođen upravo korisnikovom stambenom prostoru u kojem se nalazi kućna automatizacija. Da bi ovo bilo omogućeno, neophodno je napraviti odgovarajuću interaktivnu web stranicu. Za ove potrebe je iskorišćen Angular, zajedno sa HTML-om i CSS-om.



Slika 5: Izgled web aplikacije

5. KOMUNIKACIJA

Komunikacija između korisnika i RPi je ostvarena bežičnim putem, dok je komunikacija od RPi ja preko ATM16 do PU ostvarena putem serijske komunikacije.



Slika 6: Prikaz komunikacija u sistemu kućne automatizacije

6. REZULTATI

U ovom poglavlju je pokazano da sistem prikazan u radu funkcioniše na željeni način.

Tabela 1: Prikaz testova sa rezultatima

Korišćenje interaktivne web stranice	Test uspešno prošao
Ispravnost poslanih poruka	Test uspešno prošao
Pokretanje web servera zajedno sa korisničkim interfejsom	Test uspešno prošao
Prihvatanje poruke koje RPi primi od pametnih uređaja	Test uspešno prošao
Ispravnost poruka koje RPi šalje ATM16	Test uspešno prošao
Simulacija ATM16	Test uspešno prošao
Ponovo slanje podataka usled greške	Test uspešno prošao
Uspešnost slanja podataka od PU do RPi-a	Test uspešno prošao
Vreme neophodno za slanje podataka	Test uspešno prošao

7. ZAKLJUČAK

Prednost ovog rešenja se ogleda u korišćenju RPi-ja koji je veoma jeftin, relativno jednostavan za upotrebu i za programiranje lak računar koji je u ovom slučaju napravljen za kontrolu i nadzor krajnjih korisničkih uređaja u kući. Atmega16 kao kontroler zbog svojih analognih ulaza i PWM-a, samo nadopunjuje ovo rešenje i omogućava lakšu kontrolu PU, kao i lakše očitavanje sa senzora.

Naredno unapređenje sistema se sastoji od proširenja sistema sa mogućnosti kontrole većeg broja PU. Drugi pravac poboljšanja je taj da se omogući upravljanje PU preko interneta kada je korisnik i van objekta.

Drugi pravac poboljšanja korisničkog interfejsa bi se ogledao u tome da se on prilagodi slepim i slabovidim osobama ili osobama sa hendikepom.

8. LITERATURA

- [1] <http://www.smarthomeusa.com/smarthome/>
- [2] <https://www.raspberrypi.org/>
- [3] <http://www.atmel.com/images/doc2466.pdf>
- [4] <http://www.rs485.com/rs485spec.html>
- [5] <http://www.rfidjournal.com/site/faqs#Anchor-What-363>
- [6] <https://www.techopedia.com/definition/25705/communication-protocol>
- [7] <https://www.nginx.com/>

Kratka biografija:



Ivan Sušić iz Šida, rođen je u Sremskoj Mitrovici 1992. god. Upisuje osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu 2011. godine, na smeru Mehatronika. Master akademske studije na Fakultetu tehničkih nauka upisuje 2015. godine, na smeru Mehatronika, robotika i automatizacija. Master rad odbranio je 2016 godine.

U realizaciji Zbornika radova Fakulteta tehničkih nauka u toku 2015. godine učestvovali su sledeći recenzenti:

Aco Antić	Duško Bekut	Milan Rackov	Slavko Đurić
Aleksandar Erdeljan	Đorđe Ćosić	Milan Rapajić	Slobodan Dudić
Aleksandar Ristić	Đorđe Lađinović	Milan Simeunović	Slobodan Krnjetin
Bato Kamberović	Đorđe Obradović	Milan Trifković	Slobodan Morača
Biljana Njegovan	Đorđe Vukelić	Milan Trivunić	Sonja Ristić
Bogdan Kuzmanović	Đura Oros	Milan Vidaković	Srđan Kolaković
Bojan Batinić	Đurđica Stojanović	Milena Krklješ	Srđan Popov
Bojan Lalić	Emil Šećerov	Milica Kostreš	Srđan Vukmirović
Bojan Tepavčević	Filip Kulić	Milica Miličić	Staniša Dautović
Bojana Beronja	Goran Sladić	Milinko Vasić	Stevan Milisavljević
Branislav Atlagić	Goran Švenda	Miloš Slankamenac	Stevan Stankovski
Branislav Nerandžić	Gordana	Miloš Živanov	Strahil Gušavac
Branislav Veselinov	Milosavljević	Milovan Lazarević	Svetlana Nikoličić
Branislava Kostić	Gordana Ostojić	Miodrag Hadžistević	Tanja Kočetov
Branislava	Igor Budak	Miodrag Zuković	Tatjana Lončar
Novaković	Igor Dejanović	Mirjana Damjanović	Turukalo
Branka Nakomčić	Igor Karlović	Mirjana Malešev	Todor Bačkalić
Branko Milosavljević	Ilija Kovačević	Mirjana Radeka	Toša Ninkov
Branko Škorić	Ivan Beker	Mirko Borisov	Uroš Nedeljković
Cvijan Krsmanović	Ivan Župunski	Miro Govedarica	Valentina Basarić
Damir Đaković	Ivana Katić	Miroslav Hajduković	Velimir Čongradec
Danijela Lalić	Ivana Kovačić	Miroslav Plančak	Velimir Todić
Darko Čapko	Jasmina Dražić	Miroslav Popović	Veljko Malbaša
Darko Marčetić	Jelena Atanacković	Mitar Jcanović	Veran Vasić
Darko Reba	Jeličić	Mladen Kovačević	Veselin Avdalović
Dejan Ubavin	Jelena Borocki	Mladen Radišić	Veselin Perović
Dragan Ivanović	Jelena Kiurski	Momčilo Kujačić	Vladan Radlovački
Dragan Ivetić	Jelena kovačević	Nađa Kurtović	Vladimir Katić
Dragan Jovanović	Jureša	Nebojša Pjevalica	Vladimir Radenković
Dragan Kukolj	Jelena Radonić	Neda Pekarić Nađ	Vladimir Strezoski
Dragan Mrkšić	Jovan Petrović	Nemanja	Vladimir Škiljajica
Dragan Pejić	Jovan Tepić	Stanisavljević	Vlado Delić
Dragan Šešlija	Jovan Vladić	Nenad Katić	Vlastimir
Dragana Bajić	Jovanka Pantović	Nikola Brkljač	Radonjanin
Dragana	Karl Mičkei	Nikola Đurić	Vuk Bogdanović
Konstantinović	Katarina Gerić	Nikola Jorgovanović	Zdravko Tešić
Dragana Šarac	Ksenija Hiel	Nikola Radaković	Zoran Anišić
Dragana Štrbac	Laslo Nađ	Ninoslav Zuber	Zoran Brujic
Dragi Radomirović	Leposava Grubić	Ognjen Lužanin	Zoran Jeličić
Dragiša Vilotić	Nešić	Pavel Kovač	Zoran Mijatović
Dragoljub Novaković	Livija Cvetičanin	Peđa Atanasković	Zoran Milojević
Dragoljub Šević	Ljiljana Vukajlov	Petar Malešev	Zoran Mitrović
Dubravka Bojanić	Ljiljana Cvetković	Predrag Šiđanin	Zoran Papić
Dušan Dobromirov	Ljubica Duđak	Radivoje Rinulović	Željen Trpovski
Dušan Gvozdenac	Maja Turk Sekulić	Rado Maksimović	Željko Jakšić
Dušan Kovačević	Maša Bukurov	Radovan Štulić	
Dušan Sakulski	Matija Stipić	Rastislav Šostakov	
Dušan Uzelac	Milan Kovačević	Slavica Mitrović	