UDC: 004 Info M str. 33-37

GENERISANJE POSLOVNE LOGIKE SOFTVERSKOG SISTEMA U NASTAVNOM PROCESU
GENERATING THE BUSINESS LOGIC OF SOFTWARE SYSTEMS IN THE EDUCATIONAL PROCESS

Ana Trujić, Miloš Milić, Dušan Savić, Ilija Antović, Siniša Vlajić

REZIME: Tokom izrade seminarskih radova na predmetu Projektovanje softvera, koji se izvodi na Fakultetu organizacionih nauka Univerziteta u Beogradu, studenti često prave sintaksne i semantičke greške u fazama prikupljanja korisničkih zahteva i analize. Greške koje studenti prave otežavaju dalji razvoj njihovih softverskih sistema, a nastavnici više vremena provode u otklanjanju tih grešaka, nego u diskusiji o konkretnim domenima. Kako bi se unapredio proces izrade dokumentacije razvijen je SILAB generator. SILAB generator je alat pomoću kojeg se automatizuje proces definisanja poslovne logike softverskog sistema. U okviru rada objašnjen je način upotrebe SILAB generatora u radu sa studentima, kao i prednosti i ograničenja predloženog pristupa u nastavi. Osnova SILAB generatora su metamodeli, odnosno unapred definisani šabloni konceptualnih modela, koji usmeravaju studente kroz proces definisanja domenskih koncepata, relacija između njih i slučajeva korišćenja. Na osnovu unetih podataka, generator generiše dokument koji sadrži prve dve faze SILAB metode, faze prikupljanja korisničkih zahteva i analize. Prema iskustvu nastavnika, primenom ovog alata česte sintaksne i semantičke greške koje su studenti pravili su se smanjile. Takođe, postignut je uniforman izgled seminarskih radova.
KLJUČNE REČI: SILAB generator, automatizacija razvoja dokumentacije, projektovanje softvera, poslovna logika.
ABSTRACT: In the Software Design course at the Faculty of Organizational Sciences, University of Belgrade, students often make syntax and semantic errors during the preparation of seminar papers, particularly in the requirements gathering and analysis phases. These errors make further development of their software systems more difficult, while teachers spend more time correcting them than discussing specific problem domains. To improve the documentation development process, the SILAB generator was developed. The SILAB generator is a tool used to automate the process of defining the business logic of a software system. This paper describes the use of the SILAB generator in teaching, as well as the advantages and limitations of the proposed approach. The generator is based on metamodels, i.e., predefined conceptual model templates, which guide students in defining domain concepts, relationships between them, and use cases. Based on the provided inputs, the generator produces a document that includes the first two phases of the SILAB method, namely requirements gathering and analysis. The application of this tool reduced common syntax and semantic errors made by students and resulted in a more uniform structure of seminar papers.
KEY WORDS: SILAB generator, documentation automation, software design, business logic.

1. UVOD

Proces razvoja softverskih sistema sastoji se od više faza: prikupljanje korisničkih zahteva, analiza, projektovanje, implementacija, testiranje i održavanje (Vlajić, 2025; Hossain, 2023; Alazzawi, Yas & Rahmatullah, 2023; Acharya & Sahu, 2020). Ove faze predstavljaju elemente životnog ciklusa razvoja softvera (Software Development Life Cycle - SDLC) i sastavni su deo različitih metoda razvoja softverskih sistema (Kumar & Bhatia, 2014; Munassar & Govardhan, 2010; Boehm, 1988). Najčešće korišćeni modeli razvoja softvera su model vodopada, spiralni model i iterativno-inkrementalni model. Razvoj ovih modela doveo je do pojave agilnih metoda, poput ekstremnog programiranja, scrum-a, jedinstvenog procesa razvoja softvera (Unified Software Development Process) i Larmanove metode razvoja softvera (Larman, 1997; Beck, 2000; Jacobson, Booch & Rumbaugh, 1999; Schwaber & Beedle, 2002; Zhang & Patel, 2011).

U okviru predmeta Projektovanje softvera, koji se izvodi na Fakultetu organizacionih nauka Univerziteta u Beogradu, studenti razvijaju svoje softverske sisteme koristeći SILAB metodu razvoja softvera (Vlajić, 2025). SILAB metoda razvoja softvera obuhvata faze prikupljanja korisničkih zahteva, analize, projektovanja, implementacije i testiranja. Zasniva se na postepenom razvoju sistema kroz više iteracija i primeni modela u procesu razvoja softvera. Obavezan deo ispita na ovom predmetu predstavlja izrada dokumentacije koja prati proces razvoja softverskog rešenja, kroz sve faze SILAB metode. Na početku semestra organizuju se dodatni termini tokom kojih nastavnici rade sa studentima, sa ciljem da im pomognu u izradi prve dve faze seminarskog rada i ukažu na greške u dokumentaciji. Tokom višegodišnjeg rada sa studentima, nastavnici su uočili da se u početnim fazama razvoja softvera (prikupljanje korisničkih zahteva i analiza) ponavljaju slične sintaksne i semantičke greške. Ove greške mogu uticati na kvalitet narednih faza razvoja softvera, a nastavnici puno vremena tokom konsultacija utroše na njihovo ispravljanje, umesto na razmatranje samih domena različitih tema seminarskih radova sa studentima. U cilju rešavanja uočenih nedostataka u nastavi na predmetu Projektovanje softvera razvijen je SILAB generator.

SILAB generator je alat koji omogućava automatizaciju definisanja poslovne logike softverskog sistema u nastavnom procesu. Generator sadrži unapred definisane metamodele, koje studenti koriste kako bi definisali svoje konceptualne modele i funkcionalnosti sistema kroz slučajeve korišćenja. Kroz jasno definisane korake, studenti generišu dokumentaciju za prve dve faze SILAB metode razvoja softvera. Primenom SILAB generatora koristi se strategija razvoja zasnovana na modelima, jer metamodel predstavlja osnovu za definisa-

nje strukture sistema. Korišćenjem ovog alata, studentima je omogućeno da prate strukturiran proces izrade dokumentacije, pri čemu se smanjuje mogućnost pojave grešaka. Takođe, ovakvim pristupom omogućena je jednostavna izmena dokumentacije, kroz promenu ulaznih podataka i ponovno generisanje dokumentacije.

Nakon uvodnog poglavlja, u drugom poglavlju ovog rada objašnjena je poslovna logika softverskog sistema. Treće poglavlje sadrži opis SILAB generatora, dok je u četvrtom poglavlju dat studijski primer. U petom poglavlju objašnjene su prednosti primene SILAB generatora, kao i njegova ograničenja. U šestom poglavlju data su zaključna razmatranja ovog rada.

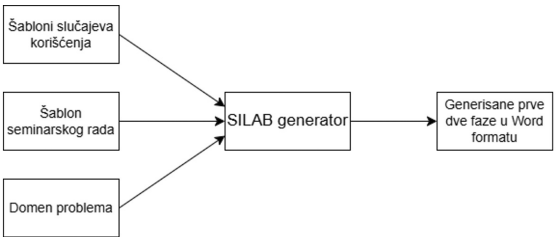
Početni rezultati u vezi sa SILAB generatorom su predstavljeni na XV Skupu privrednika i naučnika SPIN 2025 u formi kratkog rada (Korunović et al., 2025).

2. POSLOVNA LOGIKA SOFTVERSKOG SISTEMA

Poslovna logika opisuje način funkcionisanja softverskog sistema, kroz definisanje koncepata, njihovih međusobnih odnosa, funkcionalnosti sistema i ograničenja pod kojima se mogu izvršavati te operacije. U okviru SILAB metode razvoja softvera, poslovna logika nastaje kao rezultat faze analize i predstavlja osnovu za naredne faze razvoja softvera (projektovanje, implementacija i testiranje) (Vlajić, 2025). Poslovna logika obuhvata logičku strukturu i ponašanje softverskog sistema. Struktura softverskog sistema se opisuje preko konceptualnog i relacionog modela, dok se ponašanje definiše preko sistemskih dijagrama sekvenci i sistemskih operacija (Vlajić, 2025). Konceptualni model prikazuje koncepte sistema i veze između njih, dok dijagrami sekvenci i sistemske operacije definišu na koji način entiteti sistema međusobno sarađuju da bi ispunili korisničke zahteve i funkcionalnosti sistema. Poslovna logika se definiše na osnovu informacija prikupljenih tokom faze prikupljanja korisničkih zahteva, odnosno verbalnog opisa i liste slučajeva korišćenja. Konceptualni i relacioni model nastaju na osnovu koncepata definisanih u slučajevima korišćenja, dok se sistemske operacije i dijagrami sekvenci izvode iz scenarija slučajeva korišćenja, odnosno definisanih funkcionalnosti sistema. Upravo zbog navedene povezanosti između slučajeva korišćenja, sistemskih operacija, konceptualnog modela i relacionog modela, prve dve faze seminarskog rada su pogodne za automatsko generisanje.

3. SILAB GENERATOR

SILAB generator je razvijen sa ciljem da studentima pomogne u izradi dokumentacije za prve dve faze razvoja softverskog sistema primenom SILAB metode razvoja softvera. Generator je implementiran u Microsoft Access razvojnom okruženju, korišćenjem programskog jezika Visual Basic for Applications (VBA). U radu je predstavljena verzija 2.9 SILAB generatora, koja je primenjivana u radu sa studentima tokom zimskog semestra, školske 2025/26 godine. Na Slici 1. prikazani su ulazi i izlaz iz SILAB generatora.



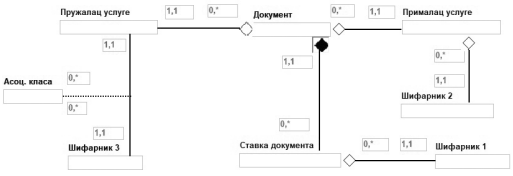
Slika 1: Ulazi i izlaz iz SILAB generatora

Ulazni podaci u SILAB generator čine šabloni slučajeva korišćenja, šablon seminarskog rada i specifikacija domena problema. Uloga šablona jeste da obezbedi generisanje konzistentne i sintaksno i semantički ispravne dokumentacije za prve dve faze seminarskog rada.

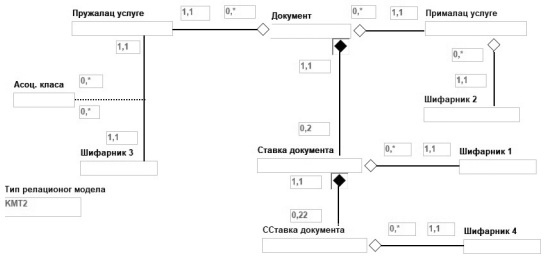
Šablon slučajeva korišćenja predstavlja Word dokument koji sadrži osnovne funkcionalnosti sistema, kao što su kreiraj, pretraži, promeni, obriši, ubaci i prijavi. Za svaki od slučajeva korišćenja definisani su naziv, aktori, učesnici, preduslovi, osnovni i alternativni scenariji. Šablon se sastoji od generičkih delova teksta, koji nisu vezani za konkretan domen i specifičnih delova koji se tokom generisanja zamenjuju konceptima iz domenskog modela. Na ovaj način automatski se generišu sintaksno i semantički ispravni slučajevi korišćenja, kojima su definisane funkcionalnosti sistema.

Pored šablona slučajeva korišćenja, generator koristi i šablon seminarskog rada kojim je definisana struktura dokumentacije. Šablon obuhvata naslovnu stranu, sadržaj, uvod, poglavlja koja odgovaraju fazama SILAB metode razvoja softvera, zaključak i literaturu. Uloga ovog šablona je da obezbedi standardizovanu i dobro formatiranu strukturu dokumenta, koja se tokom generisanja popunjava podacima koji su specifični za domen.

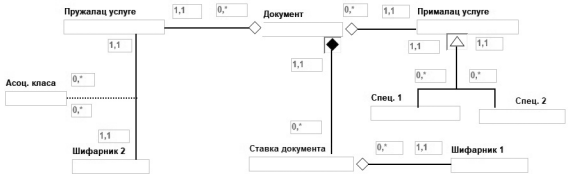
Za popunjavanje prethodno navedenih šablona potrebno je uneti dve grupe podataka: (I) osnovne informacije o studentu i seminarskom radu i (II) podatke koji opisuju domen (seminarskog rada). Osnovne informacije obuhvataju naziv teme, ime i prezime mentora, ime i prezime studenta, broj indeksa i akademsku godinu. Ovi podaci se koriste za generisanje naslovne strane seminarskog rada. Podaci koji se odnose na domen koriste se za generisanje poslovne logike softverskog sistema. Osnovni ulaz SILAB generatora je metamodel, pri čemu studenti biraju jedan od tri ponuđena metamodela: KMT1, KMT2 ili KMT3. Metamodeli predstavljaju šablone konceptualnih modela koji studentima predlažu strukturu konceptualnog modela. Svaki metamodel sadrži skup unapred definisanih metaklasa (primalac usluge, pružalac usluge, dokument, stavka dokumenta, šifarnici i asocijativna klasa) i njihove međusobne veze definisane preko kardinalnosti preslikavanja. Na slikama 2, 3 i 4 prikazani su metamodeli KMT1, KMT2 i KMT3.



Slika 2: Metamodel KMT1



Slika 3: Metamodel KMT2



Slika 4: Metamodel KMT3

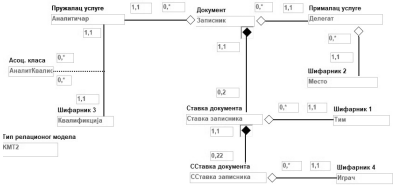
Studenti izabrani metamodel prilagođavaju konkretnom domenu svog seminarskog rada, tako što definišu odgovarajuće domenske klase za svaku metaklasu. Na taj način se formira konceptualni model softverskog sistema za konkretan domen. Za svaki definisani koncept, studenti određuju koje funkcionalnosti će sistem imati (slučajevi korišćenja kreiranje, brisanje, pretraga, prijava, izmena i ubacivanje). Na osnovu izabranog metamodela i unetih podataka o domenu, SILAB generator povezuje koncepte sistema i njihove funkcionalnosti sa odgovarajućim šablonima slučajeva korišćenja i generiše dokumentaciju poslovne logike. Tokom generisanja dokumentacije generator zamenjuje generičke delove šablona konkretnim domenskim konceptima.

Izlaz iz SILAB generatora je formatiran Word dokument koji sadrži detaljnu dokumentaciju poslovne logike softverskog sistema, a koja uključuje faze prikupljanja korisničkih zahteva i analize. SILAB generator ne generiše implementaciju poslovne logike korišćenjem određene implementacione tehnologije, već je fokusiran na prve dve faze SILAB metode. Takođe, generator ne generiše sekvencne dijagrame, niti definiše atribute koncepata, prosta i složena vrednosna ograničenja. Generisani dokument sadrži kompletnu strukturu seminarskog rada, uključujući naslovnu stranu, uvod, poglavlja koja se odnose na faze SILAB metode razvoja softvera, zaključak i literaturu. Međutim, generator automatski popunjava samo ona poglavlja koja se odnose na prve dve faze razvoja softvera, kao što su verbalni opis, slučajevi korišćenja, sistemske operacije, ugovori o sistemskim operacijama, konceptualni model, relacioni model i tabela sa strukturnim i vrednosnim ograničenjima.

Trenutna verzija SILAB generatora podržava tri različita metamodela i generisanje dokumentacije za faze prikupljanja korisničkih zahteva i analize. Koristeći generator studenti primenjuju pristup razvoja softvera zasnovan na modelima, jer metamodel predstavlja osnovu za definisanje poslovne logike softverskog sistema. Ostatak dokumentacije, koji obuhvata faze projektovanja, implementacije i testiranja, studenti samostalno izrađuju.

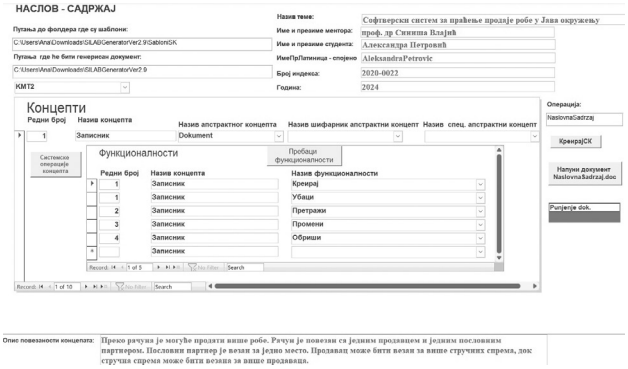
4. PRIMER KORIŠĆENJA SILAB GENERATORA

U okviru ovog poglavlja dat je studijski primer kojim je pokazan rad sa SILAB generatorom za domen Utakmica. Prvi korak u procesu generisanja dokumentacije je izbor tipa konceptualnog modela i popunjavanje metaklasa sa konkretnim konceptima. Na Slici 5. prikazan je KMT2 koji je popunjen konkretnim klasama domenskog modela.



Slika 5: KMT2 za domen Utakmica

Nakon popunjavanja metamodela, putem glavne ekranske forme NaslovnaSadržaj (Slika 6.) unose se osnovni podaci o seminarskom radu (tema, broj indeksa, godina upisa, ime i prezime mentora i studenta) i definišu funkcionalnosti (kreiraj, pretraži, promeni, ubaci, prijavi i obriši) za svaki koncept iz konceptualnog modela.



Slika 6: Ekranska forma NaslovnaSadržaj

SILAB generator generiše dokumentaciju tako što za svaku definisanu funkcionalnost traži šablon slučaja korišćenja i zamenjuje specifične (crvene) delove teksta sa konceptima domenskog modela. Na Slici 7. prikazan je primer šablona slučaja korišćenja Kreiraj koji je povezan sa konceptom Zapisnik.

СК1- Креирај записник

Назив СК

Креирај записник

Актори СК

Аналитичар

Учесници СК

Аналитичар, кориснички интерфејс (клијентски програм) и систем (серверски програм)

Предуслови: Кориснички интерфејс (клијентски програм) и систем (серверски програм) су покренути. Аналитичар је пријављен под својом шифром. Систем приказује форму за рад са записником. *Учитане су листе: a) Аналитичар b) Делегат c) Играч*

Основни сценарио СК:
1. Аналитичар **позива** систем **да креира** записник. (АПСО)
2. Систем **креира** записник. (СО)
3. Систем **приказује** аналитичару записник и поруку: "Систем је креирао записник". (ИА)
4. Аналитичар **уноси** податке о записнику. (АПУСО)
5. Аналитичар **контролише** да ли је коректно унео податке о записнику. (АНСО)
6. Аналитичар **позива** систем да запамти податке о записнику. (АПСО)
7. Систем **памти** податке о записнику. (СО)
8. Систем **приказује** аналитичару записник и поруку: "Систем је запамтио записник." (ИА)

Алтернативна сценарија:
3.1 Уколико систем не може да креира записник он **приказује** аналитичару поруку: "Систем не може да креира записник". Прекида се извршење сценарија. (ИА)

8.1 Уколико систем не може да запамти податке о записнику он **приказује** аналитичару поруку: "Систем не може да запамти записник". (ИА)

Slika 7: Generisani slučaj korišćenja Kreiraj za koncept Zapisnik

Nakon što se u generator unesu svi potrebni podaci, generiše se dokumentacija koja sadrži opisanu poslovnu logiku u Word formatu. Dokument sadrži: (I) naslovnu stranu sa nazivom teme, godinom, podacima o studentu i mentoru, (II) sadržaj, (III) uvod, (IV) poglavlja prikupljanje korisničkih zahteva, analiza, projektovanje, implementacija i testiranje, (V) zaključak i (VI) literaturu. Poglavlja uvod, zaključak, literatura, projektovanje, implementacija i testiranje su nepotpuna (prazna) i studenti moraju samostalno da ih urade. Poglavlje Prikupljanje korisničkih zahteva sadrži definisan verbalni opis domena, tabelu sa konceptima i listom funkcionalnosti za svaki koncept, kao i sve slučajeve korišćenja (funkcionalnosti). Svaki slučaj korišćenja se generiše iz odgovarajućeg šablona (Kreiraj, Pretraga, Promeni i slično).

Poglavlje Analiza se popunjava podacima koji se izvode iz slučajeva korišćenja. U okviru ovog poglavlja generišu se tabela sa potpisima metoda svih sistemskih operacija, ugovori za sistemske operacije, konceptualni i relacioni model, kao i tabele sa strukturnim i vrednosnim ograničenjima za svaki koncept sistema. Generator automatski definiše strukturna ograničenja, dok studenti samostalno dopunjuju modele atributima koncepata, kao i prostim i složenim vrednosnim ograničenjima. Takođe, studenti u okviru faze analize dodatno definišu sekvencne dijagrame slučajeva korišćenja i proširuju konceptualni i relacioni model atributima domena. Na Slici 8. prikazan je relacioni model generisan za metamodel KMT2 za domen Utakmica. Na Slici 9. prikazana je generisana tabela sa strukturnim ograničenjima za koncept Zapisnik.

1.Analiticar (idAnaliticar, . . .)
2.Tim (idTim, . . .)
3.Mesto (idMesto, . . .)
4.Kvalifikacija (idKvalifikacija, . . .)
5.Igrac (idIgrac, . . .)
6.Delegat (idDelegat, . . . , *idMesto*)
7.Zapisnik (idZapisnik, . . . , *idAnaliticar*, *idDelegat*)
8.StavkaDokumenta (*idZapisnik*, rbStavka, . . . , *idTim*)
9.SStavkaZapisnika (*idZapisnik*, *rbStavka*, rbSStavka, . . . , *idIgrac*)
10.AnalitKvalif (*idAnaliticar*, *idKvalifikacija*, . . .)

Slika 8: Relacioni model za domen Utakmica

7.Табела Zapisnik		Просто вредносно ограничење		Сложено вредносно ограничење		Структурно ограничење
Атрибути	*Назив*	Тип атрибута	Вредност атрибута	Међузав. атрибута једне табеле	Међузав.атрибута више табела	INSERT RESTRICTED Delegat , Analiticar
	idZapisnik					
						UPDATE RESTRICTED Delegat , Analiticar
						UPDATE CASCADES StavkaDokumenta
						DELETE RESTRICTED StavkaDokumenta

Slika 9: Tabela strukturnih i vrednosnih ograničenja za koncept Zapisnik

5. PREDNOSTI I OGRANIČENJA PRIMENE SILAB GENERATORA

Razvoj SILAB generatora i njegova primena u nastavi na predmetu Projektovanje softvera omogućili su strukturiran razvoj softverskog sistema i dokumentacije koja prati taj proces. Korišćenjem generatora, odnosno definisanih metamodela i šablona, smanjuje se mogućnost pojave sintaksnih i semantičkih grešaka u fazama prikupljanja korisničkih zahteva i analize. Prema iskustvu nastavnika, primenom generatora smanjen je i broj iteracija potrebnih da studenti dođu do ispravnog rešenja, posebno u delu definisanja konceptualnog modela, koji je studentima ranije predstavljao najizazovniji deo seminarskog rada. Iz ugla nastavnika, primena generatora u nastavi omogućava da se tokom konsultacija više vremena posveti razumevanju i analizi domena, a ne ispravljanju grešaka.

Jedna od značajnih prednosti generatora jeste standardizovana struktura seminarskih radova. Generisana dokumentacija ima ujednačenu strukturu, nezavisno od domena koji studenti obrađuju u svom seminarskom radu. Takođe, izmena dokumentacije se može brzo uraditi promenom ulaznih podataka i ponovnim generisanjem dokumentacije. Na ovaj način, studenti jednostavno mogu ispraviti greške, menjati domene i domenske koncepte njihovih softverskih sistema.

Navedene prednosti se mogu prikazati kroz uporedni pregled načina izrade dokumentacije pre i nakon uvođenja SILAB generatora. U Tabeli 1. prikazano je poređenje ova dva pristupa, zasnovano na iskustvima nastavnika u radu sa studentima.

Aspekt	Pre upotrebe SILAB generatora	Nakon upotrebe SILAB generatora
Struktura seminarskih radova	Seminarski radovi su bili neujednačeni.	Seminarski radovi imaju standardizovanu strukturu.
Definisanje konceptualnog modela	Studenti su često pravili greške u definisanju koncepata i relacija između njih.	Metamodeli daju smernice za definisanje koncepata i relacija.
Broj iteracija potrebnih za ispravljanje grešaka	Bilo je potrebno više konsultacija i izmena.	Smanjen je broj iteracija.
Izmena dokumentacije	Izmene su se radile ručno.	Izmene se vrše promenom ulaznih podataka i ponovnim generisanjem dokumentacije.
Konsultacije	Veći deo vremena bio je usmeren na ispravljanje grešaka.	Više vremena se posvećuje razmatranju domena.

Tabela 1. Poređenje karakteristika izrade seminarskog rada pre i nakon upotrebe SILAB generatora

Pored navedenih prednosti, SILAB generator trenutno ima određena ograničenja. Verzija alata 2.9 podržava samo tri tipa konceptualnih modela, odnosno sadrži šablone tri metamodela. Takođe, generator automatski generiše Word dokument koji sadrži samo faze prikupljanja korisničkih zahteva i analize, ne i ostale faze SILAB metode. Pored toga, generator ne generiše sekvencne dijagrame, niti definiše atribute koncepata i složena vrednosna ograničenja, već studenti te delove dokumentacije moraju da urade samostalno.

6. ZAKLJUČAK

U radu je prikazana primena SILAB generatora za automatizaciju prve dve faze SILAB metode razvoja softvera, koji se primenjuje u okviru predmeta Projektovanje softvera, na Fakultetu organizacionih nauka, Univerziteta u Beogradu. SILAB generator je alat koji omogućava generisanje dela dokumentacije koji se odnosi na poslovnu logiku softverskog sistema, čime se smanjuje mogućnost pojave sintaksnih i semantičkih grešaka u seminarskim radovima. Korišćenjem generatora studenti primenjuju strategiju razvoja softvera zasnovanu na modelima, gde proces razvoja započinju izborom metamodela i definisanjem konkretnih koncepata domena koji će zameniti metaklase.

Uvođenje SILAB generatora u nastavu omogućilo je izradu većeg broja seminarskih radova koji imaju ujednačenu strukturu, a razlikuju se u domenima koje studenti obrađuju. Jedna od važnih prednosti opisanog generatora je jednostavnost izmene dokumentacije, jer se promenom ulaznih podataka može brzo generisati nova verzija seminarskog rada.

Iako generator olakšava rad u prvim fazama razvoja softvera, njegova primena je trenutno ograničena na unapred definisane metamodele. U nastavku istraživanja planirano je proširenje funkcionalnosti SILAB generatora tako da se automatizuju i ostale faze seminarskog rada (projektovanje, implementacija i testiranje), pri čemu bi se dodatno unapredila podrška studentima u izradi seminarskih radova. U daljem razvoju alata planirano je proširenje skupa metamodela, kao i uvođenje podrške za automatsko generisanje UML dijagrama, atributa koncepata i vrednosnih ograničenja.

LITERATURA

[1] Vlajić, S. (2025). Projektovanje softvera (skripta – radni materijal, ver. 2.0). Beograd: Fakultet organizacionih nauka, Univerzitet u Beogradu.

[2] Hossain, M. I. (2023). Software Development Life Cycle (SDLC) Methodologies for Information Systems Project Management. International Journal for Multidisciplinary Research, 5(5), 1–36. https://doi.org/10.36948/ijfmr.2023.v05i05.6223

[3] Alazzawi, A., Yas, Q., & Rahmatullah, B. (2023). A comprehensive review of software development life cycle methodologies: Pros, cons, and future directions. Iraqi Journal for Computer Science and Mathematics, 4(4), 173–190. https://doi.org/10.52866/ijcsm.2023.04.04.014

[4] Acharya, B., & Sahu, P. K. (2020). Software Development Life Cycle models: A review paper. International Journal of Advanced Research in Engineering and Technology, 11(12), 169–176. https://doi.org/10.34218/IJARET.11.12.2020.019

[5] Kumar, G., & Bhatia, P. K. (2014, February). Comparative analysis of software engineering models from traditional to modern methodologies. In Proceedings of the IEEE 4th International Conference on Advanced Computing & Communication Technologies (ACCT) (pp. 189–196). IEEE. https://doi.org/10.1109/ACCT.2014.73

[6] Munassar, N. M. A., & Govardhan, A. (2010). A comparison between five models of software engineering. International Journal of Computer Science Issues, 7(5), 94–101.

[7] Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21(5), 61–72. https://doi.org/10.1109/2.59

[8] Larman, C. (1997). Applying UML and patterns. Prentice Hall PTR.

[9] Beck, K. (2000). Extreme programming explained. Addison-Wesley.

[10] Schwaber, K., & Beedle, M. (2002). Agile software development with SCRUM. Upper Saddle River, NJ: Prentice Hall.

[11] Jacobson, I., Booch, G., & Rumbaugh, J. (1999). The unified software development process. Addison-Wesley.

[12] Zhang, Y., & Patel, S. (2011). Agile model-driven development in practice. IEEE Software, 28(2), 84–91. https://doi.org/10.1109/MS.2010.85

[13] Korunović, A., Milić, M., Savić, D., & Vlajić, S. (2025). Generisanje poslovne logike softverskog sistema korišćenjem SILAB generatora. In Proceedings of the XV SPIN Conference – Business and Industrial Systems (SPIN 2025) (pp. 193–201). Faculty of Organizational Sciences, University of Belgrade. https://doi.org/10.5281/zenodo.18011960

Ana Trujić, Fakultet organizacionih nauka, Univerzitet u Beogradu
Kontakt: ana.trujic@fon.bg.ac.rs
Oblasti interesovanja: Softversko inženjerstvo; razvoj softvera; projektovanje softvera; softverski paterni.

Miloš Milić, Fakultet organizacionih nauka, Univerzitet u Beogradu
Kontakt: milos.milic@fon.bg.ac.rs
Oblasti interesovanja: Softversko inženjerstvo; razvoj softvera; projektovanje softvera; softverski paterni.

Dušan Savić, Fakultet organizacionih nauka, Univerzitet u Beogradu
Kontakt: dusan.savic@fon.bg.ac.rs
Oblasti interesovanja: Softversko inženjerstvo; razvoj softvera; projektovanje softvera; softverski paterni.

Ilija Antović, Fakultet organizacionih nauka, Univerzitet u Beogradu
Kontakt: ilija.antovic@fon.bg.ac.rs
Oblasti interesovanja: Softversko inženjerstvo; razvoj softvera; projektovanje softvera; softverski paterni.

Siniša Vlajić, Fakultet organizacionih nauka, Univerzitet u Beogradu
Kontakt: sinisa.vlajic@fon.bg.ac.rs
Oblasti interesovanja: Softversko inženjerstvo; razvoj softvera; projektovanje softvera; softverski paterni.