3. Nacionalna konferencija sa međunarodnim učešćem
Fakultet tehničkih nauka u Čačku, 25-27. Septembar 2015.

3rd National Conference With International Participation
Faculty of Technical Sciences in Čačak, 25-27. September 2015.

UDK: 004.42+004.738.12] :371 Stručni rad

# IZRADA SOFTVERSKE ARHITEKTURE MODELOM "4+1" UPOTREBOM WOLFRAM JEZIKA NA PRIMERU ELEKTRONSKOG DNEVNIKA

# MAKING SOFTWARE ARCHITECTURE WITH "4 + 1" MODEL USING BY WOLFRAM LANGUAGE– CASE STUDY OF ELECTRONIC LOG

**Sanja Mladenović[1], Vladimir Mladenović[2], Damnjan Radosavljević[3], Miroslav Lutovac[4]**
[1]*Ekonomsko trgovinska skola Paraćin*
[2]*Fakultet tehničkih nauka u Čačku*
[3]*Visoka poslovno-tehnička škola strukovnih studija Užice*
[4]*Univerzitet Singidunum*
[2]vladimir.mladenovic@ftn.kg.ac.rs, [3]damnjanr@mts.rs, [4]lutovac@gmail.com

***Apstrakt:*** *U ovom radu prikazana je procedura izrade softverske arhitekture korišćenjem softverskog paketa Mathematica Wolfram. Prikazan je način modelovanja softverske arhitekture na primeru modela "4+1" za kreiranje elektronskog dnevnika koji može biti korišćen u osnovnim i srednjim školama. Za izradu softverske arhitekture korišćen je simbolički jezik Wolfram.*
***Ključne reči:*** *Softverska arhitektura, Wolfram jezik, Mathematica Wolfram,*

***Abstract:*** *In this paper, the procedure of making software architecture using the software package Mathematica Wolfram is presented. The way of modeling software architecture is shown on the example of the "4 + 1" for creating an electronic log that can be used in primary and secondary schools. For creating of a software architecture the symbolic Wolfram language is used.*
***Key words:*** *Software architecture, Wolfram language, Mathematica Wolfram.*

## 1. UVOD

Softverski dizajn je dobio veliku pažnju istraživača 70-ih godina dvadesetog veka. Nastao je kao odgovor na jedinstvene probleme razvoja velikih softverskih sistema koji su prvi put prepoznati 60-ih godina. Tada su prvi koraci u dizajnu prepoznati kao odvojene aktivnosti kod implementacije, posebnih zahteva, tehnike dizajna i samih alata. Prvi rezultati ovakvog pristupa definisali su putanju korišćenja budućih alata za softver inžinjering pomoću računara (eng. Computer-Aided Software Engineering

(CASE) tools). Osamdesetih godina fokus istraživanja softverskog inženjerstva se udaljio od softverskog dizajna više ka integraciji dizajna i dizajniranju procesa u širem kontekstu softverskih procesa i upravljanja. Jedan od rezultata ove integracije je da su razvijene mnoge notacije i tehnike za softverski dizajn koje su apsorbovane jezicima za implementaciju [1].

U avgustu 1995. godine Institut inženjera elektrotehnike i elektronike (eng. Institute of Electrical and Electronics Engineers - IEEE) je pokrenuo inicijativu da se arhitektonski način mišljenja u softverskom inženjerstvu pretvori u standard. Kao rezultat, 2000. godine je usvojen dokument sa oznakom ANSI/IEEE 1471-2000 pod naslovom Recommended Practice for Architecture Description of Software-Intensive Systems, što bi značilo: Preporučena praksa za opis arhitekture softverski intenzivnih sistema. Pod sličnim naslovom je 2007. godine objavljen ISO/IEC 42010-2007 System and Software Engineering - Recommended practice for architecture deescription of software-intensive systems, što znači: Sistemsko i softversko inženjerstvo - preporučena praksa za arhitektonski opis softverski intenzivnih sistema.

Danas, usled razvoja velikih kompleksnih softverskih sistema, softverska arhitektura zauzima primarno mesto kod razvoja softvera u celosti, postavljajući sve kriterijume za razvoj, dajući viziju softverskog sistema, nudeći koncepte i rešenja neophodna da se jedan kompleksni softverski sistem potpuno stavi u eksploataciju. Sama izrada softverske arhitekture nije standardizovana u smislu alata koji se koriste za njenu izradu, ali se često, kao alat, koristi objedinjeni jezik za modelovanje ili (eng. Unified Modeling Language - UML).

Sa druge strane, sistem obrazovanja, kao jedan dinamičan sistem u smislu promena aktivnosti u radu, prilagođavanju nastavnih planova i programa usled novonastalih tehnologija, primeni novih modela u prenošenju znanja, i kao rezultat svega - vrednovanje, ocenjivanje i kompletno dokumentovanje procesa edukacije, nameće potrebu da u obrazovanje jedne individue, koja je najčešće mlada osoba ili dete, uključi sve aktere (učesnike) u njihovo obrazovanje i vaspitanje, gde su pored samih nastavnika obuhvaćeni i roditelji [2]. Uloga roditelja ogleda se u smislu autoriteta kao neko ko će uticati na intenzitet edukacije učenika kada je učenik kod kuće, ako ima slabiji uspeh u školi, ili se insistira na dodatnom savladavanju gradiva u cilju sticanja radnih navika i prihvatanju gradiva koje se izučava u školi. Dosadašnji najčešći način vođenja evidencije uključuju vođenje dnevnika i matičnih knjiga, a za realizaciju nastave to su operativni i globalni planovi. Jedini zvanični način da se roditelji upoznaju sa uspehom učenika je na roditeljskom sastanku ili individualnim dolaskom u školu zbog uvida u dnevnik. Savremena tehnologija je uticala je na pojavu nove pedagoške grane, koja korišćenjem računara i interneta redefiniše teoriju i praksu vaspitavanja i obrazovanja. Na sličan način kao i do sada polazi se od pretpostavke o važnosti uticaja saradnje porodice i škole na dečiji razvoj i edukaciju, i s tim u vezi kao ideja se nametnuo novi način ostvarivanja dobre komunikacije roditelja i nastavnika koja se ogleda kroz elektronski dnevnik. Iako elektronski dnevnik sada više nije ništa novo postojale su predrasude da će njegovim uvođenjem roditelji biti otuđivani od škole. Međutim, to se

nije pokazalo tako. Prvi elektronski dnevnici predstaljali su jednostavnije aplikacije na pojedinačnim računarima, dok se brzim razvojem interneta sfera njihovog korišćenja prenela na globalne mreže. Povećanjem korisnika interneta, gde spadaju i sami roditelji, aplikacije su se širile, opcije za njihovo korišćenje su se povećavale, a sa svim tim i servisi su dobijali na značaju. Ovo je nametnulo traženje modela koji će imati izgled jedne softverske arhitekture. U ovom radu konstruisana je jedna vrsta softverske arhitekture za elektronski dnevnik korišćenje Wolfram jezika za opis i izradu dokumentacije. Predložena softverska arhitektura elektronskog dnevnika obavlja i funkcije analize na osnovu statističkih podataka, strogo poštujući privatnost podataka i obavlja kumunikaciju na relaciji nastavnik-sistem-roditelj.

## 2. SOFTVERSKA ARHITEKTURA

Softverska arhitektura se bavi dizajnom i implementacijom strukture softvera na visokom nivou. To je rezultat kreiranja određenog broja arhitektonskih elemenata u nekim dobro odabranim oblicima koji zadovoljavaju visoke zahteve funkcionalnosti i performanse sistema, kao i nefunkcionalne zahteve, koji obuhvataju pouzdanost, skalabilnost, prenosivost i dostupno. Takođe, bavi se apstrakcijom, uz kompoziciju i dekompoziciju, sa stilom i estetikom.

Sledeća formula opisuje opšti pristup formulacije softverske arhitekture sa svojim elementima [1]:

Softver arhitektura = {Elementi, Obrasci, Obrazloženje/Ograničenja}

Opis sistema je često praćen sa nekoliko sugestija kojih se treba pridržavati tokom izrade dokumentacije. Svaki početak projekta sadrži osnovne elemente o projektu i obično se sastoji od elemenata koji predstavljaju smernice za njihovu izradu. Za uvodne napomene i opis jedne softverske arhitekture piše se dokumentacija sa tačke gledišta čitaoca. Treba imati na umu da se arhitektonski dokument piše jednom, ali se zato čita više puta. Za pisanje sa stanovišta čitaoca važno je da se zna kojoj ciljnoj grupi pripada, i šta im je potrebno ili šta žele da saznaju iz dokumenta. Prilikom pisanja treba izbegavati često ponavljanje tipičnih termina i rečenica. Takođe, treba izbegavati dvosmislenost (kontekste koji mogu na više načina da se tumače, od kojih je bar jedan pogrešan). Potrebno je jasno i precizno objasniti oznake koje se koristite, čak i oznake koje ne moraju da imaju neko značenje. Treba objasniti svaku odluku, uključujući i obrazloženje. Uključiti odbijene alternative sa obrazloženjem jer mogu biti od koristi u kasnijim fazama razvoja, kada se, na primer, pojavljuju novi zahtevi. Ne treba dozvoliti da dokumentacija zastari. Jedini izuzetak su privremene odluke tokom razvoja.

## 3. WOLFRAM JEZIK

Da bi se efikasno koristio Wolfram jezik (eng. Wolfram language - WL) potrebno je razumeti detalje o tome kako se WL program izvršava kada je pokrenut. Najčešća primena WL sastoji se u upotrebi različitih aspekata koji imaju drugačiji pristup od

tradicionalnih proceduranih jezika, i oni se ogledaju u prirodi pisanja izraza, načinu izračunavanja izraza, izrade odgovarajućih obrazaca za radove, kreiranja ponovo pisanih pravila, korišćenje funkcija višeg reda, kao i grafičko predstavljanje modula i relacija između njih kroz grafički korisnički interfejs, isve to opisno kroz izraze. WL predstavlja osnovno jezgro softversko paketa *Mathematica Wolfram* [3].

Iako je WL najčešće jezik koji opisuje izraze, vrlo je jednostavno da se na osnovu njih predstave grafičke ili bilo koje druge interpretacije sistema ili procesa, što će i u ovom radu biti ilustrovano. Prednost je ta što se na potpuno automatizovan način, nakon unošenja parametara i njihovih veza lako generišu objekti i relacije između njih, a grafičke interpretacije se kreću od najjednostavnijih veza preko stabala do uređenih hijerarhijskih struktura, kod kojih je na veoma pregledan način prikazan određeni proces ili sistem.

## 4. SPECIFIKACIJE NOSILACA I INTERESA

Navedena softverska arhitektura izgrađena je po opisu sistema na osnovu kog su definisane specifikacije nosilaca i interesa i realizovana je asocijativna matrica. Softverski zahtevi, gde spadaju funkcionalni, nefunkcionalni i procesni zahtevi, napisani su na osnovu prethodno definisanih specifikacija. Umesto UML, dijagrami slučajeva korišćenja su realizovani pomoću WL. Zbog ograničenja ovog rada autori nisu u mogućnosti da prikažu sve elemente softverske arhitekture elektronskog dnevnika, ali će biti prikazani najznačajniji njeni delovi. Prvo korak predstavlja definisanje specifikacije nosilaca. *Dizajner (arhitekta) sistema* - osoba koja ima novčanu korist od izrade arhitekture i kasnijeg ažuriranja i nadogradnje sistema, zadužena je za implementaciju i idejnu zamisao svih bitnih funkcija sistema gde su interesi zarada, ažuriranje i dobro poslovanje; *administrator sistema* - osoba koja treba da održava sistem, bilo da je to neko od profesora informatike iz škole ili neki drugi administrator koji će biti zaposlen i ima direktan kontakt sa korisnicima sistema i od njih sakuplja povratne informacije, i tu su interesi prikupljanje podataka o učenicima, prikupljanje novih ideja, njihovo prosleđivanje administratoru, povratne informacije, rukovođenje sistemom, i koordinacija sistema i prijavljivanje grešaka dizajneru sistema; *školski sistem* - sam elektronski dnevnik olakšava celokupan školski sistem ili program te škole, kao i administraciju kako učenika tako i zaposlenih u školi gde su interesi poboljšanje kvaliteta obrazovanja učenika i pozitivni stimulus na učenike u konjunkciji sa samim e-dnevnikom, profesorima i roditeljima; *direktor škole* - osoba koja ima uvid u administraciju i u svakom trenutku ima uvid u detaljnu analizu rada u toku školske godine kako profesora i učenika gde su interesi kvalitet nastave, zarada, zadovoljstvo korisnika i učesnika sistema i optimizacija radne snage i troškova rada, repromaterijala i energije; *Ministarstvo prosvete* - predstavlja zainteresovanu stranu iz razloga što ima uvid u sve škole koje pripadaju tom sistemu gde su interesi statistika, rangiranje, sankcije i pohvale; *profesori i nastavnici (osoblje)* - predstavljaju zainteresovanu stranu koja predstavlja ključni element u funkcionisanju sistema, vode kompletnu evidenciju događaja na jednom času, bilo da se tiče upisa ocena, izostanaka ili časa, u delu administracije ima mogućnost da vidi popis svih razreda u školi, uspeh i prosek svog

odeljenja i detaljnu analizu izostanaka koja mu olakšava prikupljanje podataka za nastavne sednice, gde su interesi profesionalnost u ocenjivanju i upotrebi sistema, izbegavanje grešaka prilikom učenja i osvajanja nove metodologije nastave, opreme i alata u skladu sa zahtevima sistema; *roditelji* - u mogućnosti su da putem interneta u e-dnevniku pogledaju uspeh, vladanje, rezultate sa takmičenja i naravno izostanke svoje dece bez dolaska u školu, uz mogućnost da unese podatke, mišljenja, predloge, sugestije, koje se onda razmatraju i nalaze se rešenja, gde su interesi valjano obrazovanje dece, opterećenost dece, dobro funkcionisanje sistema i fleksibilnost sistema podataka; i na kraju *učenici* - omogućen im je uvid u svoje ocene i izostanke što je trenutno stanje i automatski im je pokazan prosek ocena iz nekog predmeta i već znaju koja je to ocena i imaju predstavu šta bi trebala biti sledeća ocena u slučaju da žele bolje rezultate, i tu su interesi uvid u ocene, izostanci, ukori i prosek ocena. U tabeli 1 prikazane su veze nosilaca i interesa povezanih u asocijativnoj matrici nosilaca i interesa.

Tabela 1. Asocijativna matrica nosilaca i interesa

| Nosioci / Interesi | Direktor | Profesor | Učenik | Dizajner sistema | Administrator | Sistem |
|---|---|---|---|---|---|---|
| Profit | K, O, R, P | K | / | K, O, R, P | K, O, R, P | P |
| Povratne informacije | P, O | P, O | P, O | O, R | K, O, R, P | P |
| Kvalitet nastave | K, O, R, P | O, R | P | P | P | P |
| Uvid u ocene | K, O, R, P | R, O | K, P | R | R | P |
| Unapređenje sistema | K, O, R, P | K | P, K | K, O, R, P | K, O, R, P | P |
| Smanjivanje troškova sistema i ostvarivanje efikasnosti sistema | K, O, R, P | O, R, P | K, P | O,R,P | O, R, P | P |
| K – korist; O – odgovornost; R – reguliše; P - pogođen | | | | | | |

Kada su u pitanju specifikacije interesa tu se definišu: *profit koji nosi sistem* - zarada koju donosi sistem, lična zarada profesionalno napredovanje, uticaj na zadovoljstvo/nezadovoljstvo korisnika, gde su nosioci direktor, administrator i dizajner sistema; *feedback (povratne informacije)* - stvaranje baze podataka o učenicima i profesorima preduslov je za ovaj interes, na osnovu analitike dobija se opis ključnih problema koji postoje u sistemu, dizajner sistema mora da implementira formu preko koje će učenici i roditelji prijavljivati određene probleme gde su nosioci administrator, dizajner sistema; *kvalitet nastave* - dobijanje kvalitetnog sistema što bržem vremenskom periodu, smanjenje troškova održavanja, dobro korisničko iskustvo,

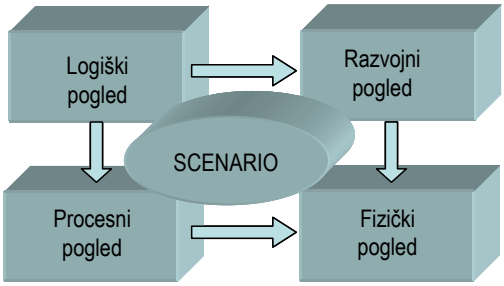
profesor ima bitan uticaj prilikom kreiranja sadržaja dok je dizajner tu da obezbedi nesmetane predispozicije gde su nosioci profesor, direktor, dizajner sistema; *uvid u ocene* - obezbeđivanje nesmetanog toka ostvarivanje prava uvida u ocene i ostale povezane stvari gde su nosioci administrator sistema, dizajner sistema, roditelji, profesori, učenik; *unapređenje sistema* - profesionalno usavršavanje nastavnika u korišćenju sistema, odgovaranje na zahteve učenika, unapređenje sistema i implementacija novih rešenja gde su nosioci profesor i dizajner sistema; *smanjivanje troškova sistema i ostvarivanje efikasnosti sistema* - osvajanje novih programa, održavanje i usavršavanje starih, efikasno čuvanje baze podataka o sistemu u finansijskom i tehničkom pogledu gde je nosilac računarski sistem obrade podataka.

## 5. OPIS SOFTVERSKE ARHITEKTURE U MODELU "4 + 1"

Da bi se opisala softverska arhitektura, koristi se model sastavljen od više pogleda ili perspektiva. Za krajnje rešavanje velikih i izazovnih arhitektura predlaže se model koji se sastoji od pet glavnih pogleda:

- *logički pogled* koji je predmet modela dizajna,
- *procesni pogled* koji snima konkurentnost i sinhronizaciju aspekta dizajna,
- *fizički pogled* koji opisuje mapiranje (mapiranja) softvera na hardver i odražava njegov distribuirani aspekt,
- *razvojni pogled* koji opisuje statičku organizaciju softvera u svom razvojnom okruženju, i
- scenario

Opis arhitekture može biti organizovan oko prva četiri pogleda, a zatim ilustrovan kroz nekoliko odabranih slučajeva korišćenja (use cases), ili scenarija koji postaju peti pogled. Na slici 1. prikazan je izgled modela "4 + 1" [1].




Slika 1. Softverski arhitektonski model "4 + 1"

**Logički pogled**. Kratak opis gledišta, korisnika sistema, funkcionalnosti i servisa koje sistem treba da obezbedi krajnjim korisnicima. Logičko gledište podržava funkcionalne zahteve, specificira servise koje sistem treba da obezbedi krajnjim korisnicima i pokazuje ključne apstrakcije u sistemu (npr. klase i njihove međusobne interakcije). *Nosioci* su dizajner (arhitekta) sistema, administrator sistema, školski sistem, direktor škole, profesori i nastavnici(osoblje), roditelji, učenici, ministarstvo prosvete, *Interesi* su funkcionalnost, povratne informacije, kvalitet nastave, uvid u ocene, unapređenje

sistema, smanjivanje troškova sistema i ostvarivanje efikasnosti sistema. Kod logičkog pogleda opisuju se dijagrami klasa, dijagram komunikacije/kolaboracije/saradnje i dijagrami sekvenci.

**Procesni pogled**. Kratak opis gledišta i nefunkcionalnih zahteva (performanse, skalabilnost, dostupnost, integritet) koji su u nadležnosti programera. *Nosioci* su developeri i integratori, a *Interesi* nefunkcionalni zahtevi: performanse, skalabilnost, dostupnost, integritet. Kod procesnog pogleda opisuju se dijagrami procesa i aktivnosti.

**Razvojni pogled**. Kratak opis gledišta, procesni zahtevi, organizacija softverskih modula iz kojih se sastoji sistem, opisuje instance statičke organizacije softvera u svom razvojnom okruženju. *Nosioci* su programer i softverski menadžer, a *Interesi* modularnost, hijerarhija slojeva, ponovno korišćenje, upravljanje softverskim projektom, ograničenja (procesni zahtevi: razvojno okruženje i alati). Kod razvojnog pogleda opisuju se dijagrami komponenata i paketa.

**Fizički pogled**. Kratak opis gledišta, topologija arhitekture, razmeštaj komponenata sistema (višeslojna arhitektura i sl.), konfiguracije za testiranje i konfiguracije za prilagođavanje sistema različitim uslugama za različite klijente, i sl. Nosioci su sistemski inženjeri, a Interesi topologija i komunikacije. Kod fizičkog pogleda opisuju se dijagrami razmeštaja/raspoređivanja.

**Scenariji**. Pogled slučajeva korišćenja opisuje ponašanje sistema, objedinjuje ostala 4 pogleda u konzistentnu celinu, obuhvata funkcionalnosti sistema kako ih vidi korisnik. Gradi se u ranim fazama razvoja. Razvijaju ga sistem analitičari i domenski eksperti. Nosioci su korisnici, sistem analitičari i domenski eksperti, a Interesi funkcionalnost i ponašanje sistema. Kod scenarija se opisuju pogleda opisuju se dijagrami slučajeva korišćenja.

## 6. IZRADA SLUČAJA KORIŠČENJA POMOĆU WOLFRAM JEZIKA

U ovom delu rada biće dat prikaz modelovanja scenarija softverske arhitekture elektronskog dnevnika pomoću Wolfram jezika. Prvi korak je definasanje naziva na osnovu prethodno preciziranih specifikacija nosilaca i interesa, a relacije su formulisane na osnovu asocijativna matrica nosilaca i interesa.

```
nazivi = {1 → "Osoblje", 2 → "Gleda raspored", 3 → "Menja raspored", 4 → "Dodaje termine za sastanak",
   5 → "Gleda termine za sastanak", 6 → "Upravlja razredima", 7 → "Pretražuje učenike",
   8 → "Modifikuje ocene učenicima", 9 → "Sprema izveštaje za roditelje", 10 → "Nastavnik", 11 → "Direktor",
   12 → "Administrator", 13 → "Učenici", 14 → "Roditelj", 15 → "Pohadjanje predmeta", 16 → "Upis ocena",
   17 → "Izdavanje svedočanstava", 18 → "Promena lozinke", 19 → "Generisanje statistike", 20 → "Sistem upravljanja",
   21 → "Rešavanje problema", 22 → "Nastavnici na obuci", 23 → "Opis problema"};
```

Sledeći korak je definisanje relacija [4]:

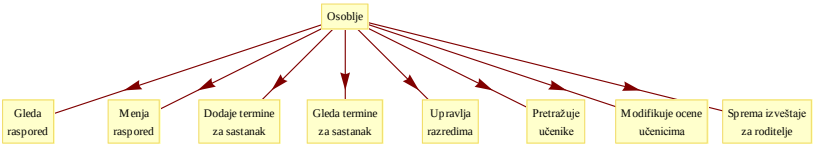
```
model1 = {1 → 2, 1 → 3, 1 → 4, 1 → 5, 1 → 6, 1 → 7, 1 → 8, 1 → 9};
```

gde se na osnovu toga crta grafički prikaz:

```
GraphPlot[model1 /. nazivi, VertexLabeling → True, DirectedEdges → True]
```

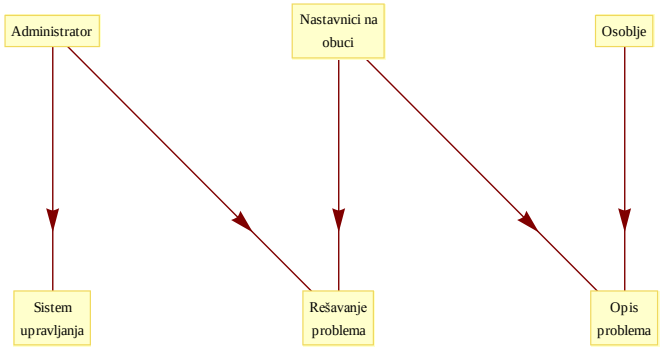
dok se za hijerarhijski prikaz piše sledeći izraz i to je prikazano na slici 2:

```
LayeredGraphPlot[model1 /. nazivi, VertexLabeling → True, EdgeLabeling → Automatic, LabelStyle → Directive[Black, 17],
 ImageSize → 800]
```




Slika 2. Hijerarhijski prikaz osnovne funkcije nastavnog osoblja u sistemu na osnovu specifikacije nosilaca sa nazivima

Na sličan način dobijaju se pojedinačne funkcionalnosti nosilaca sistema prikazane na slici 7 kao hijerarhijski model.




Slika 3. Hijerarhijski prikaz obuke nastavnika za rad na ednevniku i rešavanje problema sa neobučenim osobljem

## 7. ZAKLJUČAK

U ovom radu prikazan je novi alat za modelovanje softverske arhitekture modelom "4+1". Korišćen je Wolfram jezik koji je sastavni deo softverskog paketa Mathematica Wolfram. Prikazano je kako se na sistematski način pisanjem izraze automatski dobijaju grafičke i vizuelne interpretacije softverske arhitekture što je u krajnjem slučaju primenjeno na elektronski dmevnik koji se može da se koristi za vođenje evidencije u osnovnim i srednjim škola, koji u isto vreme omogućuje bolju informisanost roditelja o uspehu učenika i statističkim podacima.

## LITERATURA

[1] Mladenović, V., Milošević, D., (2015), *Softverske arhitekture - teorija i primena u praksi*, udžbenik FTN Čačak (u pripremi za štampu),.

[2] Popović, D., Vuković, N., (2010), *Značaj elektronskog dnevnika za ostvarivanje uspešne komunikacije između porodice i škole*, Tehnika i informatika u obrazovanju, 3. Internacionalna Konferencija, Tehnički fakultet Čačak, 7−9. maj 2010.

[3] Notebooks and documents, Wolfram Mathematica ®Tutorial Collection, www.wolfram.com/learningcenter/tutorialcollection/NotebooksAndDocuments/

[4] Graph drawing, Wolfram Mathematica ®Tutorial Collection, www.wolfram.com/learningcenter/tutorialcollection/GraphDrawing/