Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/06BE18Kukic

RAZVOJ GRAFIČKOG NAMENSKOG JEZIKA ZA ARHIVIRANJE I MIGRACIJU BAZA PODATAKA

A DEVELOPMENT OF A GRAPHICAL DOMAIN SPECIFIC LANGUAGE FOR DATABASE BACKUP AND MIGRATION

Marija Kukić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu, prezentovan je grafički namenski jezik za arhiviranje i migraciju baza podataka kreiranih pod različitim sistemima za upravljanje bazama podataka. Dat je detaljni opis razvoja jezika i metodologija koje su korišćene tokom razvoja. Predstavljeni su meta-model sa osnovnim konceptima jezika, ograničenja jezika, konkretna sintaksa i generator koda. Izvršeno je poređenje razvijenog jezika sa postojećim jezicima slične namene kako bi bile uočene prednosti i mane rešenja i potrebe za unapređenjima.*

Ključne reči: *Domenski jezik, metamodel, sistemi za upravljanje bazama podataka, generator koda.*

Abstract – *In this paper we present a graphical domain specific language for database backup and migration with a suppport for different database management systems. We give a detailed description of the language development and methodologies used during the development. Metamodel with main concepts of the language, language constraints, concrete syntax and code generator are presented. A comparison between the proposed language and other languages with a similar purpose is given for the purpose of future improvements.*

Keywords: *Domain specific language, metamodel, database management systems, code generator.*

1. UVOD

Prilikom razvoja softvera neophodno je ispoštovati korisničke zahteve sistema, smanjiti troškove razvoja i održavanja i dostaviti softver u određenom vremenskom okviru. Da bi navedeni zahtevi bili zadovoljeni neophodno je razviti strategiju i upotrebiti određenu metodologiju kako bi se obezbedile osobine koje karakterišu kvalitetan softver. Jedno od rešenja jeste primena koncepata koje propisuje softversko inženjerstvo zasnovano na modelima, eng. *Model Driven Software Engineering* (MDSE) i posebno disciplina Razvoj softvera zasnovan na modelima, eng. *Model Driven Software Development* (MDSD), kao njegov deo. MDSE je inženjerski pristup koji se zasniva na definisanju i upotrebi modela unutar softverskog razvoja [1]. MDSD, kao deo pristupa MDSE, predstavlja pristup koji definiše modele kao primarne koncepte koji se ne koriste samo u svrhu dokumentovanja, već u svrhu automatizovanog ili automatskog kreiranja krajnjeg softverskog proizvoda, odnosno generisanja programskog koda. Cilj primene navedene paradigme jeste da se automatizuje proces razvoja softvera kroz sve faze životnog ciklusa.

Razvoj namenskih jezika poštuje koncepte MDSD paradigme. Preduslov korišćenja namenskih jezika, eng. *Domain Specific Language* (DSL) jeste poznavanje koncepata jezika i njihove semantike. Korist primene ovih jezika je u tome što se transformacijama automatski iz semantike modela generiše kod. U ovom radu razvijan je grafički namenski jezik za arhiviranje i migraciju baza podataka kreiranih pod sistemom za upravljanje bazama podataka (SUBP) različitih proizvođača.

Prilikom pada baze podataka postoji rizik da se podaci iz baze podataka izgube. Kako bi bilo obezbeđeno da korisnici budu zaštićeni od gubitka podataka, podatke je potrebno arhivirati. S obzirom na to da SUBP-ovi različitih proizvođača koriste iste koncepte koji definišu osobine arhiviranja, a koriste različite procedure za sprovođenje zadataka koji imaju iste efekte, uočava se potreba za razvojem pristupa koji bi omogućio korisniku da poznavajući osobine arhiviranja specificira proceduru arhiviranja na nezavisan način od izabranog sistema za upravljanje bazama podataka. Namenskim jezikom opisanim u ovom radu korisnik specificira osobine koje definišu način i vrstu arhiviranja, a automatski se generišu skripte koje odgovaraju konkretnim sistemima za upravljanje bazama podataka različitih proizvođača. Jedan podsistem ovog grafičkog jezika namenjen je za podršku migracije baze podataka. Migracija podataka je složen proces, sastavljen od niza aktivnosti, kojima se vrše transformacije nad šemom baze podataka tako da se podaci iz starog sistema očuvavaju i prenose u novi sistem, tako da odgovaraju novim zahtevima sistema [2]. Namenski jezik opisan ovim radom podržava određeni skup transformacija kojima se vrši migracija baze podataka, tako što se vrše izmene šeme baze podataka, kreirane pod jednim konkretnim izabranim SUBP-om. Razvijeni jezik omogućava specificiranje konkretnih scenarija migracije sa ciljem da se korisnik oslobodi ručnog kreiranja skripti za izmene nad šemom. Scenarija su odabrana tako da je očuvanje podataka zagarantovano. Primarni cilj ovog rada je kreiranje novog pristupa koji će omogućiti da korisnik bez poznavanja konkretne sintakse specifičnog sistema za upravljanje bazom podataka generiše skripte za arhiviranje prema željenim parametrima. Pored toga, korisniku je potrebno omogućiti da bez poznavanja konkretne sintakse pod kojom je šema baze podataka kreirana izvrši migraciju šeme prema predefinisanim scenarijima.

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

Kako bi navedeni cilj bio postignut, neophodno je razviti namenski grafički jezik koji omogućava korisniku da izmodeluje svoje zahteve za arhiviranje i migraciju i da automatski izgeneriše skripte sa naredbama za sisteme za upravljanje bazama podataka izabranih proizvođača: *Oracle, Microsoft SQL Server* i *MySQL*. Za dostizanje prethodno definisanog cilja neophodno je izvršiti sledeće zadatake: proučiti i izvršiti analizu postupka arhiviranja i migriranja baze podataka između SUBP-ova različitih proizvođača, kreirati meta-model, definisati skup grafičkih simbola koji predstavljaju konkretnu sintasku i definisati ograničenja za validnost modela.

2. PREGLED PRETHODNIH ISTRAŽIVANJA U OBLASTI

U literaturi se može pronaći značajan broj radova koji se bave temom primene i uticaja principa MDSE na razvoj softvera. Posebno su izdvojeni radovi koji se bave primenom ovih principa u svrhu razvoja generatora SQL koda. S obzirom na to da se ovaj rad bavi razvojem grafičkog jezika, dodatano je dat osvrt na radove koji proučavaju slično okruženje za razvoj konkretne grafičke sintakse.

Upotreba MDSE principa u realizaciji generatora SQL-PL koda na osnovu definisanih OCL izraza predstavljen je u radu [3]. Sličnost ovog rada sa trenutnim istraživanjem je u tome što su i ovim radom opisana preslikavanja izvornih koncepata na ciljne, tako da se generišu skripte za nekoliko SUBP-ova, kao što su *MySQL, MariaDB, PostgreSQL* i *SQL Server*. Ovim radom takođe je objašnjena pojava da različiti RDBMS implementiraju određene sintaktičke varijacije u standardnu SQL notaciju. Dodatno, ovaj rad naglašava kako je izgradnja modela centralna u dizajnu sistema. Osim toga, naglašava da MDSE principi ispoljavaju pozitivne aspekte sa stanovišta kvaliteta razvoja softvera, kada se izvorni jezik za modeliranje može potpuno povezati sa izabranim ciljnim jezikom.

Prednosti razvoja softvera zasnovanog na modelima, motivacija za potrebom generisanja SQL skripti i koraci implementacije SQL generatora detaljno su opisani u radu [4]. Autori ovog rada bave se razvojem alata za razvoj softvera kojim se dizajnira model nezavisan od tehnologije, na osnovu koga se kao rezultat generiše izvršni programski kod za različita ciljna okruženja i SQL skripte, za različite ciljne RBDMS.

Sa stanovišta sličnih tehnologija i domena istraživanja izdvojen je rad [5] koji opisuje modele baza podataka. U njemu je detaljno prikazan meta-model šeme relacionih baza podataka specificiran pomoću *Eclipse Modeling Framework* (EMF), čime je specificiran i meta-model razvijanog jezika. S obzirom na to da je tema ovog rada razvoj grafičkog jezika, rad koji predstavlja sličan razvoj konkretne sintakse je [6]. U ovom radu korišćen je *Sirius* okvir za razvoj grafičkog uređivača domenskog modela, koji pojednostavljuje specifikaciju proizvoda, smanjuje vreme dizajna i brzo povećava ukupnu produktivnost.

3. DOMEN PROBLEMA

Sistemi za upravljanje bazama podataka sadrže mehanizme koji obezbeđuju kreiranje arhive podataka kao vid preventivnog načina za zaštitu od gubitka podataka. Kreiranje arhive naziva se *backup*. *Backup* je kopija podataka iz baze koja se koristi kako bi bilo moguće rekonstruisati podatke kad je to neophodno. *Backup* podataka moguće je sprovesti u skladu sa potrebama korisnika. Pod tim se podrazumeva da postoji mogućnost odabira strategije, vrste, tipa i režima u kojem se vrši *backup*. Specificiranje strategije, vrste, tipa i režima arhiviranja podržavaju svi izabrani SUBP-ovi različitih proizvođača za koje je arhiviranje razvijenim jezikom podržano.

Arhiviranje *Oracle* baze podataka moguće je sprovesti na tri načina, a to su: koristeći *Recovery Manager* (RMAN), *Enterprise Manager* (EM) ili ručnim kreiranjem skripte. Namenski jezik opisan u ovom radu funkcioniše tako što se grafičkim simbolima zadaju tip, strategija, vrsta i režim arhiviranja nakon čega se automatski generiše RMAN skripta koju treba izvršiti kroz komandnu liniju. Sličnost sa arhiviranjem koristeći EM je to da postoji korisnički interfejs i to što se skripte automatski generišu, a sličnost arhiviranju koristeći RMAN je da skripte treba izvršiti kroz komandnu liniju. Ukoliko korisnik ima potrebu da arhivira *Microsoft SQL Server* bazu podataka, neophodno je da prouči koncepte za korišćenje *SQL Server Agent* i da ručno kodira *T-SQL* skriptove. Korišćenjem grafičkog namenskog jezika opisanog u ovom radu, ove skripte generišu se automatski. Za kreiranje arhive *MySQL* baze podataka postoji više načina od kojih je najzastupljeniji onaj koji je zasnovan na *MySQLDump*-u. *MySQLDump* je uslužni program za *MySQL* i omogućava zadavanje niza opcija za arhiviranje baze podataka. Razvijenim grafičkim jezikom omogućeno je automatsko generisanje naredbi sa željenim parametrima za ovaj program. Koristeći ove načine, moguće je definisati strategiju kreiranja arhive, vrstu arhive, tip arhive i režim u kojem se vrši arhiviranje. Strategija podrazumeva odabir da li se arhivira celokupna baza ili samo deo baze podataka. Vrstom se definiše da li se kopiraju svi podaci ili samo oni koji su se promenili u odnosu na prethodni *backup*. Tipom se definiše na koji način će podaci biti skladišteni i kompresovani. Režim može biti *offline* ili *online*.

Proces migracije podrazumeva bilo koju transformaciju strukture baze podataka koja nastaje kao posledica izmena u strukturi šeme baze podataka. Migracijom se ne transformiše samo šema baze podataka, već se transformišu podaci u bazi podataka, da bi odgovarali novom opisu koji zadaje šema baza podataka. Grafičkim namenskim jezikom opisanim u ovom radu podržano je nekoliko scenarija za promenu nad šemom baze podataka, čime se programeri oslobađaju potrebe za poznavanjem konkretne sintakse i ručne implementacije SQL skriptova. Podržana scenarija su:

- dodavanje nove kolone datog tipa i s mogućnošću opcionog zadavanja predefinisane vrednosti i određenih ograničenja,
- razdvajanje jedne kolone sa postojećim podacima na više kolona sa očuvanim podacima prilagođenim specificiranim zahtevima,
- spajanje više kolona sa postojećim podacima u jednu kolonu sa očuvanim podacima prilagođenim specificiranim zahtevima,
- razdvajanje postojeće tabele na više tabela sa definisanim preuzetim kolonama i
- spajanje više tabela prema zajedničkoj koloni, pri čemu se specificira mogući način spajanja.

Prilikom razvoja namenskog jezika potrebno je utvrditi korisničke zahteve i u skladu sa njima definisati koncepte jezika, tako da se opravda potreba za razvojem takvog alata. Očekivano je da potrebu za ovim akcijama imaju programeri koji koriste podatke i poznaju tehnologiju, ali ne poznaju u dovoljnoj meri koncepte administracije baza podataka i ne žele da se izlažu riziku da poremete strukturu baze podataka, ukoliko bi ručno vršili izmene.
Ključna razlika između ovog alata i alata slične namene u domenu poslova administracije baze podataka je što većina alata obezbeđuje arhiviranje podataka putem samo jednog konkretnog SUBP-a, izabranog proizvođača. Ovim jezikom korisnik može specificirati da se istovremeno formiraju skripte za arhiviranje baza podataka kreiranih pod SUBP-om različitih proizvođača. Osim toga, prednost ovog alata u odnosu na alate namenjene za migraciju podataka jeste u tome što sadrži predefinisani skup scenarija korišćenja u kojima se garantuje da će podaci biti sačuvani i da ručne izmene nisu potrebne.

4. RAZVOJ GRAFIČKOG NAMENSKOG JEZIKA ZA ARHIVIRANJE I MIGRACIJU BAZE PODATAKA

U ovom odeljku detaljno je opisan metamodel na kojem se zasniva konkretna sintaksa, OCL ograničenja koja obezbeđuju pravila za validnost modela, pregled grafičkih simbola sa opisanom semantikom korišćenja, opis korišćene tehnologije u svrhu implementacije i opis generatora koda.

4.1. Metamodel grafičkog namenskog jezika za arhiviranje i migraciju baze podataka

Metamodel biće opisan kroz dva dela: metamodel za definisanje koncepata za arhiviranje baze podataka i metamodel za definisanje koncepata za migraciju baze podataka. Metamodel kreiran je korišćenjem *Eclipse Modeling Framework*-a, koji se zasniva na korišćenju *Ecore* jezika za metamodelovanje.
Korenski element koji sadrži sve ostale elemente je metaklasa *Backup*. Na ovakav način se obezbeđuje, da ukoliko se pristupi jednom elementu, dobijaju se informacije o svim ostalim elementima. Elementi za definisanje parametara za arhiviranje su strategija, tip, vrsta i režim što je predstavljeno metaklasama *Strategija*, *Tip*, *Vrsta* i *Rezim*. Pored toga data je metaklasa *DodatneOsobine* kojom se obezbeđuje da se pored obaveznih parametara zadaju i parametri koji bi arhivu prilagodili korisničkim potrebama. Minimalna specifikacija mora da sadrži strategiju i režim gde se zadaju obavezni parametri za arhiviranje. Ukoliko se vrši parcijalno arhiviranje, neophodno je da se definiše da li se arhivira tabela, datoteka ili tablespace što se opisuje metaklasama *Table*, *File* i *Tablespace*. Za bazu podataka, proizvođač se definiše zadavanjem atributa *subp* čije vrednosti su iz enumeracije *Proizvodjac*. Enumeracija *Proizvodjac* definiše da li će se generisati naredbe za *Oracle, MySql ili SQLServer*. Dodatno, moguće je opciono specificirati *ZipArhivu* koja obuhvata na kojoj lokaciji se skladišti arhiva, pod kojim nazivom i koje datoteke ona obuhvata.
Osim elemenata koji su neophodni kako bi bili opisani procesi arhiviranja, postoje i metaklase koje su neophodne za pravilno definisanje modela. U cilju specifikacije načina povezivanja elemenata, kao i redosleda povezivanja, uvodi se posebna metaklasa *Element*. Metaklasu *Element* nasleđuju *Strategija, Tip, Vrsta, Rezim* i *DodatneOsobine*. Definisanjem elementa je podignut nivo apstrakcije gde svaka od ovih metaklasa predstavlja element sa osobinom redosleda.
Scenarija migracije podržane ovim grafičkim jezikom su dodavanje kolone, spajanje više kolona u jednu, razdvajanje jedne kolone na više, spajanje i razdvajanje tabela po određenom kriterijumu. U skladu sa tim, metamodel sadrži metaklase *RadSaKolonama* i *RadSaTabelama*. *RadSaKolonama* obuhvata atribute neophodne za generisanje skriptova koji sadrže SQL naredbe za dodavnje nove kolone, razdvajanje jedne kolone na više kolona po određenom delimiteru i spajanje više kolona u jednu sa očuvanim podacima. *RadSaTabelama* obuhvata atribute neophodne za generisanje skriptova koji sadrže SQL naredbe za spajanje više tabela u jednu prema zajedničkoj koloni ili razdvajanje jedne tabele i njenih kolona na više tabela.

4.2. Ograničenja grafičkog namenskog jezika za arhiviranje i migraciju baze podataka

Pri kreiranju jezika nije moguće sva potrebna ograničenja implementirati na nivou strukture metamodela, jer jezici za metamodelovanje obuhvataju ograničen skup koncepata za definisanje novog jezika. Implementacija ograničenja obezbeđuje kreiranje pravila koja definišu validan model. U slučaju narušavanja ograničenja, korisnik je obavešten o konkretnoj grešci i onemogućeno je generisanje daljih koraka. Ograničenja su implementirana pomoću OCL-a. OCL je formalni jezik koji se koristi za dopunu strukture metamodela kroz skup tekstualnih pravila sa kojim svaki kreirani model mora biti u skladu. Neka od ograničenja su: postavljanje određenih polja kao obaveznih, definisanje ograničenja jedinstvenosti, ograničenja koja kontrolišu validnost unosa određenog atributa, ograničenja koja kontrolišu redosled i način povezivanja elemenata modela.

4.3. Konkretna sintaksa grafičkog namenskog jezika za arhiviranje i migraciju baze podataka

Razvoj grafičke konkretne sintakse namenskog jezika za arhiviranje i migraciju baze podataka obuhvata više koraka. Najpre je definisan skup simbola. Skup simbola izabran je tako da jezik bude intuitivan za korišćenje i da simboli odgovaraju semantici. Za simbole koje nemaju uobičajnu reprezentaciju izabrana je proizvoljna grafička predstava. Nakon specifikacije simbola i njihove semantike, definisana su pravila kombinovanja i povezivanja pojedinačnih elemenata. Na kraju je svaki element modela preslikan na koncept metamodela. Za razvoj konkretne sintakse korišćen je *Sirius* koji predstavlja *Eclipse* projekat koji omogućava da se kreira radna površina za grafičko modelovanje oslanjajući se na tehnologije *Eclipse Modeling Frameworka.*

4.4. Generator koda

Ideja razvoja ovog jezika je da se podrže specifični, jasno definisani scenariji za koje je smatrano da je potreba najveća, tako da korisnik dobije kompletno izgenerisane skripte za konkretnu situaciju. Prednost je to što ukoliko korisnik koristi jezik u opisanom domenu, neće imati potrebe da dodaje ručno pisani kod. Mana je to da su

scenariji arhiviranja i migracije usko specificirani, tako da ukoliko korisnik ima potrebe za dodatnim funkcionalnostima, morao bi ručno da dodaje kod u skriptove. Generator koda realizovan je pisanjem šablona unutar *.xtend* fajla. Generator obuhvata šablone za generisanje skriptova za arhiviranje i šablone za generisanje skriptova za migraciju baza podataka. Uz svaki kodirani šablon automatski se kreira jedna, ili više kompitabilnih izvršivih *Java* metoda. S obzirom na to da različiti SUBP-ovi imaju različitu sintaksu i postupke za kreiranje arhive, neophodno je za svaki SUBP kreirati po jedan šablon. Rezultat svakog šablona je po jedna skripta, koju je potrebno pokrenuti pod određenim SUBP-om.

5. POREĐENJE SA ALATIMA SLIČNE NAMENE

Poređenje je vršeno sa alatima koji su izabrani tako da kao krajnji proizvod generišu SQL skripte prilagođene za *Oracle*, a to su: *Oracle Enterprise Manager* (EM) i *Oracle Data Integrator* (ODI). EM predstavlja veb alat za upravljanje bazama podataka koji je ugrađen unutar SUBP *Oracle*. EM obezbeđuje zadavanje parametara za kreiranje arhive, gde se kao rezultat generiše RMAN skripta. Sličnost sa razvijanim jezikom je u tome, što korisnik ukoliko poznaje domen problema i vrednosti parametara koji su neophodni, bez poznavanja konkretnih procedura i sintakse može da arhivira svoje podatke. Razlika je u tome što je korisnički interfejs za zadavanje parametara drugačiji. Prednost EM u ovom segmentu je u tome što korisnik ne mora da proučava pravila jezika za kreiranje modela. Sa druge strane, prednost razvijanog alata je u tome što podržava generisanje skripti za više proizvođača, dok je EM isključivo primenljiv u okviru SUBP Oracle.

ODI je alat čije korišćenje ima drugačiju svrhu od razvijanog jezika. Bez obzira na to, razlog zbog čega je ovaj alat izabran za poređenje je u tome što postoje određene sličnosti sa razvijenim jezikom. Sličnosti su:

- ODI predstavlja grafički alat kojim se specificira grafički model sa određenim osobinama,
- namenjen je za sličnu grupu korisnika kao i razvijeni jezik, a to su eksperti domena koji poznaju relacioni model, ali korišćenjem alata su oslobođeni od kodiranja i
- grafičke elemente je potrebno povezati u sekvencu, kako bi se izgenerisala i izvršila SQL skripta koja vrši procese specificirane grafičkim modelom.

6. ZAKLJUČAK

U ovom radu opisan je razvoj namenskog grafičkog jezika za automatsko generisanje skripti za arhiviranje i migraciju za određene proizvođače baza podataka. Razvoj je obuhvatao specifikaciju zahteva, definisanje meta-modela zasnovanog na *Ecore*, implementaciju ograničenja, definisanje konkretne sintakse i šablona za generisanje koda.

Jezik obuhvata deo namenjen za administraciju baze podataka i deo namenjen za migraciju baze podataka. Deo za administraciju baze podataka obuhvata zajedničke koncepte sistema za upravljanje bazama podataka različitih proizvođača i time rešenje može da zadovolji uobičajne korisničke potrebe u domenu poslova administracije. U poređenju sa alatima slične namene, prednost je u tome što je generisanje obezbeđeno za više SUBP-ova različitih proizvođača definisanjem samo jednog modela. U daljem razvoju očekuje se proširenje jezika konceptima kojima bi se pokrile specifičnosti za kreiranje arhive pojedinačnih proizvođača. Proširenje bi obuhvatalo definisanje koncepata u meta-modelu, definisanje i implementaciju ograničenja, definisanje grafičkih simbola i dopunu šablona za generisanje koda.

Deo za migraciju podataka obuhvata predefinisana scenarija migracije baze podataka. Istraživanjem literature ustanovljeno je da su ova scenarija najviše potrebna u praksi. Prednost, u poređenju sa ostalim rešenjima, u tome je što ovim jezikom korisnik kao rezultat dobija izmenjenu bazu podataka sa zasigurno očuvanim podacima u željenoj formi. Mana je u tome što postoji ograničeni skup predefinisanih scenarija.

Nakon poređenja grafičkog namenskog jezika za arhiviranje i migraciju baze podataka sa jezicima slične namene i nakon proučavanja radova koji se bave sličnim temama uočene su potrebe za buduća proširenja. Sledeći korak u razvoju grafičkog jezika za arhiviranje i migraciju baza podataka je proširenje jezika tako da podrži koncepte neophodne za proceduru oporavka. Pored toga, kako bi se poboljšalo korisničko iskustvo, potrebno je određene parametre preuzeti iz baze podataka, čime se izbegavaju greške zadavanja parametara kroz atribute modela. Osim ovog proširenja potrebno je proširiti skup predefinisanih scenarija za migraciju.

7. LITERATURA

[1] Marco Brambilla, Jordi Cabot, and Manuel Wimmer (2012), Model-Driven Software Engineering in Practice. A Publication in the Morgan & Claypool Publishers series SYNTHESIS LECTURES ON SOFTWARE ENGINEERING

[2] Sava Jelisavčić, Ivan Jelisavčić (2011), Metodologija migracije podataka velikih informacionih sistema, INFOTEH-JAHORINA Vol. 10, Ref. E-I-4, p. 409-413

[3] Marina Egea, Carolina Dania (2017), SQL-PL4OCL: An Automatic Code Generation from OCL to SQL Procedural Language ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)

[4] Ivan Luković, Sonja Ristić, Slavica Aleksić, Aleksandar Popović (2008), An Application of the MDSE Principles in IIS*Case, SIG Model Driven Software Engineering: 3rd Workshop MDSE 08, Serbia

[5] Sonja Ristić, Slavica Aleksić , Milan Čeliković, Ivan Luković (2013), An EMF Ecore Based Relational DB Schema Meta-Model, ICIT 2013 The 6th International Conference on Information Technology

[6] Vladimir Vujović, Mirjana Maksimović, Branko Perišić (2014), Sirius: A Rapid Development of DSM Graphical Editor, IEEE 18th International Conference on Intelligent Engineering Systems, Tihany, Hungary

Kratka biografija:

Marija Kukić rođena je u Zrenjaninu 1995. god. Fakultet tehničkih nauka upisala 2014. god. Bečelor rad iz oblasti Elektrotehnike i računarstva – Računarske nauke i informatika odbranila je 2018. god. Master rad iz iste oblasti odbranila je 2019. god.