UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
Departman za računarstvo i automatiku

Jelena Banović

JEDAN PRISTUP GENERISANJU IZVRŠNIH SOFTVERSKIH SPECIFIKACIJA INFORMACIONOG SISTEMA

- DOKTORSKA DISERTACIJA -

Mentor
Prof. dr Ivan Luković

Novi Sad, 2010.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Докторска дисертација
Аутор, **АУ**:		мр Јелена Бановић
Ментор, **МН**:		др Иван Луковић, редовни професор
Наслов рада, **НР**:		Један приступ генерисању извршних софтверских спецификација информационог система
Језик публикације, **ЈП**:		Српски
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Србија
Уже географско подручје, **УГП**:		АП Војводина
Година, **ГО**:		2010
Издавач, **ИЗ**:		Факултет техничких наука
Место и адреса, **МА**:		Трг Д. Обрадовића 6, 21000 Нови Сад
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		6/ 159/ 0/ 30/ 100/ 0/ 3
Научна област, **НО**:		Софтверско инжењерство
Научна дисциплина, **НД**:		Примењене рачунарске науке и информатика
Предметна одредница/Кључне речи, **ПО**:		Софтверско инжењерство, алат IIS*Case, генератор апликација, прототип апликације, општи модел корисничког интерфејса, CASE алат
УДК		004.414.32:004.4'242(043.3)
Чува се, **ЧУ**:		Библиотека Факултета техничких наука, Трг Д. Обрадовића 6, 21000 Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Циљ истраживања реализованих у овом раду, био је моделовање и формална презентација једног приступа рјешењу проблема генерисања извршних софтверских спецификација информационог система. Креирањем речника општег модела корисничког интерфејса, омогућено је дефинисање параметара који специфицирају визуелни дизајн корисничког интерфејса. Практична употреба оваквог приступа састоји се у аутоматском генерисању прототипова који су врло блиски финалној имплементацији апликативних система неког информационог система, коришћењем генератора имплементираног у оквиру алата IIS*Case. Тиме је омогућено да се, у врло кратком року, изгенерише функционалан кориснички интерфејс за приказ и ажурирање података.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	Др Слободанка Ђорђевић-Кајан, редовни професор
	Члан:	др Бранко Перишић, ванредни професор
	Члан:	др Миро Говедарица, ванредни професор
	Члан:	др Соња Ристић, ванредни професор
	Члан, ментор:	др Иван Луковић, редовни професор

Потпис ментора

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:		
Identification number, **INO**:		
Document type, **DT**:		Monograph documentation
Type of record, **TR**:		Textual printed material
Contents code, **CC**:		Ph.D. thesis
Author, **AU**:		Jelena Banović, M.Sc.
Mentor, **MN**:		Ivan Luković, Ph.D., Full Professor
Title, **TI**:		An Approach to Generating Executable Software Specifications of an Information System
Language of text, **LT**:		Serbian
Language of abstract, **LA**:		Serbian
Country of publication, **CP**:		Serbia
Locality of publication, **LP**:		AP Vojvodina
Publication year, **PY**:		2010
Publisher, **PB**:		Faculty of Technical Sciences
Publication place, **PP**:		Trg D. Obradovića 6, 21000 Novi Sad
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		6/ 159/ 0/ 30/ 100/ 0/ 3
Scientific field, **SF**:		Software engineering
Scientific discipline, **SD**:		Applied computer science
Subject/Key words, **S**/**KW**:		Software engineering, IIS*Case tool, application generator, application prototype, UI common model, CASE tool
UC		004.414.32:004.4'242(043.3)
Holding data, **HD**:		Library of Faculty of Technical Sciences, 21000 Novi Sad, Trg Dositeja Obradovića 6
Note, **N**:		
Abstract, **AB**:		The aim of the research presented in this work is a modeling and formal presentation of an approach to the problem of generating executable software specifications of an information system. A specification of common user interface models is provided by defining the appropriate repository definitions. Practical application of the approach is provided by the implementation of an application generator as a part of IIS*Case tool. It provides automated generation of the prototypes that are very close to the final implementation of the system applications. By this, it is possible to efficiently generate dynamic user interface for data presentation and database update operations.
Accepted by the Scientific Board on, **ASB**:		
Defended on, **DE**:		
Defended Board, **DB**:	President:	Slobodanka Đorđević-Kajan, Ph.D., Full Professor
	Member:	Branko Perišić, Ph.D., Associate Professor
	Member:	Miro Govedarica, Ph.D., Associate Professor
	Member:	Sonja Ristić, Ph.D., Associate Professor
	Member, Mentor:	Ivan Luković, Ph.D., Full Professor

Menthor's sign

Sadržaj

1. Uvod

Uspješno generisanje programskog koda oslanja se na dobro definisanu specifikaciju projektovanog informacionog sistema. Umjesto pisanja specifikacija u tekstualnom formatu, potrebno je razviti formalni model sa grafičkom reprezentacijom, putem kojeg je moguće precizno specificirati informacioni sistem. Model mora sadržati dovoljno informacija da nedvosmisleno opiše zahtijevane funkcionalne i nefunkcionalne karakteristike informacionog sistema. U tom cilju potrebno je kreirati metamodel za projektovanje specifikacija informacionog sistema, odnosno jezik koji opisuje strukturu datih specifikacija. Mnogi alati koji podržavaju grafičko modelovanje, imaju ugrađene funkcionalnosti za provjeru postojanja nekompatibilnosti ili dvosmislenosti u specifikacijama informacionog sistema, pomoću kojih se već u ranoj fazi razvojnog ciklusa mogu detektovati neusaglašenosti u definiciji poslovnih pravila i ograničenja. Zato, pomenuti model mora da implementira i postupke provjere usaglašenosti specifikacija, prije samog sprovođenja procesa generisanja programskog koda.

Cilj istraživanja, prikazanih u ovom radu, bio je modelovanje i formalizovana prezentacija jednog pristupa rješenju problema generisanja izvršnih softverskih specifikacija, koji bi omogućio automatsko generisanje prototipova aplikacija u ranoj fazi razvoja informacionog sistema. Dodatno, izvršne softverske specifikacije informacionog sistema proširene su specifikacijom opšteg modela korisničkog interfejsa. Ovo znači precizno definisanje svih parametara koji specificiraju vizuelni i dijelom funkcionalni dizajn korisničkog interfejsa. Kreiranjem rečnika opšteg modela korisničkog interfejsa, omogućava se specificiranje šablona korisničkog interfejsa. Projektant bi, birajući jedan od predefinisanih šablona ili kreirajući svoj, mogao na različite načine vizuelno interpretirati generisane izvršne specifikacije. Ti šabloni bi, osim vizuelnih karakteristika, specificirali način prikaza strukture menija, izgled i strukturu ekranskih formi za prikaz i ažuriranje podataka. Praktična upotreba ovakvog pristupa sastoji se u generisanju prototipova koji su vrlo bliski finalnoj implementaciji aplikativnih sistema nekog informacionog sistema, što omogućava otklanjanje svih grešaka i nedostataka u modelu sistema još u najranijim fazama razvojnog procesa. Takođe, na ovaj način moguće je u vrlo kratkom roku izgenerisati funkcionalan korisnički interfejs za prikaz i ažuriranje podataka.

IIS*Studio je skup alata koji između ostalih funkcionalnosti, pruža podršku generisanju izvršnih softverskih specifikacija i specificiranju dizajna korisničkog interfejsa. Najvažniji alat ovog razvojnog okruženja je IIS*Case, CASE alat, koji podržava konceptualno modelovanje šema baza podataka zasnovano na konceptu tipa forme i generisanje programskih specifikacija. IIS*UIModeler je takođe dio okruženja IIS*Studio, i služi za specificiranje šablona korisničkog interfejsa koji mogu biti izabrani prilikom generisanja

izvršnih softverskih specifikacija.

Tipovi formi predstavljaju modele koji specificiraju forme korisničkog interfejsa putem kojih krajnji korisnici komuniciraju sa informacionim sistemom. Pristup zasnovan na konceptu tipa forme podrazumijeva da projektant, tokom modelovanja informacionog sistema, specificira tipove formi, posredno zadajući polazni skup ograničenja, koji će, tokom procesa generisanja, biti transformisan u implementacionu šemu baze podataka u trećoj normalnoj formi, na automatizovan način. Specifikacije tipova formi su polazna tačka i za generisanje izvršnih softverskih specifikacija. Osim elemenata koji implementiraju funkcionalnosti koje omogućavaju interaktivnost sa korisnikom kao i komunikaciju sa odgovarajućom bazom podataka, ove specifikacije uključuju i elemente koji specificiraju strukturu korisničkog interfejsa.

Rezultati istraživanja, sprovedenog u ovoj doktorskoj disertaciji, odnose se na kreirane postupke i algoritme kojima se obezbjeđuje oblikovanje opštih modela korisničkog interfejsa i generisanje prototipova aplikacija informacionih sistema. Njihovom implementacijom, postojeće funkcionalnosti alata IIS*Case proširuju se dodatnim alatima koji omogućavaju specifikaciju šablona korisničkog interfejsa i generisanje odgovarajućih funkcionalnih prototipova aplikacija informacionog sistema. U okviru istraživanja, realizovani su sljedeći zadaci:

- formirana je opšta specifikacija šablona korisničkog interfejsa aplikacija informacionih sistema i implementiran alat za zadavanje datih specifikacija;
- implementirani su algoritmi za generisanje podšema tipova formi neophodnih za implementaciju funkcionalnosti prikaza i ažuriranja podataka putem korisničkog interfejsa generisanih prototipova aplikacija; i
- implementiran je generator koji na osnovu zadatih specifikacija aplikacija informacionih sistema i izabranog šablona korisničkog interfejsa generiše funkcionalne prototipove aplikacija.

Ostvarivanjem navedenih zadataka, zaokružen je proces projektovanja informacionih sistema od početne faze zadavanja skupa polaznih ograničenja, preko generisanja šema baza podataka, do finalnog generisanja prototipova aplikacija informacionih sistema. Proces je podržan skupom vizuelno orijentisanih alata koji, s jedne strane, automatizuju složene postupke generisanja, a s druge podržavaju neophodne domenski specifične koncepte i time obezbjeđuju korisniku jednostavniju upotrebu i intuitivni pristup u radu. Projektanti informacionih sistema dobijaju razvojno okruženje u okviru kojeg bez visoko formalnih, naprednih znanja i iskustva u oblasti projektovanja informacionih sistema, mogu u kratkom vremenu da dođu do željenih rezultata: implementacione šeme baze podataka i funkcionalne aplikacije informacionog sistema.

Rad se sastoji od šest poglavlja uključujući Uvod i Zaključak, kao i tri priloga.

Nakon Uvoda, u drugom poglavlju rada, pod nazivom *Pregled stanja u oblasti*, prikazani su izabrani rezultati aktuelnih istraživanja u oblasti razvoja informacionih sistema, koji se odnose na pristupe projektovanja informacionih sistema zasnovanih na modelima. U ovom poglavlju diskutovani su *Model-Driven* pristupi koji se zasnivaju na kreiranju visoko formalnih modela, ne samo u funkciji dokumentovanja, već i u funkciji generisanja programskih specifikacija. Na taj način svrha specificiranja modela nije samo ograničena na rane faze razvoja kada se kreira projektna dokumentacija sistema, već i na kasnije faze, kada

se ti modeli aktivno upotrebljavaju u smislu sprovođenja niza formalnih transformacija koje rezultiraju finalnom implementacijom aplikativnog sistema. Model je usko povezan sa konkretnom implementacijom i bilo kakve izmjene specifikacije modela direktno izazivaju izmjene programskog koda u implementaciji aplikacija informacionog sistema. Na tržištu postoji veliki broj alata, koji u manjoj ili većoj mjeri podržavaju pristupe razvoja informacionih sistema zasnovanih na modelima i, bar u određenoj mjeri, automatizuju takav proces razvoja softvera.

U trećem poglavlju rada opisuju se osnovne funkcionalnosti alata koji pripadaju razvojnom okruženju IIS*Studio. Predstavljeni su alati IIS*Case i IIS*UIModeler namijenjeni modelovanju informacionih sistema i generisanju funkcionalnih prototipova aplikacija. Napredna znanja iz oblasti upravljanja bazama podataka i poznavanje programskih jezika ili složenih okruženja za razvoj softvera, nisu strogi preduslov za korišćenje ovih alata. Praktičan i intuitivan korisnički interfejs ovih alata omogućava projektantima koji nisu stručnjaci iz oblasti informatike i računarskih nauka, da nakon kraće obuke mogu samostalno i u kratkom roku, doći do funkcionalnog prototipa aplikacije. Specificiranjem polaznih ograničenja, pravila poslovanja, funkcionalnosti, interfejsa i različitih struktura, projektant započinje proces modelovanja informacionog sistema. Tokom tog procesa, projektant je podržan različitim alatima koji obezbjeđuju vođenu izradu specifikacija i automatizaciju određenih projektantskih aktivnosti. Krajnji rezultat ovog procesa predstavlja konkretna implementacija šeme baze podataka i generisane aplikacije koje se oslanjaju na datu šemu baze podataka i implementiraju željene funkcionalnosti. U poglavlju je opisano na koji način projektant upotrebom alata IIS*Case generiše šemu relacione baze podataka. Proces generisanja buduće šeme baze podataka informacionog sistema je iterativan. On uključuje i integrisanje šeme baze podataka datog aplikativnog sistema sa podšemama koje odgovaraju aplikativnim podsistemima datog sistema, kao i konsolidaciju integrisane šeme sa podšemama. Kroz taj proces, projektant kreira šablone korisničkog interfejsa upotrebom IIS*UIModeler-a, definiše dodatna poslovna pravila, i kao krajnji rezultat generiše željene prototipove aplikativnih sistema.

Kreiranje opšteg modela vizuelnog izgleda korisničkog interfejsa nezavisno od modelovanja funkcionalnosti aplikativnih sistema, dobra je praksa u razvoju informacionih sistema. IIS*Studio je okruženje za razvoj informacionih sistema koje omogućava modelovanje vizuelnog dizajna korisničkog interfejsa. U četvrtom poglavlju rada pod nazivom *Kreiranje šablona korisničkog interfejsa* opisano je kako je korišćenjem alata IIS*UIModeler moguće kreirati različite šablone korisničkog interfejsa, koji mogu biti izabrani kroz IIS*Case alat prilikom automatskog generisanja izvršnih softverskih specifikacija. Na ovaj način, generisana aplikacija je, osim toga što implementira željene funkcionalnosti, i dizajnerski prilagođena korisničkim zahtjevima. To znači da je moguće generisanje prototipova aplikacija koji su vrlo bliski ili identični krajnjoj verziji implementirane aplikacije informacionog sistema.

U petom poglavlju pod nazivom *Automatizovano generisanje izvršnih softverskih specifikacija* opisani su postupci i algoritmi za generisanje izvršnih softverskih specifikacija, odnosno prototipova aplikacija informacionih sistema. Ti algoritmi implementirani su u okviru alata IIS*Case. Postupak generisanja prototipova aplikacija sastoji se iz dvije faze:

- generisanje podšema tipova formi aplikativnog sistema i
- generisanje aplikacije.

U prvoj fazi automatski se generišu podšeme koje odgovaraju tipovima formi koje su definisane u okviru specifikacije aplikativnog sistema. Ova aktivnost modelovana je i implementirana upotrebom kompleksnih algoritama, koji su u radu navedeni. Generisane podšeme predstavljaju neophodne ulazne specifikacije za drugu fazu, u okviru koje se generiše ne samo korisnički interfejs aplikacije, već i komunikacija sa odgovarajućom bazom podataka.

U šestom poglavlju dati su zaključci doktorskog rada.

U prilogu A prezentovani su algoritmi za generisanje podšema tipova formi. Ovi algoritmi su implementirani u okviru alata IIS*Case kao dio modula za automatsko generisanje izvršnih softverskih specifikacija. U prilogu B prikazan je relevantan dio specifikacije aktuelne verzije repozitorijuma razvojnog okruženja IIS*Studio. U prilogu C data je *User Interface Markup Language* (UIML) specifikacija prototipa korisničkog interfejsa.

2. Pregled stanja u oblasti

Upotreba modela odavno je prepoznata kao dobra praksa tokom procesa razvoja informacionih sistema. Primjeni u razvoju softverskih sistema, prethodi dugogodišnja praksa primjene ovakvog pristupa tokom razvoja složenih hardverskih sistema. Korišćenje modela u procesu razvoja informacionog sistema prisutno je već decenijama. Međutim, sa pojavom metodologija i alata koji omogućavaju kreiranje softverskih modela na visoko apstraktnom nivou, pristup koji podrazumijeva upotrebu modela kao ključne i nerazdvojive komponente razvojnog procesa, postao je jedan od vodećih pristupa u razvoju softverskih sistema.

U ovom poglavlju dat je pregled nekih od najvažnijih pristupa u projektovanju informacionih sistema koji se zasnivaju na modelima. Opisan je i dio koncepata, standarda i jezika koje dati pristupi definišu. Poseban osvrt dat je na koncept transformacije modela u cilju uvođenja standardizacije i obezbjeđivanja interoperabilnosti između različitih razvojnih okruženja.

2.1. Model-Driven Architecture

Jedan od pristupa koji daju modelima vodeću ulogu u razvojnom procesu je *Model-Driven Architecture* ili skraćeno MDA [6, 21, 39]. Upotreba ovog pristupa predstavlja mogući način da se umanji kompleksnost razvojnog procesa, da se postignu visoki nivoi ponovne upotrebljivosti postojećih programskih modula i značajno smanji napor uložen u projekte razvoja softvera. Alati koji podržavaju MDA olakšavaju projektantu da premosti jaz između analize projektnih zahtjeva i konkretne implementacije sistema.

MDA predstavlja takav pristup modelovanju informacionih sistema koji razdvaja specifikaciju funkcionalnosti i struktura sistema od specifikacije konkretne implementacije na izabranoj platformi. Specifikacije sistema opisane su modelima. Glavna karakteristika upotrebe modela u MDA pristupu je portabilnost, tj. mogućnost da jedan model može biti realizovan na različitim platformama, odnosno implementiran upotrebom različitih tehnologija. MDA koncept omogućava povezivanje modela različitih aplikativnih sistema, njihovu integraciju i interoperabilnost. Na taj način, razvoj apstraktnih specifikacija sistema postaje nezavisan od konkretnog izbora platforme, a samim tim i od stalnih promjena i inovacija na polju informacionih tehnologija. Upotreba novih tehnologija za potrebe

implementacije aplikativnog sistema ne zahtjeva dodatni napor za prilagođavanje specifikacija modela.

OMG grupa (*Object Management Group*) ustanovila je 2000. godine MDA pristup [39] kao novi pristup razvoju aplikacija i kreiranju specifikacija aplikacija baziranom na dva tipa modela:

- *Platform-Independent Model* (PIM model) i
- *Platform-Specific Model* (PSM model).

PIM model predstavlja specifikaciju dijela ili cijelog sistema koja ne zavisi od izabrane platforme na kojoj će sistem biti implementiran. PSM model podrazumijeva specifikaciju koja je usko povezana sa izabranom platformom i u sebi uključuje detalje specifične za konkretnu tehnologiju. [38]

MDA specifikacija sistema sastoji se od:

- jednog ili više PIM modela,
- jednog ili više PSM modela i
- skupa definicija interfejsa koje opisuju kako je osnovni PIM model implementiran na različitim platformama.

MDA aplikacija sastoji se od:

- jednog ili više PIM modela,
- jednog ili više PSM modela i
- implementacije aplikacije na svakoj od izabranih platformi.

U MDA pristupu životni ciklus softverskog sistema predstavljen je transformacijama modela koje se ostvaruju kroz faze životnog ciklusa i koje, na kraju, rezultiraju u konkretnoj implementaciji sistema. Transformacije jednog PIM modela u drugi PIM model, tzv. PIM-u-PIM transformacije modela, prisutne su tokom faza analize i projektovanja. Transformacije PIM modela u PSM modele, tzv. PIM-u-PSM transformacije, izvršavaju se tokom transformacije specifikacija projekta u implementaciju sistema. Takođe postoje i transformacije PSM modela u PIM modele, tzv. PSM-u-PIM transformacije koje se primjenjuju u procesu reverznog inženjeringa. [38, 53]

Postoji nekoliko ključnih jezika i/ili okruženja u okviru MDA pristupa od kojih su najvažniji: UML, XMI, QVT, OCL i MOF. *Unified Modeling Language* (UML) postao je vodeći jezik za projektovanje modela i arhitekture sistema. *XML Metadata Interchange* (XMI) je standard koji se koristi za reprezentaciju UML modela i omogućavanje interoperabilnosti između MDA alata različitih proizvođača. *Query/View/Transformation* (QVT) je jezik sa softverskim okruženjem koje omogućava kreiranje različitih transformacija modela. *Object Constraint Language* (OCL) je jezik koji se koristi za specifikaciju ograničenja u modelima. *Meta Object Facility* (MOF) definiše konstrukcione elemente neophodne za kreiranje reprezentacije metamodela. Svi koncepti obuhvaćeni MDA pristupom, zasnovani su na MOF-u.

Jedna od najvažnijih prednosti MDA pristupa je da razdvaja detalje implementacije od poslovne logike aplikativnog sistema. Na taj način nije neophodno ispočetka specificirati funkcionalnosti sistema svaki put kada se želi primijeniti nova tehnologija. Za razliku od

nekih drugih pristupa koji mogu biti usko orijentisani na konkretnu tehnologiju, MDA podrazumijeva modelovanje poslovnih pravila i funkcionalnosti sistema samo jednom. PIM modeli se, kroz PSM modele, preslikavaju u izvedbe na izabranim, konkretnim platformama. Na tržištu postoji veliki broj alata koji podržavaju ova preslikavanja i pružaju široki izbor platformi na kojima sistem može biti implementiran. [39]

2.2. Modelovanje softverskih sistema korišćenjem UML jezika

Modelovanje softverskih sistema predstavlja dizajniranje strukture i funkcionalnosti softverskih sistema koje prethodi samom kreiranju programskog koda. Predstavlja neodvojivu komponentu razvojnih projekata velikih korporativnih softverskih sistema, ali se isto tako koristi i u slučaju razvoja manjih softverskih sistema i aplikacija.

Upotrebom modela, softverski sistem se razvija na visoko apstraktnom nivou. Skrivajući detalje, model daje različite slike sistema. S jedne strane to može da bude generalna slika sistema (pogled na sistem koji daje informacije na kojoj je platformi implementiran, kako je povezan sa drugim sistemima, itd.), dok s druge strane model može da bude fokusiran na određene aspekte strukture sistema (poslovni procesi koje aplikativni sistem automatizuje, poslovna pravila i sl.).

OMG grupa ustanovila je UML 2.0 sa ciljem da pomogne projektantima da specificiraju, vizuelizuju i dokumentuju modele softverskih sistema. UML se takođe koristi za poslovno modelovanje i modelovanje sistema koji nisu softverski.

Danas postoji veliki broj UML alata na tržištu. Neke od osnovnih funkcionalnosti ovih alata su: podrška za izradu UML dijagrama (dijagrami struktura, ponašanja i interakcija), podrška za direktno i reverzno inženjerstvo, podrška za dokumentovanje i praćenje verzija, itd. Neki od popularnijih UML alata su:

- *Rational Rose - IBM Software*,
- *Rational Software Architect - IBM Software* (UML 2.0),
- *Borland Together* (UML 2.0),
- *Poseidon for UML – Gentleware* (UML 2.0),
- *Enterprise Architect - Sparx Systems* (UML 2.1),
- *MagicDraw UML - No Magic, Inc.* (UML 2.0),
- *System Architect*,
- *PowerDesigner - Sybase* (UML 2.0),
- *Altova UModel* (UML 2.1).

UML modeli softverskog sistema mogu biti zavisni ili nezavisni od izabrane platforme. MDA pristup u razvoju sistema koristi obje forme UML modela (kao PSM i PIM modele).

Proces pribavljanja i analize projektnih zahtjeva i njihovo ugrađivanje u projekat sistema je složen proces, i trenutno postoji veliki broj tehnologija koje definišu formalne procedure koje podržavaju ovaj proces. Jedna od glavnih karakteristika UML-a je da ne zavisi od izabrane metodologije, tj. bez obzira na to koja metodologija se koristi tokom faza analize i projektovanja, rezultati mogu biti opisani UML-om.

XML Metadata Interchange (XMI) je MDA standard koji se koristi za definisanje, razmjenu, manipulaciju i integraciju XML podataka i objekata, transformaciju jednog modela u drugi [41]. XMI je zasnovan na *Extensible Markup Language* (XML) [56]. XMI definiše pravila po kojima se može generisati šema za svaki validni XMI-prenosiv MOF metamodel. XMI omogućava transformaciju MOF specifikacija u XML specifikacije.

Uz XMI podršku moguće je prenošenje UML modela iz jednog UML alata u drugi radi daljeg "razrađivanja", prenos u repozitorijum ili prelazak na sljedeći korak u razvojnom procesu. Ovo su prednosti standardizacije alata za modelovanje i razvojnih platformi bez koje interoperabilnost ne bi bila moguća. [43, 51]

2.3. Meta-modelovanje

OMG grupa zvanično je objavila *MetaObject Facility* (MOF) jezik u Avgustu 2004. godine definišući da svi jezici za modelovanje korišćeni u MDA pristupu razvoju softverskih sistema moraju biti opisani MOF jezikom.

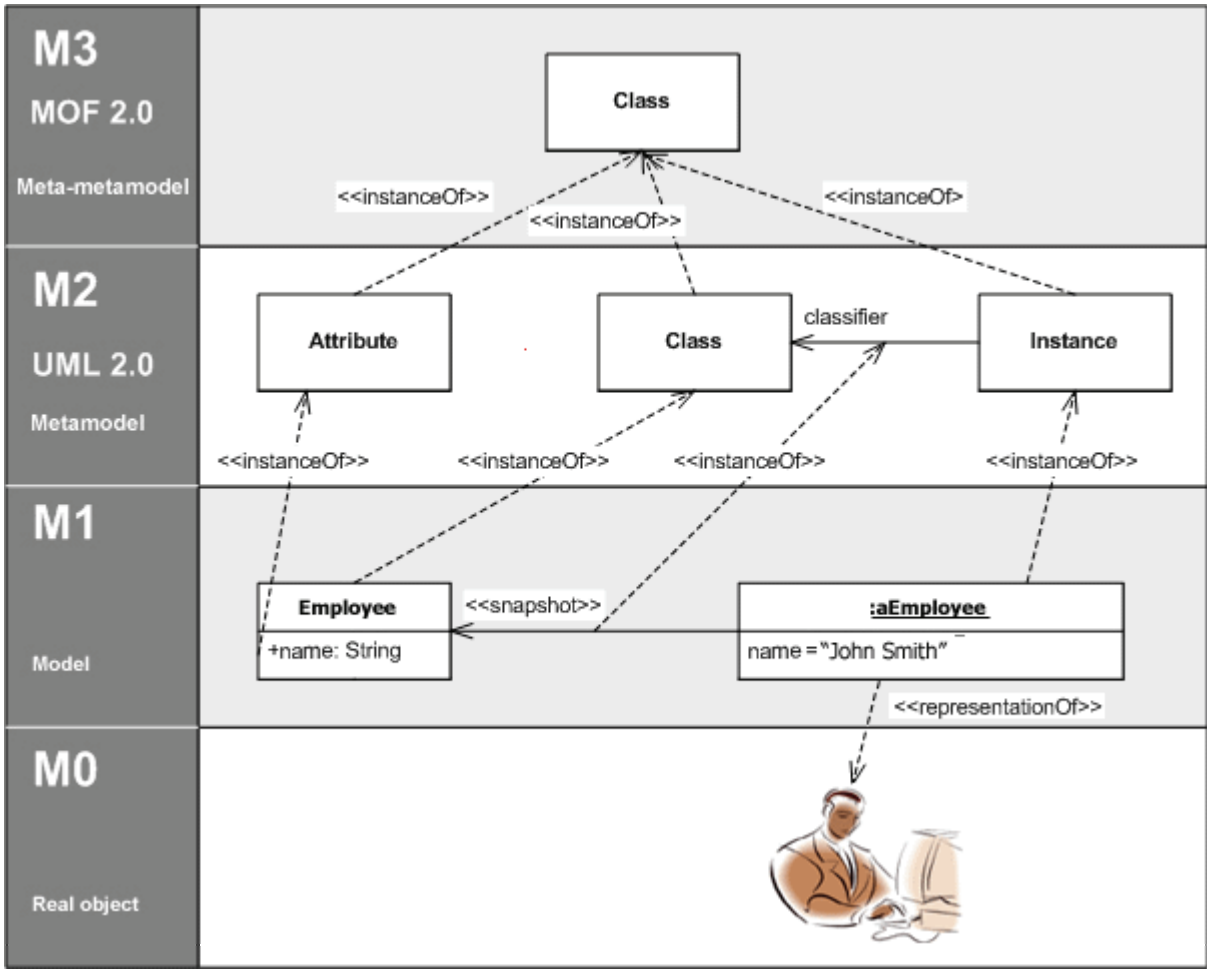
U MOF terminologiji izraz "metapodaci" označava takvu vrstu podataka čija je uloga da opišu druge podatke. Analogno, izraz "metamodel" označava model određene vrste metapodataka.

I UML i MOF su bazirani na metamodel arhitekturi koja se sastoji od hijerarhijski organizovanih konceptualnih nivoa. Elementi jednog konceptualnog nivoa opisuju elemente konceptualnog nivoa koji je u ovoj hijerarhiji direktno podređen datom nivou.

MOF arhitektura meta podataka predstavlja hijerarhiju četiri meta-nivoa (slika 2.3.1):

- M0 – Nivo realnih objekata. Čine ga objekti iz realnog svijeta koje želimo da opišemo.
- M1 – Nivo modela. Predstavlja metapodatke koji opisuju realne objekte. Metapodaci se neformalno agregiraju kao modeli.
- M2 – Nivo metamodela. Sastoji se od meta-metapodataka koji opisuju strukturu i semantiku metapodataka. Meta-metapodaci su organizovani u metamodele koji na apstraktnom nivou opisuju različite vrste podataka. Ovaj nivo prestavlja "apstraktan jezik".
- M3 – Nivo meta-metamodel opisuje strukturu i semantiku meta-metapodataka. Ovaj nivo prestavlja "apstraktan jezik" za definisanje različitih vrsta metapodataka.

U okviru ove arhitekture, MOF meta-metamodel postaje jezik kojim se specificira UML metamodel. UML metamodel definiše UML modele, dok UML specificira funkcionalnosti sistema. Na taj način UML metamodel može biti opisan kao instanca MOF meta-metamodela, a UML model kao instanca UML metamodela.



Slika 2.3.1. *MOF arhitektura*

Broj meta-nivoa u MOF arhitekturi nije fiksiran. U zavisnosti od implementacije MOF-a, može biti manje ili više od četiri nivoa. Štaviše, MOF specifikacija ne zahtijeva isključivo razlikovanje meta-nivoa. MOF meta-nivoi predstavljaju samo konvenciju koja definiše način na koji su opisane veze između različitih podataka i meta-podataka. U tom smislu ni model nije isključivo ograničen samo na jedan meta-nivo.

MOF model opisuje samog sebe, odnosno MOF model se može opisati sopstvenim konstrukcionim elementima za metamodelovanje. Ova činjenica znači da je MOF dovoljno izražajan jezik za široku praktičnu upotrebu u oblasti metamodelovanja. [42]

2.4. Transformacije modela

OMG grupa je, uspostavljajući MOF pristup, obezbjedila da se metapodaci specificiraju na standardizovan način, čime se ispunjava preduslov za automatizovanje različitih transformacija modela između različitih razvojnih okruženja. U ovoj tački biće predstavljeni različiti pristupi, odnosno jezici, koji podržavaju postupke transformacije modela. Istraživanja u oblasti transformacija modela važna su za budući razvoj alata razvojnog okruženja IIS*Studio. Primjenom ovih postupaka, bilo bi omogućeno da se specifikacije modela kreiranih alatom IIS*Case transformišu u specifikacije koje se mogu uvoziti u druge MDA alate. I obrnuto, specifikacije modela kreiranih upotrebom nekih od alata, koji su danas u

širokoj upotrebi tokom procesa projektovanja informacionih sistema, bilo bi moguće importovati i dalje razvijati upotrebom alata okruženja IIS*Studio.

2.4.1. OCL jezik za specifikaciju ograničenja

UML koncepti sa dijagramskom reprezentacijom modela nisu sami po sebi dovoljni da bi se kroz njih opisali svi relevantni aspekti specifikacije, odnosno putem ovih koncepata moguće je izraditi semi-formalne, ali ne i formalne specifikacije. Često postoji potreba da se opišu dodatna ograničenja na objektima modela. U praksi se pokazalo da opisivanje ovih ograničenja "prirodnim jezikom" često rezultira dvosmislenošću i nejednoznačnošću. U cilju kreiranja nedvosmislenih i jednoznačnih ograničenja, bilo je potrebno kreirati formalni jezik.

Object Constraint Language (OCL) predstavlja formalni, tekstualni jezik koji omogućava precizno definisanje ograničenja i upita na bilo kom MOF modelu. Iako je kreiran kao formalan jezik, OCL je lako čitati i pisati. U pitanju je jezik za kreiranje specifikacija, i zato nije moguće, kao kod programskih jezika, njime kreirati (programirati) poslovnu logiku sistema, pokrenuti proces ili aktivirati operaciju koja nije tipa upita. Rezultat evaluiranja OCL izraza je vraćena vrijednost. Ovo znači da OCL izraz ne može da utiče na stanje sistema, tj. ne može da promjeni model uz koji je specificiran. [44]

OCL je prvobitno kreiran kao deklarativni jezik za opisivanje pravila koja se primjenjuju na UML modele i u suštini je i sam predstavljao dio UML specifikacije, odnosno formalnu ekstenziju jezika UML. Danas se OCL primjenjuje na sve MOF metamodele, uključujući i UML.

OCL je ključna komponenta jednog od najvažnijih standarda MDA pristupa - transformacije modela. U [8] je predstavljen jedan pristup implementaciji transformacije modela koji uvodi pojam ugovora za transformaciju (originalni naziv na engleskom: *model transformation contracts*). Ugovorima je omogućeno specificiranje transformacije modela nezavisno od odabrane platforme. Specifikacija ugovora za transformaciju je bazirana na standardnim konceptima UML-a i OCL-a. Cilj specificiranih ugovora je da definišu šta radi transformacija modela, pod kojim uslovima može biti primjenjena i šta se može očekivati kao rezultat transformacije. U [8] su ugovori predstavljeni kao važna komponenta procesa razvoja softvera, zasnovanom na modelima, jer njihova primjena može da unaprijedi specificiranje i dokumentovanje transformacija, validaciju i testiranje transformacija.

Ugovori za transformaciju modela definisani su kao strukture nad sljedećim skupovima:

- skup ograničenja koja moraju biti ispunjena kako bi model mogao biti kandidat za izvorni model transformacije,
- skup ograničenja koja moraju biti ispunjena kako bi se model mogao smatrati validnim ciljnim modelom transformacije i
- skup ograničenja na vezama između različitih elemenata izvornog i ciljnog modela.

U [7] je opisan jedan drugi pristup koji predlaže postupke za verifikaciju UML modela, a koji se, takođe, zasnivaju na ugovorima i OCL ograničenjima.

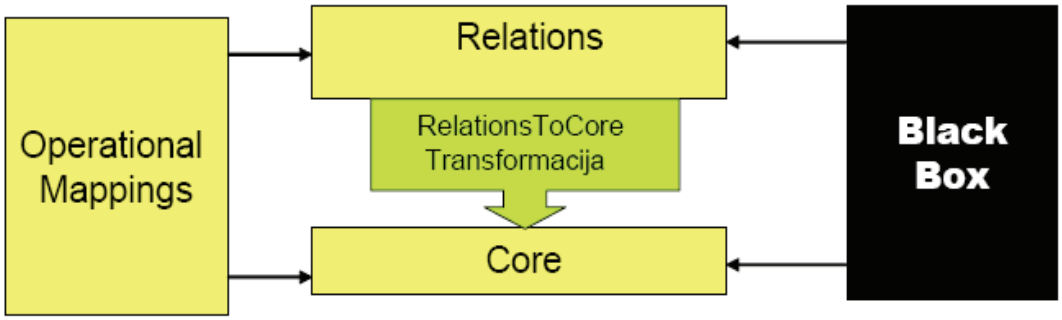
2.4.2. QVT jezik za transformaciju modela

Transformacija modela je proces u kojem se neki model pod nazivom Ma, kojem odgovara metamodel MMa, konvertuje u model pod nazivom Mb kojem odgovara metamodel MMb. U slučaju da važi MMa=MMb, transformacija se smatra endogenom. U protivnom transformacija je egzogena.

OMG grupa ustanovila je QVT kao jezik za transformaciju modela. MOF QVT specifikacija data je u [40]. QVT integriše koncepte OCL 2.0. Struktura QVT-a organizovana je u više nivoa kao što je prikazano na slici 2.4.2.1.

QVT definiše tri DSL jezika:

- *Relations*,
- *Core* i
- *Operational Mappings*.



Slika 2.4.2.1. *QVT arhitektura*

Relations i *Core* predstavljaju deklarativne jezike. *Relations* specifikacije je moguće transformisati u *Core* specifikacije.

Relations predstavlja jezik za specifikaciju veza između MOF objekata i podržava njihovo uparivanje po zadatom obrascu. *Relations* definiše transformacije između modela kandidata. Modeli kandidati, ili kandidatski modeli, predstavljaju ulazne parametre transformacije. Na slici 2.4.2.2 data je deklaracija jednog primjera *Relations* transformacije, sa nazivom `umlRdbms`. Definisana su dva modela kandidata, `uml` i `rdbms`. `SimpleUML` je paket definisan kao metamodel modela kandidata `uml`, dok je `SimpleRDBMS` paket definisan kao metamodel modela kandidata `rdbms`. Transformacija može biti izvršena tako da samo provjerava konzistentnost jednog modela kandidata u odnosu na drugi, ili da mijenja jedan od modela kandidata da bi konzistentnost bila zadovoljena.

```
transformation umlRdbms (uml : SimpleUML, rdbms : SimpleRDBMS) {…}
```

Slika 2.4.2.2. *Deklaracija Relations transformacije*

Relations transformacije sastoje se iz skupova relacija koje definišu skup ograničenja koja moraju da zadovoljavaju elementi modela kandidata. Relacije, specificirane definicijom dva ili više domena i parom `when` i `where` predikata, definišu vezu koja mora da važi između elemenata modela kandidata. Na slici 2.4.2.3 dati su primjeri relacija `PackageToSchema` i

`ClassToTable`. U primjeru relacije `PackageToSchema`, deklarisana su dva domena koja uparuju elemente modela `uml` i `rdbms`. Svaki od domena specificira jednostavan obrazac po kojem upareni elementi `p` tipa `Package`, i `s` tipa `Schema`, moraju imati isto ime. Specifikacija relacije `ClassToTable` sadrži predikate `when` i `where`. Predikat `when` definiše uslove pod kojim data relacija važi. U skladu s tim, relacija `ClassToTable` važi samo kada važi i relacija `PackageToSchema`. Predikat `where` definiše uslove koje moraju da zadovoljavaju svi elementi modela koji učestvuju u relaciji, što znači da ako relacija `ClassToTable` važi, tada relacija `AttributeToColumn` takođe mora da važi.

```
relation PackageToSchema
{
      domain uml p:Package {name=pn}
      domain rdbms s:Schema {name=pn}
}

relation ClassToTable
{
      domain uml c:Class {
            namespace = p:Package {},
            kind='Persistent',
            name=cn
      }
      domain rdbms t:Table {
            schema = s:Schema {},
            name=cn,
            column = cl:Column {
                  name=cn+'_tid',
                  type='NUMBER'},
            primaryKey = k:PrimaryKey {
                  name=cn+'_pk',
                  column=cl}
      }
      when {
            PackageToSchema(p, s);
      }
      where {
            AttributeToColumn(c, t);
      }
}
```

Slika 2.4.2.3. *Primjeri Relations relacija*

Core je deklarativni jezik koji podržava uparivanje nad skupom promjenljivih, provjerom zadovoljenosti određenih uslova u odnosu na skup modela. *Core* specifikacija je jednako moćna kao *Relations*, ali je nešto jednostavnije sintakse. S druge strane specifikacije *Relations* transformacije jesu složenije, ali su i deskriptivnije. Specifikacija *Core* transformacije deklariše ograničenja koja moraju da važe među skupovima modela kandidata i *trace* modela. *Trace* model opisuje šemu izlaznog rezultata transformacije u *Relations* specifikaciji i tipizuje jedan ulazni parametar za *Core* transformaciju. *Trace* klase u *trace* modelu predstavljaju MOF klase koje ukazuju na objekte modela obuhvaćenih transformacijama. Prilikom izvršavanja *Relations* transformacije *trace* klase i njihove instance se generišu implicitno, kako bi se sačuvale kreirane veze između modela. Kod *Core* transformacija, *trace* klase specificiraju se eksplicitno kao MOF modeli, a kreiranje i brisanje *trace* objekata je definisano na uobičajen način, kao kod standardnih klasa.

Core transformacija može biti izvršena na dva načina. Prvim načinom provjerava se zadovoljenost uslova koji moraju da budu ispunjeni između modela kandidata i *trace* modela. Drugi način podrazumijeva specificiranje jednog od modela kandidata kao ciljnog modela, a zatim izvršavanje transformacije, koja prvo provjerava ispunjenost uslova, a onda u slučaju da neko od ograničenja nije zadovoljeno, forsira izmjene na ciljnom i *trace* modelu kako bi konzistentnost bila postignuta.

Operational Mappings predstavlja proceduralni jezik koji proširuje *Relations* i *Core*. Njegova sintaksa obuhvata konstrukcione elemente koji se tipično koriste u proceduralnim programskim jezicima kao što su npr. petlje, uslovi i sl. Jezik *Operational Mappings* omogućava definisanje transformacija primjenom potpuno proceduralnog pristupa koji uključuje specificiranje operativnih transformacija (*operational transformations*). U tzv. hibridnom pristupu, *Operational Mappings* omogućava nadogradnju transformacija proceduralnim operacijama koje implementiraju relacije. Na slici 2.4.2.4. dat je primjer operativne transformacije koja kao ulazni parametar uzima model `uml` tipa `UML`, a kao rezultat daje model `rdbms` tipa `RDBMS`. U datom primjeru funkcija `main()` prvo učitava objekte tipa `Package`, a zatim primjenjuje operaciju transformacije (*mapping operation*) `packageToSchema` na svaki učitani objekat. Operacije transformacije definišu na koji način se elementi jednog ili više izvornih modela transformišu u elemente jednog ili više ciljnih modela.

```
transformation Uml2Rdbms(in uml:UML,out rdbms:RDBMS) {
      main() {
            uml.objectsOfType(Package)->map packageToSchema();
      }
      …
}
```

Slika 2.4.2.4. *Primjer operativne transformacije*

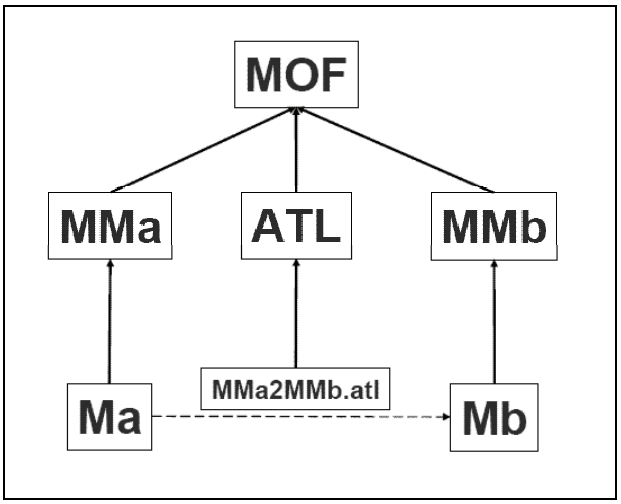
Black-box predstavlja dio arhitekture QVT-a koji omogućava transformacije zasnovane na drugim jezicima (npr. XSLT, XQuery).

2.4.3. ATL okruženje za transformaciju modela

ATLAS Transformation Language (ATL) predstavlja alat i jezik za transformaciju modela koji podržava QVT specifikaciju [15]. ATL je proizvod *Eclipse* fondacije, neprofitabilnog društva čiji projekti su usmjereni ka razvoju *open-source* sistema i alata. U oblasti inženjerstva vođenog modelima (*Model-Driven Engineering* ili skraćeno MDE) [5, 16, 52], ATL, kao *open-source* proizvod, ima veliki broj korisnika. Na slici 2.4.3.1 predstavljena je opšta šema ATL transformacije izvornog modela Ma kojem odgovara metamodel MMa u ciljni model Mb kojem odgovara metamodel MMb.

ATL transformacioni program sastavljen je od pravila koja definišu kako se elementi izvornog modela prepoznaju i obrađuju na način koji rezultira kreiranjem elemenata ciljnih modela. Osim osnovne funkcije transformacije modela, ATL definiše i dodatni model koji pruža funkcionalnost specifikacije različitih upita na modelima. [15]

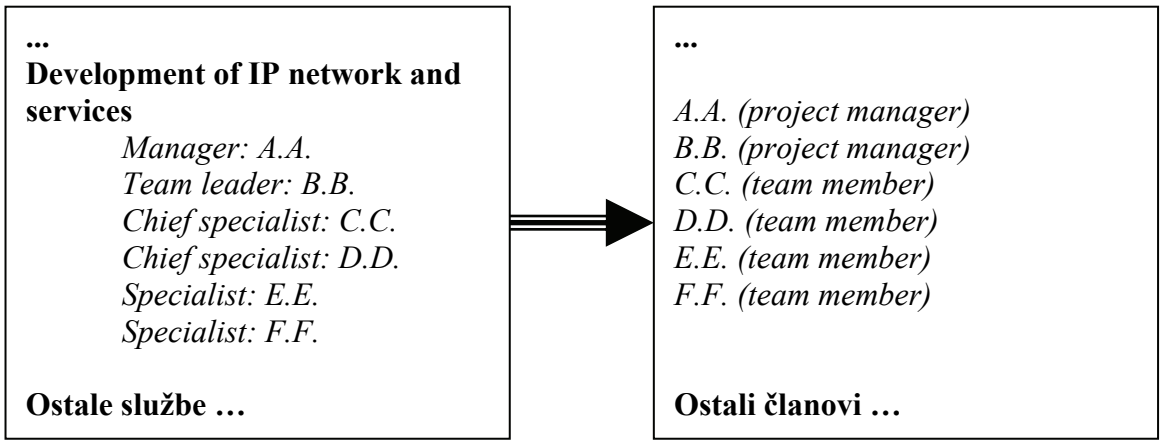
Francuski nacionalni institut za istraživanja u oblasti računarskih nauka i primjenjene matematike *Institut national de recherche en informatique et en automatique* (INRIA), definisao je *Kernel Meta Meta Model* (KM3) kao jezik za definisanje metamodela i specificiranje sintakse *Domain Specific Language* (DSL) jezika, koji ne zavisi od izabrane platforme. *Eclipse* platforma implementira koncepte KM3 jezika za potrebe specifikacije metamodela koji odgovaraju modelima čiju transformaciju treba izvršiti. Specifikacija sintakse KM3 jezika data je u [4].



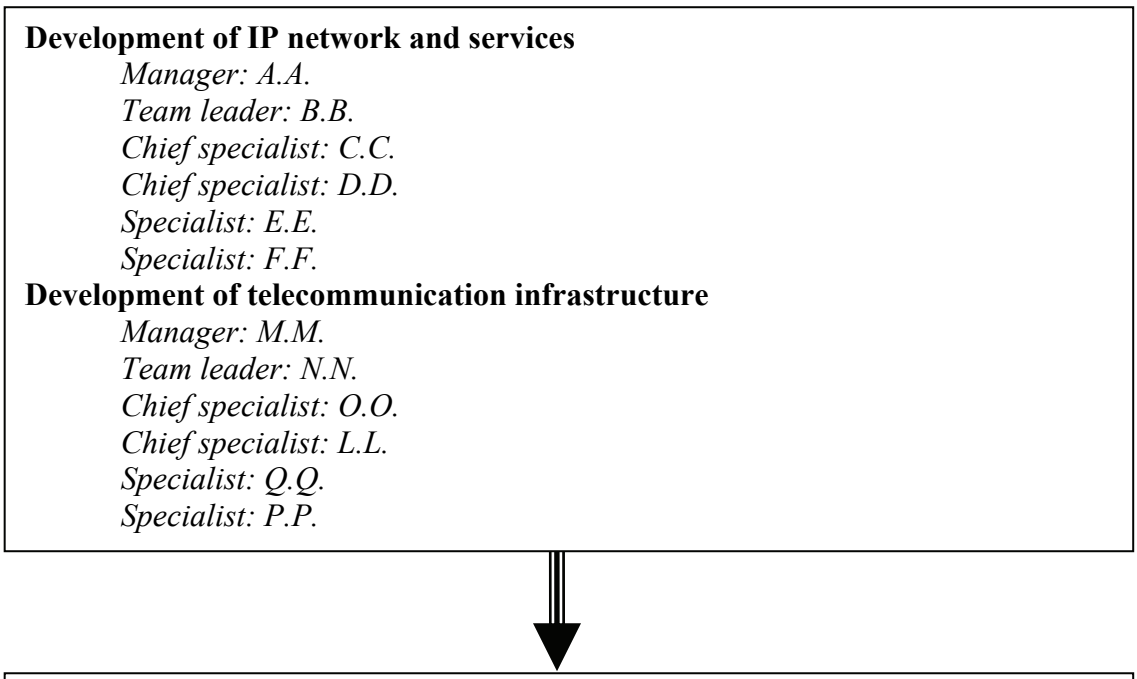
Slika 2.4.3.1. *ATL transformacija modela*

Sljedeći primjer predstavlja slučaj korišćenja transformacije jednog modela u drugi korišćenjem ATL-a.

Primjer 2.4.3.1. Primjer ilustruje slučaj transformacije modela *Departments* (model predstavlja listu službi sektora za Razvoj fiksne širokopojasne mreže Crnogorskog Telekoma) u model *Project team roles* (model predstavlja listu članova projektnih timova u sektoru za Razvoj fiksne širokopojasne mreže Crnogorskog Telekoma i njihovih mogućih uloga u projektnom timu). Inicijalno postoji tekstualni zapis koji opisuje listu službi koju želimo da transformišemo u drugi oblik tekstualnog zapisa liste članova projektnih timova (slika 2.4.3.2). Date tekstualne zapise možemo predstaviti modelima, u ovom slučaju XML specifikacijama, kao što je prikazano na slikama 2.4.3.3 i 2.4.3.4.



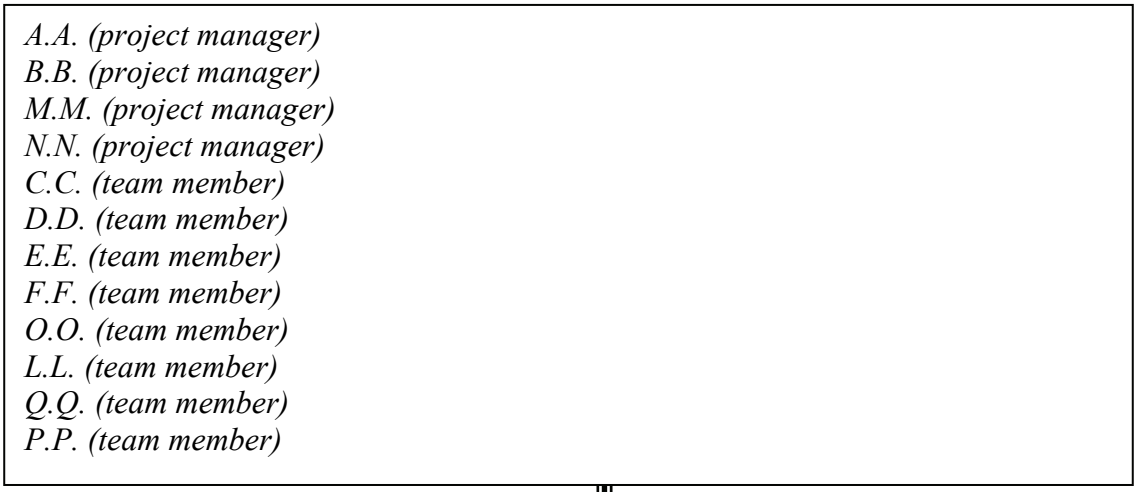
Slika 2.4.3.2. *Transformacije liste službi u listu članova projektnih timova*

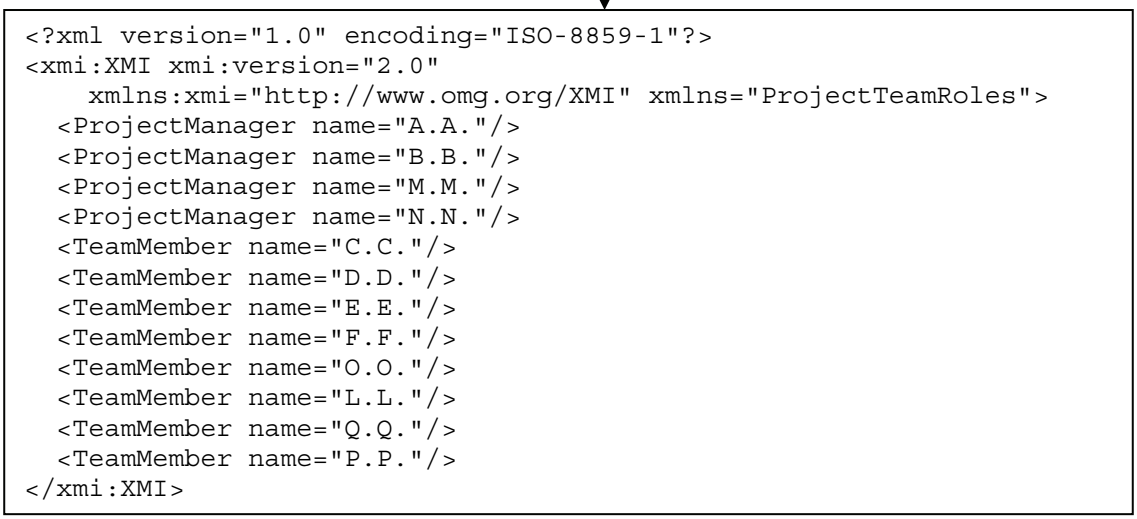



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns="Departments">
<Department name="Development of IP network and services">
    <manager fullName="A.A."/>
    <teamLeader fullName="B.B."/>
    <chiefSpecialist fullName="C.C."/>
    <chiefSpecialist fullName="D.D."/>
    <specialist fullName="E.E."/>
    <specialist fullName="F.F."/>
  </Department>
  <Department name="Development of telecommunication
infrastructure">
    <manager fullName="M.M."/>
    <teamLeader fullName="N.N."/>
    <chiefSpecialist fullName="O.O."/>
    <chiefSpecialist fullName="L.L."/>
    <specialist fullName="Q.Q."/>
    <specialist fullName="P.P."/>
  </Department>
</xmi:XMI>
```

Slika 2.4.3.3. *Model "Departments"*

Svi zaposleni aktivno učestvuju u razvojnim projektima svoje službe, a u zavisnosti od radnog mjesta koje pokrivaju mogu imati ulogu menadžera projekta ili člana projektnog tima. Rukovodioci i vođe tima imaju uloge menadžera projekata, dok glavni specijalisti i specijalisti imaju uloge članova projektnog tima.





```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI" xmlns="ProjectTeamRoles">
  <ProjectManager name="A.A."/>
  <ProjectManager name="B.B."/>
  <ProjectManager name="M.M."/>
  <ProjectManager name="N.N."/>
  <TeamMember name="C.C."/>
  <TeamMember name="D.D."/>
  <TeamMember name="E.E."/>
  <TeamMember name="F.F."/>
  <TeamMember name="O.O."/>
  <TeamMember name="L.L."/>
  <TeamMember name="Q.Q."/>
  <TeamMember name="P.P."/>
</xmi:XMI>
```

Slika 2.4.3.4. *Model ProjectTeamRoles*

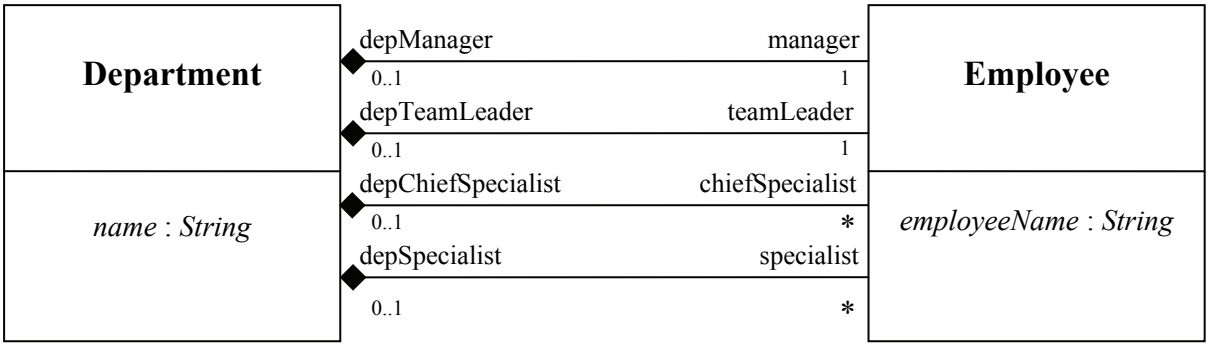
Da bismo obezbjedili transformaciju potrebno je definisati:

- KM3 specifikaciju metamodela izvornog modela (*Departments*),
- XMI specifikaciju izvornog modela,
- KM3 specifikaciju metamodela ciljnog modela (*ProjectTeamRoles*) i
- ATL specifikaciju transformacije modela.

Rezultat transformacije treba da bude XMI specifikacija ciljnog modela.

Na slikama 2.4.3.5 i 2.4.3.6 date su, redom, KM3 specifikacije metamodela *Departments* i *ProjectTeamRoles*. KM3 specifikacija sadrži definiciju jednog ili više `package` elemenata. Pri tom, jedan od `package` elemenata mora imati isti naziv kao naziv fajla koji sadrži datu specifikaciju (sa izuzimanjem ekstenzije .km3). KM3 specifikacija mora sadržati i jedan `package` elemenat koji definiše potrebne primitivne tipove podataka. Element `package` sastoji se iz skupa klasa koje formiraju model. Klase sadrže definicije atributa i referenci. Na

slici 2.4.3.5 prikazan je metamodel *Departments* i njegova KM3 specifikacija koja sadrži definiciju `package` elemenata `Departments` i `PrimitiveTypes`. `Departments` sadrži specifikaciju klasa `Department` i `Employee` kao i njihovih atributa i referenci. Svaka referenca ima definisanu suprotnu referencu (`oppositeOf`) i definisana je kao `container`, što znači da data referenca sadrži referencirane elemente. Tako npr. klasa `Department` ima definisanu suprotnu referencu `manager` tipa `Employee`, dok klasa `Employee` ima nula ili jednu suprotnu referencu `depManager` tipa `Department`. Na slici 2.4.3.6 prikazana je KM3 specifikacija metamodela *ProjectTeamRoles*, koja sadrži definiciju `package` elemenata `ProjectTeamRoles` i `PrimitiveTypes`. Elemenat `ProjectTeamRoles` tipa `package` sadrži specifikaciju apstraktne klase `Role` i njenih atributa, kao i izvedenih klasa `ProjectManager` i `TeamMember`.



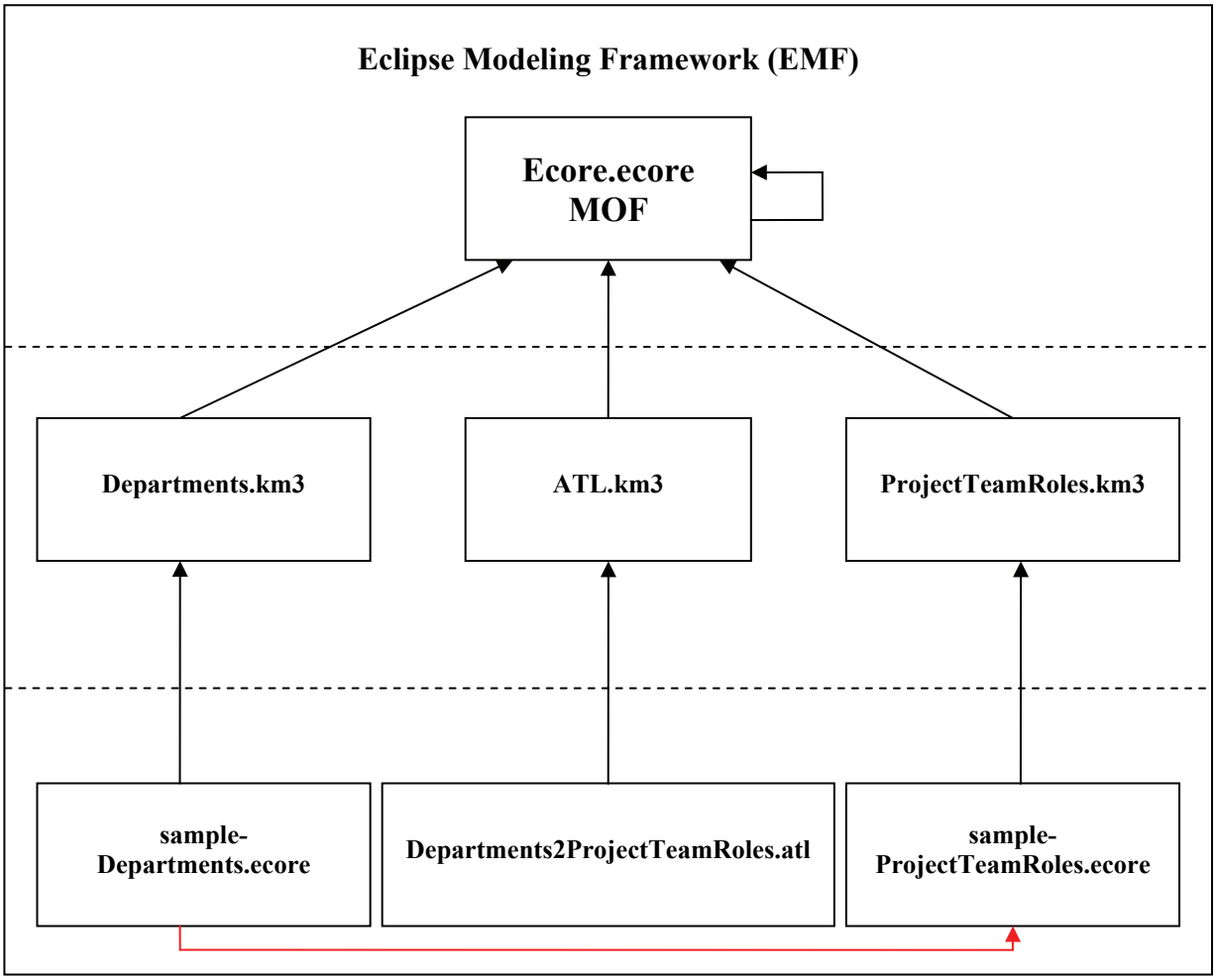

```
package Departments{
  class Department{
    attribute name: String;
    reference manager container: Employee oppositeOf depManager;
    reference teamLeader container: Employee oppositeOf depTeamLeader;
    reference chiefSpecialist[*] container: Employee oppositeOf
    depChiefSpecialist;
    reference specialist[*] container: Employee oppositeOf
depSpecialist;
  }
  class Employee{
    attribute fullName: String;
    reference depManager[0-1] container: Department oppositeOf manager;
    reference depTeamLeader[0-1] container: Department oppositeOf
    teamLeader;
    reference depChiefSpecialist[0-1] container: Department oppositeOf
    chiefSpecialist;
    reference depSpecialist[0-1] container: Department oppositeOf
    specialist;
  }
}
package PrimitiveTypes{
    datatype String;
```

Slika 2.4.3.5. *KM3 specifikacija metamodela Departments*

```
package ProjectTeamRoles{
  abstract class Role{
    attribute name: String;
    }
  class ProjectManager {} extends Role;
  class TeamMember{} extends Role;
}
package PrimitiveTypes{
  datatype String;
}
```

Slika 2.4.3.6. *KM3 specifikacija metamodela "ProjectTeamRoles"*

Sljedeći korak u procesu transformacije modela je kreiranje specifikacije ATL transformacije, tj. ATL modula `Departments2ProjectTeamRoles.atl`. Neka je specifikacija modela *Departments* sa slike 2.4.3.3 data u fajlu `sample-Departments.ecore`. Kao rezultat primjene ATL transformacije na model dat u fajlu `sample-Departments.ecore` biće kreiran fajl sample-`ProjectTeamRoles.ecore` koji sadrži specifikaciju modela *ProjectTemaRoles* sa slike 2.4.3.4.



Slika 2.4.3.7. *Arhitektura primjene ATL transformacije Departments2ProjectTeamRoles.atl*

Prije definicije transformacije modela potrebno je definisati metamodele koji odgovaraju izvornom i ciljnom modelu. Definicije metamodela date su fajlovima `Departments.km3` i `ProjectTeamRoles.km3`. Fajl `ATL.km3` sadrži definiciju, odnosno, metamodel ATL jezika, dok `Ecore.ecore` predstavlja definiciju meta-metamodela (slika 2.4.3.7). Sljedeći korak je kreiranje ATL modula koji specificira ATL transformaciju. Na slici 2.4.3.8 dato je zaglavlje ATL modula u kojem se imenuje transformacioni model ("*Departments2ProjectTeamRoles*") i varijable koje definišu izvorni i ciljni model (polja "IN" i "OUT") kao i odgovarajuće metamodele ("*Departments*" i "*ProjectTeamRoles*").

```
module Departments2ProjectTeamRoles;
create OUT : ProjectTeamRoles from IN : Departments;
```

Slika 2.4.3.8. *Zaglavlje ATL modula*

Nadalje, u okviru specifikacije ATL modula definišu se tzv. *helper* funkcije i pravila. *Helper* predstavlja pomoćnu funkciju koja izračunava rezultat na osnovu odgovarajućeg pravila. Pojam je ekvivalentan pojmu metoda u *Java* okruženju. *Helper* funkcija `isProjectManager()` definisana na slici 2.4.3.9 kao rezultat daje ulogu zaposlenog u projektnom timu na osnovu njegovog radnog mjesta. Specifikacija pravila `Employee2Manager` i `Employee2TeamMember`, koje koristi definisana funkcija, prikazana je na slici 2.4.3.10.

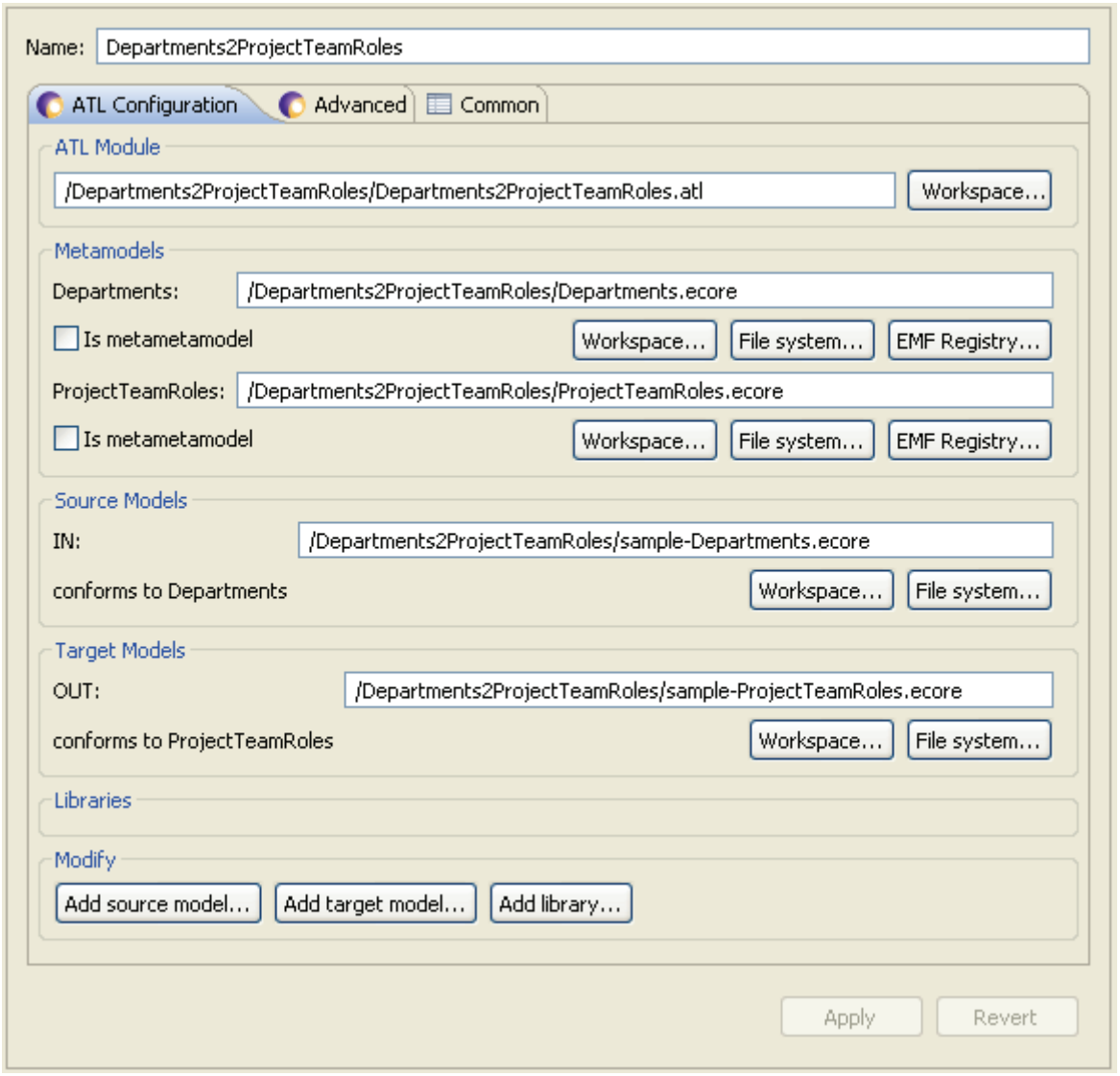
```
helper context Departments!Employee def: isProjectManager() : Boolean =
      if not self.depManager.oclIsUndefined() then
            true
      else
            if not self.depTeamLeader.oclIsUndefined()
            then
                  true
            else
                  false
            endif
      endif;
```

Slika 2.4.3.9. *Helper funkcija isProjectManager()*

```
rule Employee2Manager {
      from
            s : Departments!Employee (s.isProjectManager())
      to
            t : ProjectTeamRoles!ProjectManager(
                  name <- s.fullName
            )
}
rule Employee2TeamMember {
      from
            s : Departments!Employee (not s.isProjectManager())
      to
            t : ProjectTeamRoles!TeamMember(
                  name <- s.fullName
            )
}
```

Slika 2.4.3.10. *Definicija pravila Employee2Manager i Employee2TeamMember*

Na slici 2.4.3.11 prikazana je forma za zadavanje ATL transformacije putem koje se specificiraju putanje do izvornog i ciljnog modela (polja IN i OUT), kao i putanje do odgovarajućih metamodela (panel *Metamodels*). Izvršavanjem ATL transformacije "*Departments2ProjectTeamRoles*", prikazane na slici, dobijamo ciljni model, tj. XMI specifikaciju sa slike 2.4.3.4.



Slika 2.4.3.11. *Forma za specifikaciju ATL transformacije*

2.5. Model-Driven Software Development (MDSD)

Jedan od pristupa koji podrazumijevaju ključnu ulogu modela tokom razvoja informacionog sistema je *Model-Driven Software Development* (MDSD). Manje precizan, ali češće upotrebljavan naziv je i *Model-Driven Development* (MDD). Danas MDSD pristup ima veliku primjenu i značaj koji će u budućnosti biti još veći, što predstavlja prirodan nastavak razvoja programiranja kakvog znamo danas.

Primjena modela u razvoju softvera ima dugogodišnju tradiciju, i pogotovo postaje popularna od pojave UML-a. U pristupima zasnovanim na modelima (*model-based approaches*) u kojima model nije formalno povezan sa implementacijom aplikativnog sistema, već ima ulogu preciznog i kompletnog dokumentovanja razvojnog procesa, pojavljuju se mnogi nedostaci. Softverski sistemi nisu statični i podložni su mnogim promjenama tokom svog životnog ciklusa, zbog čega je potrebno vrlo često prilagođavati i dokumentaciju. Ovo može biti vrlo složen proces u zavisnosti od nivoa detaljnosti dokumentacije, a postoji i opasnost od nekonzistentnost usled čestih promjena. Zbog toga mnogi programeri tretiraju modele kao "međuproizvod" razvojnog procesa.

MDSD pristup na potpuno drugačiji način primjenjuje modele. Modeli ne sačinjavaju dokumentaciju već se "smatraju" programskim kodom jer je njihova implementacija automatizovana. Modeli preuzimaju centralnu i aktivnu ulogu u razvojnom procesu.

Svaki konkretni domen primjene je od velikog značaja za primjenu MDSD pristupa. Njegov cilj je da definiše domenski orjentisane (*domain-specific*) apstrakcije koje će biti raspoložive za formalno modelovanje. Ovo predstavlja veliki potencijal u oblasti automatske proizvodnje softvera, što može značajno povećati produktivnost. Takođe, ovo može imate pozitivnog uticaja i na kvalitet i na mogućnosti održavanja softvera. Primjena domenski orijentisanog pristupa pri modelovanju, odnosno projektovanju i implementaciji softvera detaljno je opisana u [20].

Da bi se uspješno primijenio koncept "domenski orjentisanog modela" neophodno je ispuniti sljedeće uslove:

- Moraju biti definisani domenski orjentisani jezici (*Domain Specific Language* - DSL) kako bi omogućili formulaciju modela.
- Moraju biti definisani jezici koji bi specificirali neophodne transformacije modela u programski kod.
- Neophodno je postojanje kompajlera, generatora i transformatora koji mogu izvršavati transformacije koje generišu programski kod na dostupnim platformama.

Iako su grafički modeli često korišćeni u MDSD pristupu, njihova upotreba nije isključivo najbolje rješenje u svim situacijama. Naprotiv, primjena tekstualnih modela predstavlja jednako upotrebljivu opciju, i ovakvi modeli su pogodniji za translaciju u izvorni kod programskog jezika.

U poređenju sa MDA pristupom koji se ograničava na alate zasnovane na UML-u, MDSD je manje restriktivan. MDA se fokusira na standardizaciju modela i interoperabilnost između alata za modelovanje. MDSD se orijentiše na kreiranje modula za razvoj softvera, primjenljivih u praksi, koji mogu biti korišćenu u različitim *Model-Driven* pristupima, nezavisno od odabranog alata. [54]

DSL jezici

U pisanju programskih specifikacija, generalno se koriste dvije vrste jezika:

- *Domain-Specific Language* (DSL) jezici, i
- *General-purpose Programing Language* (GPL) jezici.

DSL jezici predstavljaju programske jezike ili jezike za kreiranje specifikacija koje su orijentisane na određeni domen primjene, način reprezentacije problema i/ili određenu tehniku rešavanja problema. Uvođenje DSL jezika korisno je kada je njime moguće opisati posebnu vrstu problema i njegovog rješavanja na mnogo jasni način nego što je to bilo moguće pomoću postojećih jezika. Jezički orijentisano programiranje (*Language Oriented Programming*) podrazumijeva kreiranje jezika specijalne namjene kao dio standardnog procesa rješavanje problema.

Programiranje upotrebom DSL jezika daleko je jednostavnije od korišćenja GPL jezika. DSL jezici su kreirani za određen domen problema, i to donosi mnoge prednosti ali i potrebu za posebnim oprezom kada je u pitanju način na koji se koriste u procesu analize i implementacije sistema.

DSL jezik je programski jezik ili jezik za kreiranje izvršnih specifikacija koji pruža, upotrebom odgovarajućih notacija i apstrakcija, moć izražavanja koja je fokusirana, a najčešće i ograničena, na konkretan domen problema. Ovi jezici potiču od određenog domena, i kao takvi naslijeđuju glavne koncepte i karakteristike datog domena koji se ugrađuju u konstrukcione elemente jezika. Ovo rezultira time da sintaksa i semantika DSL jezika biva podignuta na veći nivo apstrakcije koji je bliži ljudskom načinu razmišljanja. [19]

Tokom razvojnog procesa, eksperti iz datog domena se najčešće uključuju u fazi analize i tu se njihova uloga završava. Jedna od najznačajnih karakteristika DSL jezika jeste to što pruža mogućnost da se oni uključe u drugim, kasnijim fazama razvojnog ciklusa softverskog sistema i daju direktan doprinos u specificiranju aplikacije. DSL jezici ubrzavaju proces kreiranja programskih modula, olakšavaju fazu testiranja tokom razvojnog procesa kao i održavanje aplikativnih sistema. Upotreba DSL jezika na ovaj način postaje dobar pristup u cilju ponovne upotrebe softverskih modula. Oni su jednostavni za upotrebu i brzo se uče. [19]

DSL jezici imaju i svoje nedostatke. Jedan od najvažnijih nedostatka je veoma niska interoperabilnost, kako međusobna tako i interoperabilnost sa standardnim programskim jezicima. Ovo nije dobro jer se može reći da je kombinacija DSL i GPL jezika realna potreba prilikom razvoja softverskih sistema.

DSL jezici omogućavaju i onima koji nisu programeri da koristeći računar riješe problema iz domena koji dobro poznaju. Međutim razvoj interpretera i kompajlera koji bi omogućili implementaciju DSL programskog koda nije nimalo jednostavno. Ipak, postoje načini kojima ove poteškoće mogu biti ublažene. Jedan od najfleksibilnijih pristupa je onaj u kojem se GPL jezici koriste kao osnova za razvoj DSL jezika. [19].

2.6. IIS*Case i primjena Model-Driven pristupa

IIS*Case je alat za projektovanje informacionih sistema i generisanje izvršivih prototipova aplikacija. On podržava projektovanje šeme baze podataka, generisanje odgovarajućih SQL skripti i transakcionih programa. Kao takav, IIS*Case primjenjuje neke od principa *Model-Driven* pristupa [23, 27].

Analogno MDA alatima, projektovanje informacionog sistema u IIS*Case-u zasnovano je na dva tipa modela: onom koji ne zavisi od izabrane platforme i onom koji zavisi. Sistem se modeluje na visoko apstraktnom nivou i projektant kreira model sistema bez specificiranja

detalja vezanih za konkretnu implementaciju. Zbog toga ovaj model može biti specificiran kao PIM model [49]. Na osnovu PIM modela, IIS*Case generiše i specifikacije prototipova aplikacija informacionih sistema koja uključuje detalje vezane za konkretnu platformu na kojoj će prototip biti implementiran (PSM model).

IIS*Case podržava i druge *Model-Driven* koncepte. Za potrebe definicije izraza ograničenja na nivou konceptualne šeme baze podataka kao PIM modela, u okviru IIS*Case-a, razvijen je specijalizovani DSL jezik. Primjenom DSL-a, IIS*Case omogućava specificiranje regularnih izraza ograničenja koristeći problemski orijentisane koncepte, na grafički orijentisan način. Primjena DSL pristupa u implementaciji programske logike IIS*Case-a se u potpunosti uklapa u koncepte MDSD pristupa razvoju informacionih sistema.

Budući razvoj IIS*Case alata okrenut je ka standardizaciji i daljem povezivanju sa *Model-Driven* pristupima. IIS*Case podržava generisanje XML specifikacija aplikativnog sistema i šeme baze podataka koje ne zavise od izabrane platforme, kao i *User Interface Markup Language* (UIML) specifikaciju korisničkog interfejsa prototipa aplikacija. Upotrebom XMI standarda i transformacije modela moguće je postići interoperabilnost sa MDA alatima. Model informacionog sistema projektovan alatom IIS*Case moguće je eksportovati kao XMI specifikaciju koja ne zavisi od izabrane platforme (PIM model). Upotrebom neke od postojećih metoda transformacije modela, XMI specifikacije se mogu prevesti u XMI specifikacije PIM modela koje je moguće importovati u druge (MDA) alate (npr. UML alate). U novom okruženju modeli se dalje mogu dodatno specificirati ili direktno transformisati u PSM modele i implementirati na izabranim platformama.

Razvoj IIS*Case-a inicijalno je počeo prije skoro dvadeset godina, i to kao razvoj CASE alata za projektovanje informacionih sistema. Pomak prema upotrebi modela kao važne komponente razvoja napravljen je prije desetak godina. Konceptualni aspekti IIS*Case-a su dovoljno bogati da relativno lako mogu da implementiraju sve nove standarde na polju razvoja softvera. Uvođenje novih koncepata kao i unapređivanje postojećih ključni su elementi aktuelnog razvoja IIS*Case, koji je usmjeren ka primjeni modernih metoda i standarda softverskog inženjerstva, prije svega *Model-Driven* koncepata.

3. Pregled funkcionalnosti razvojnog okruženja IIS*Studio

Kao rezultat dugogodišnjeg istraživanja iz oblasti projektovanja i automatskog generisanja šema baza podataka i transakcionih programa nastalo je razvojno okruženje IIS*Studio 7.0.0. IIS*Studio namijenjen je projektovanju informacionih sistema i odgovarajuće šeme baze podataka, čiji je krajnji proizvod funkcionalan prototip informacionog sistema. Ovo okruženje čini skup alata koji podržavaju vrlo složene postupke generisanja. Visok nivo automatizacije procesa pomaže projektantima da bez primjene naprednih znanja iz oblasti projektovanja informacionih sistema, relativno brzo i jednostavno dođu do profesionalnog rješenja.

IIS*Studio verzija 7.0.0 objedinjava dva alata:

1. IIS*Case - alat za projektovanje informacionih sistema.
2. IIS*UIModeler – alat za modelovanje šablona korisničkog interfejsa aplikacija informacionih sistema.

3.1. Alat za projektovanje informacionih sistema IIS*Case

Svi koncepti i metodi na kojima je IIS*Case zasnovan, kao i implementirani algoritmi opisani su, između ostalih radova, u [22, 28, 33, 34, 45, 46, 48]. Alat podržava konceptualno modelovanje šema baza podataka zasnovano na konceptu tipa forme. Ovaj koncept predstavlja originalan pristup koga su autori alata IIS*Case, tokom dužeg niza godina (od početka 90-ih), razvijali i unapređivali, najvećim dijelom, na Fakultetu tehničkih nauka Univerziteta u Novom Sadu. IIS*Case, na osnovu projektovanih tipova formi, putem visoko formalnih postupaka, generiše šemu baze podataka i transakcione programe u okviru nekog aplikativnog sistema.

Tipovi formi predstavljaju generalizacije ekranskih formi putem kojih krajnji korisnici komuniciraju sa informacionim sistemom. Pristup zasnovan na konceptu tipa forme podrazumijeva da projektant, specificiranjem tipova formi, posredno zadaje polazni skup ograničenja, koji će, tokom procesa generisanja, biti transformisan u implementacionu šemu baze podataka u trećoj normalnoj formi. Generisanje šeme baze podataka je automatizovan proces. Korak po korak, IIS*Case vodi projektanta kroz procese generisanja šema relacija i grafa zatvaranja, izbora za kandidate i propagaciju primarnih ključeva, kao i generisanje međurelacionih ograničenja. IIS*Case implementira pristup nezavisnog projektovanja

pojedinačnih delova šeme baze podataka, kroz tzv. podšeme, i njihovu integraciju u jedinstvenu, relacionu šemu baze podataka.

Specifikacije tipova formi, osim informacija relevantnih za generisanje šeme baze podataka nose i dodatne informacije, koje se tiču transakcionih programa i njihovih ekranskih formi. To omogućava da se u IIS*Case-u implementira generator aplikacija. Dakle, IIS*Case podržava postupke kojima se generišu komponente budućih programskih specifikacija informacionog sistema zasnovanog na generisanoj bazi podataka. Implementacijom algoritama koji automatizuju generisanje podšeme relacione baze podataka putem formi koji su dati u [22], omogućena je implementacija komunikacije sa bazom podataka, tako da generisane izvršne softverske specifikacije informacionih sistema postaju funkcionalni transakcioni programi. Generisani programi sa odgovarajućim korisničkim interfejsom i obezbjeđenom komunikacijom sa bazom podataka podržavaju sve osnovne operacije čitanja i ažuriranje podataka.

IIS*Case, kao dobro dokumentovan alat za projektovanje informacionih sistema, pruža mogućnost generisanja velikog broja izvještaja koji se tiču, kako specifikacija zadatih od strane projektanta, tako i izvještaja sa rezultatima i međurezultatima automatizovanih procesa generisanja.

3.1.1. Strukturni elementi specifikacije projekta u IIS*Case-u

IIS*Case se oslanja na rečnik podataka (repozitorijum). Komunikacija aplikacije sa rečnikom odvija se putem ODBC konekcije. U okviru jedne konekcije projektant kreira nove, odnosno modifikuje postojeće projekte. Istovremeno projektant može raditi na više otvorenih projekata, upoređivati rezultate, kopirati elemente i njihove specifikacije iz jednog projekta u drugi. Rad sa elementima projekta olakšan je grafičkim prikazom strukture projekta u vidu projektnog stabla, gdje se posebnim ikonama označavaju različite grupe koncepata koje čine projekat, kao što je prikazano na slici 3.1.1.1.

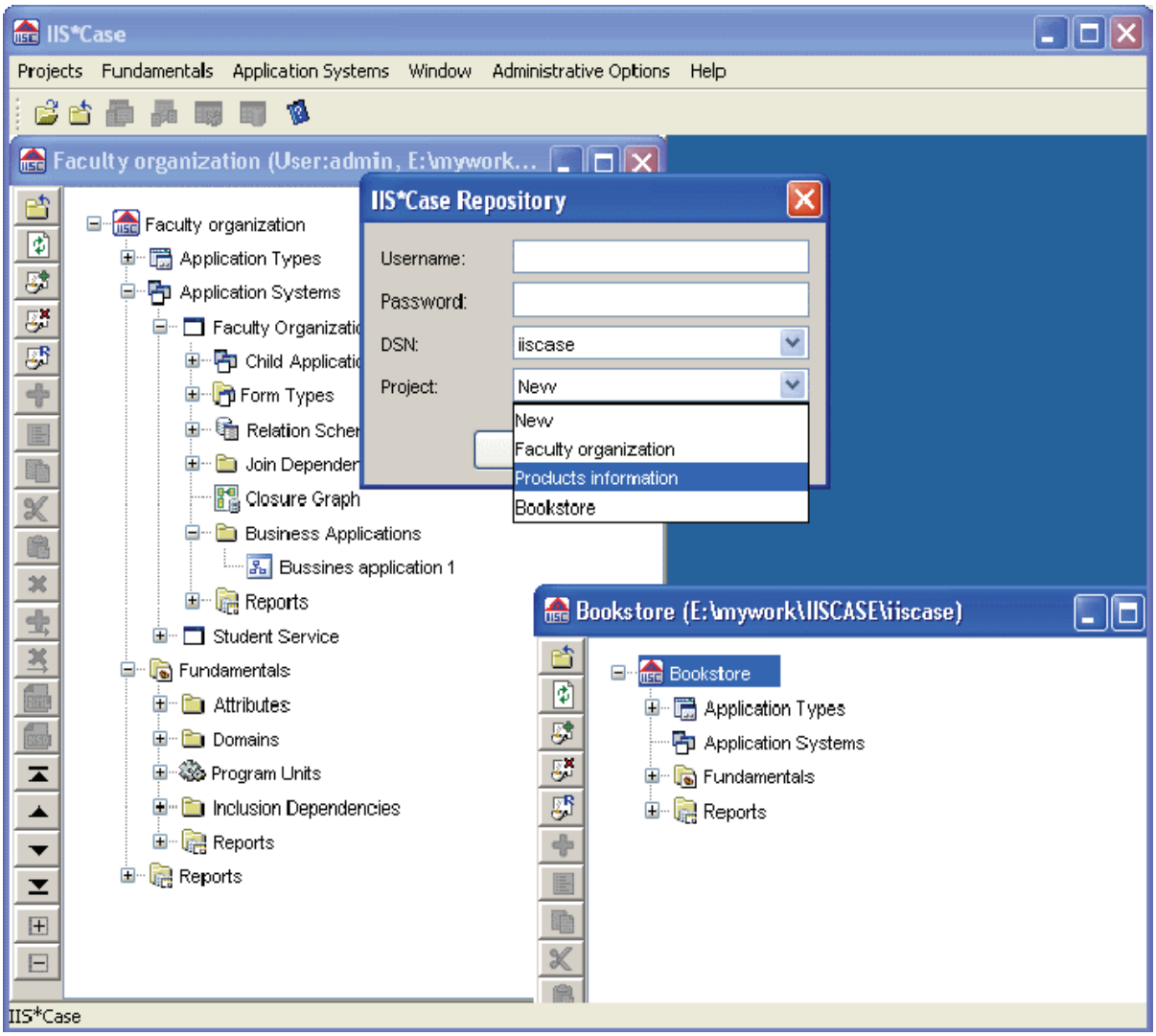
Aplikativni sistemi i tipovi aplikativnih sistema

Rad sa IIS*Case-om započinje kreiranjem projekta. Osnovnu jedinicu projekta budućeg informacionog sistema čini aplikativni sistem određenog tipa. Lista tipova aplikativnog sistema može se proširiti i mijenjati od strane projektanta. Konceptualna šema baze podataka projektuje se na nivou aplikativnog sistema.

Osnovni koncepti

Prvi korak u procesu modelovanja informacionog sistema je zadavanje osnovni koncepata projekta (slika 3.1.1.2), koji se definišu na nivou projekta, nezavisno od aplikativnog sistema. Njih čine:

- atributi ili obilježja,
- primitivni i korisnički definisani domeni,
- programske jedinice (paketi, funkcije i događaji) i
- zavisnosti sadržavanja.



Slika 3.1.1.1. *Korisnički interfejs IIS*Case-a*

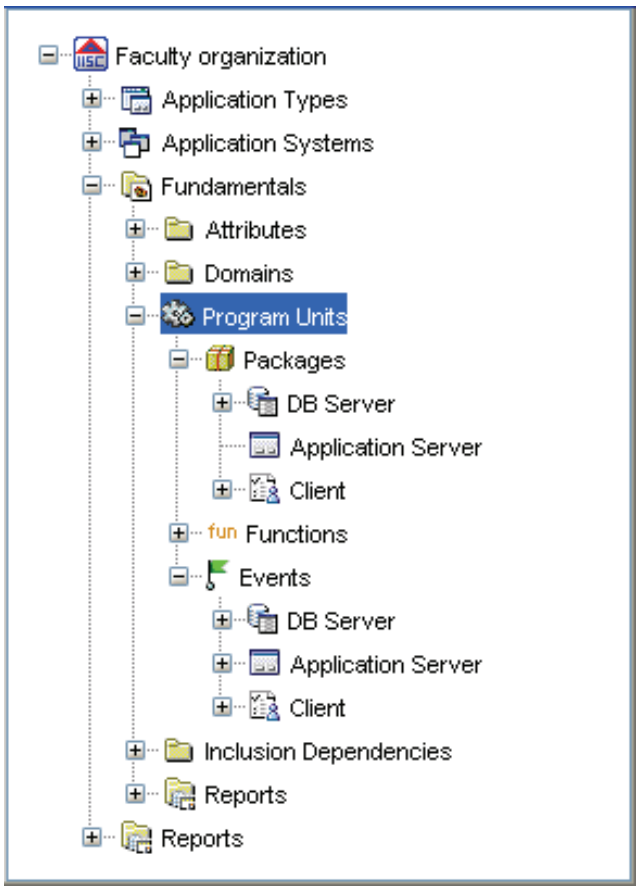
Primjena IIS*Case-a zasnovana je na pretpostavci o postojanju univerzalne šeme relacije, pa se i obilježja, tj. atributi, pri projektovanju, definišu nezavisno od šeme relacije kojoj će pripadati. Definiciji atributa prethodi definisanje potrebnih domena i funkcija. Domen se definiše na osnovu funkcija, ugrađenih primitivnih ili korisnički definisanih domena. Primitivni domeni su predstavljeni kao jedna lista koja nije fiksna i može se mijenjati i dopunjavati od strane projektanta. Jedan domen opet može naslijediti osobine drugog u okviru definisane hijerarhije. Detaljna specifikacija domena i atributa data je u [46].

Detaljna specifikacije koncepta programskih jedinica iz klase osnovnih koncepata neophodnih za izražavanje kompleksnih aplikativnih funkcionalnosti data je u [26, 37].

Koncept funkcije koristi se za specifikaciju kompleksnih funkcionalnosti. Funkcije u IIS*Case-u definisane su na nivou projekta i mogu biti referencirane iz različitih IIS*Case specifikacija.

Paket je kolekcija odabranih funkcija definisanih u IIS*Case repozitorijumu. U zavisnosti od odabranog sloja za razvoj paketa u multi-distribuiranoj softverskoj arhitekturi, na nivou platformski nezavisnog modela, dijelimo ih na pakete:

- servera baza podataka (*DB server*),
- aplikativnog servera (*Aplication Server*) i
- klijentskog sloja (*Client*).



Slika 3.1.1.2. *Stablo projekta*

Koncept događaja koristi se na nivou platformski nezavisnog modela, da predstavi bilo koji softverski događaj koji može pokrenuti neku akciju pod određenim uslovima. Događaji se dijele se na:

- događaje baze podataka (trigeri i izuzeci),
- događaje aplikativnog servera (događaj tastature, događaj miša i izuzeci) i
- događaje klijenta (događaj tastature, događaj miša i izuzeci).

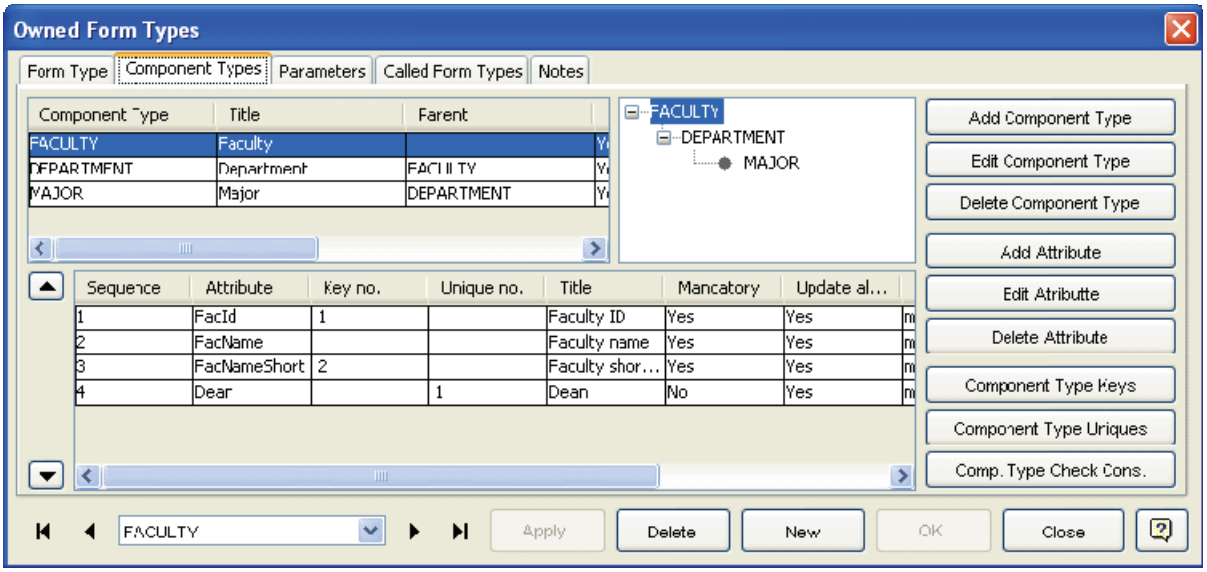
Svaki događaj bi trebalo da bude povezan sa softverskom specifikacijom na nivou platformski nezavisnog modela. Na primjer, triger baze podataka treba inicijalno da bude povezan sa nekim tipom komponente tipa forme, da bi, u kasnijim transformacijama, bila uspostavljena veza između trigera i generisane šeme relacije. Događaj tastature može biti povezan sa tipom forme, tipom komponente ili atributom tipa komponente.

Pomoću zavisnosti sadržavanja projektant može zadati ograničenja univerzalne šeme relacije tipa: skup vrijednosti niza atributa je podskup skupa vrijednosti drugog niza atributa. Ovakva ograničenja će uzrokovati pojavu odgovarajućih međurelacionih ograničenja implementacione šeme. Preimenovani atributi su specijalni slučajevi zavisnosti sadržavanja i

definisanje preimenovanja je analogno zadavanju odgovarajuće zavisnosti. Detaljna specifikacija zavisnosti sadržavanja data je u [46].

Tipovi forme

Tipovi formi predstavljaju osnovu na kojoj IIS*Case bazira algoritme za generisanje početnog skupa atributa i izdvajanje funkcionalnih zavisnosti kao ulaznih parametra postupka generisanja šeme baze podataka. Koncept tipa forme detaljno je specificiran u [9, 14, 22, 25, 29, 34, 35, 36]. Osim što se na osnovu njih može generisati šema baze, tipovi formi predstavljaju i polazne specifikacije budućih transakcionih programa, koji se kasnije tokom procesa projektovanja automatski generišu.



Slika 3.1.1.3. *Forma za specificiranje tipa forme*

Pristup zasnovan na konceptu tipa forme kao osnovnog strukturnog elementa pruža široke mogućnosti u projektovanju aplikativnih sistema različite primjene. Sličan pristup opisan je [31, 32], gdje je opisana UML specifikacija standarda korisničkog interfejsa. Ovaj standard definiše specifikaciju sistema čiji su strukturni elementi: generičke aplikacije, referati, aplikativni podsistemi, ekranske forme i izvještaji. Analogno, kod IIS*Case-a, svaki projekat sastoji se od jednog ili više aplikativnih sistema koji opet mogu imate svoje podsisteme, dok se svaki aplikativni sistem sastoji od jednog ili više tipova formi. Međutim, konceptualno gledano, tip forme razlikuje se od pojma ekranske forme koji je predstavljen u [31, 32]. Tip forme predstavlja noseći koncept projektovanja koji je po složenosti svoje specifikacije bogatiji, i na osnovu čije specifikacije se definiše ne samo korisnički interfejs, već se projektuje relaciona baza podataka sa svim međurelacionim ograničenjima.

Na tipovima formi zadaju se tipovi komponenti i njihova hijerarhijska struktura tipa stabla kao što je prikazano na slici 3.1.1.3. Jedan tip komponente je u vrhu stabla, a svi ostali su hijerarhijski pod njim. Svaki tip komponente može imati proizvoljan broj podređenih tipova komponenti.

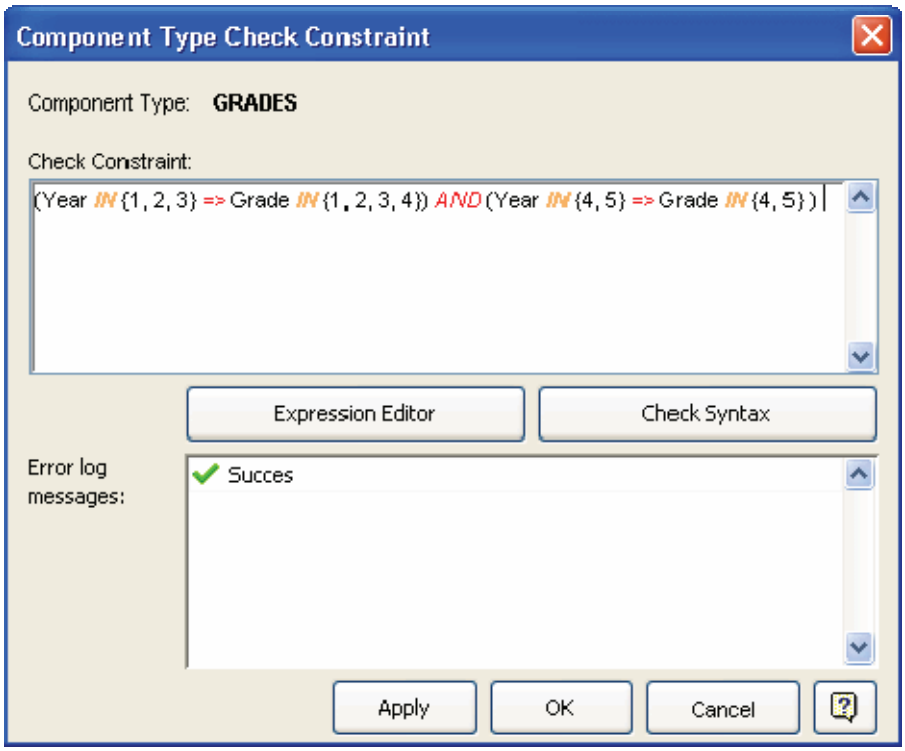
Zadavanje atributa na tipu komponente podrazumijeva izbor atributa iz skupa osnovnih koncepata i zadavanje dozvoljenih akcija nad datim atributom. Jedan atribut može biti, u okviru tipova komponenti, uključen na različitim tipovima formi, ali ne smije pripadati različitim tipovima komponenti istog tipa forme.

Nakon definisanja atributa zadaju se ključevi i ograničenja jedinstvene vrijednosti na tipu komponente.

Detaljna specifikacija tipova formi i njihovih strukturnih elemenata date su u [46, 47].

Izrazi ograničenja

IIS*Case omogućava definisanje izraza ograničenja na nivou konceptualne šeme baze podataka kao PIM modela [37]. Konceptualnu šemu čini skup tipova formi datog aplikativnog sistema. Izrazi ograničenja specificiraju se na nivou domena atributa, i tipova komponenti. Za ovu svrhu, razvijen je specijalizovani domenski orijentisan jezik. Glavna svrha izraza ograničenja domena i atributa je da ograniči dozvoljene vrijednosti pojedinačnih atributa. Suprotno ovome, izraz ograničenja tipa komponente koristi se da specificira logički uslov koji se odnosi na cijelu torku vrijednosti, za svaku moguću instancu tipa komponente. Na slici 3.1.1.4 prikazan je forma za specifikaciju izraza ograničenja tipa komponente.



Slika 3.1.1.4. *Forma za specificiranje izraza ograničenja*

3.1.2. Generisanje i konsolidacija šeme baze podataka

Hijerarhijska struktura u okviru koje jedan aplikativni sistem referencira druge aplikativne sisteme kao svoje podsisteme, implicira da u implementacionoj šemi aplikativnog sistema budu ugrađene i podšeme svih njegovih aplikativnih podsistema. To ne znači da je šema baze podataka aplikativnog sistema rezultat proste unije podšema njegovih podsistema. Podšeme će biti integrisane jednim složenim procesom, jer bi prosto uniranje moglo izazvati kolizije i redudantnost podataka.

Skup tipova formi jednog aplikativnog sistema, odnosno konceptualna šema baze podataka datog sistema, tokom faza generisanja i konsolidacije, biće transformisana u implementacionu relacionu šemu baze podataka.

Generisanje šeme baze podataka

Nakon zadavanja tipova formi, sljedeći koraci u projektovanju su: projektovanje šeme baze podataka i analiza usaglašenosti sa konsolidacijom podšema sa integrisanom šemom. Primjenom normalizacije na polazni skup ograničenja, definisan specifikacijom tipova formi i definisanjem međurelacionih ograničenja, generiše se implementaciona šema baze podataka.

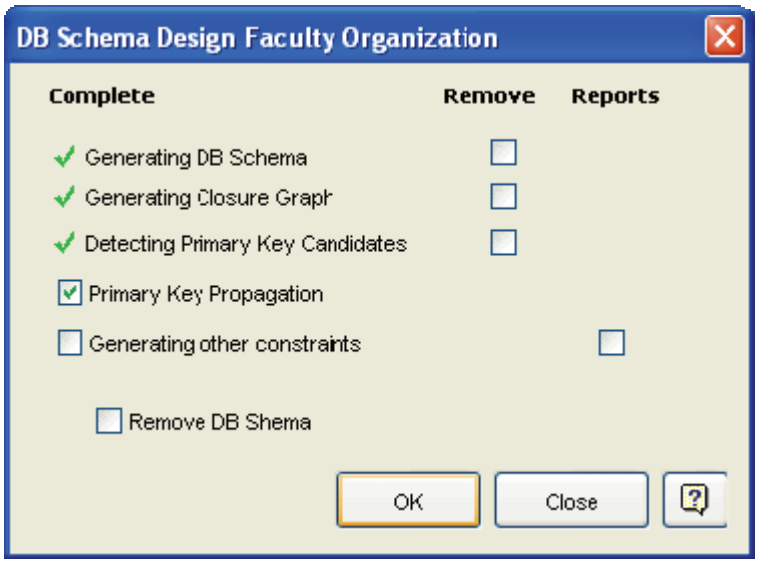
Putem tipova formi, moguće je identifikovati polazni skup ograničenja sljedećih tipova:

- funkcionalne zavisnosti,
- nefunkcionalni odnosi,
- ugrađene zavisnosti sadržavanja i
- specijalne funkcionalne zavisnosti sa ugrađenim ograničenjima jedinstvene vrijednosti.

Detaljna specifikacija ovih ograničenja data je u [46].

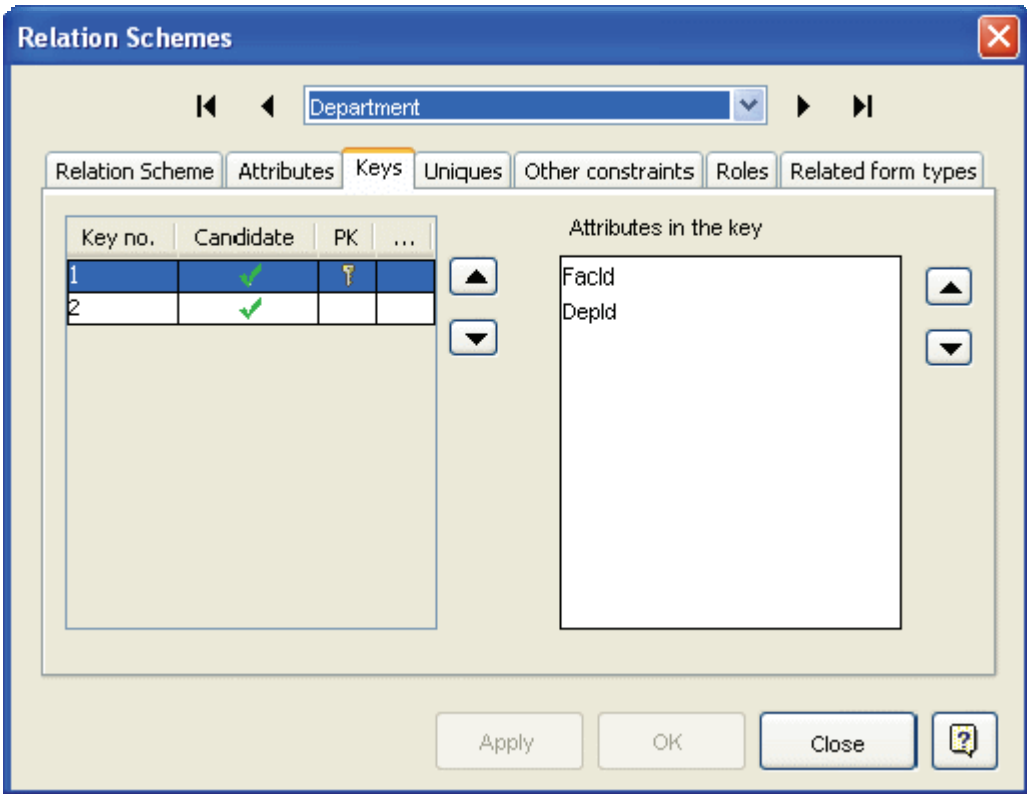
Kao što je prikazano na slici 3.1.2.1, proces generisanja šeme baze podataka aplikativnog sistema na osnovu polaznog skupa ograničenja obuhvata izvođenje sljedećih 5 koraka:

1. generisanje skupa šema relacija u trećoj normalnoj formi,
2. generisanje grafa zatvaranja,
3. određivanje kandidata za primarne ključeve,
4. prostiranje primarnih ključeva u skupu šema relacija generisane šeme i
5. generisanje skupa ograničenja date šeme baze podataka.



Slika 3.1.2.1. *Forma za generisanje šeme baze podataka*

Svi navedeni koraci se automatski izvode na osnovu zadate komande od strane projektanta kao što je prikazano na slici 3.1.2.1. Koraci se izvode na osnovu predviđenog redosleda i nije moguće preskočiti neki korak. Postupci i algoritmi implementirani u koracima procesa generisanja šeme baze podataka opisani su u [25, 35, 46, 48, 50].



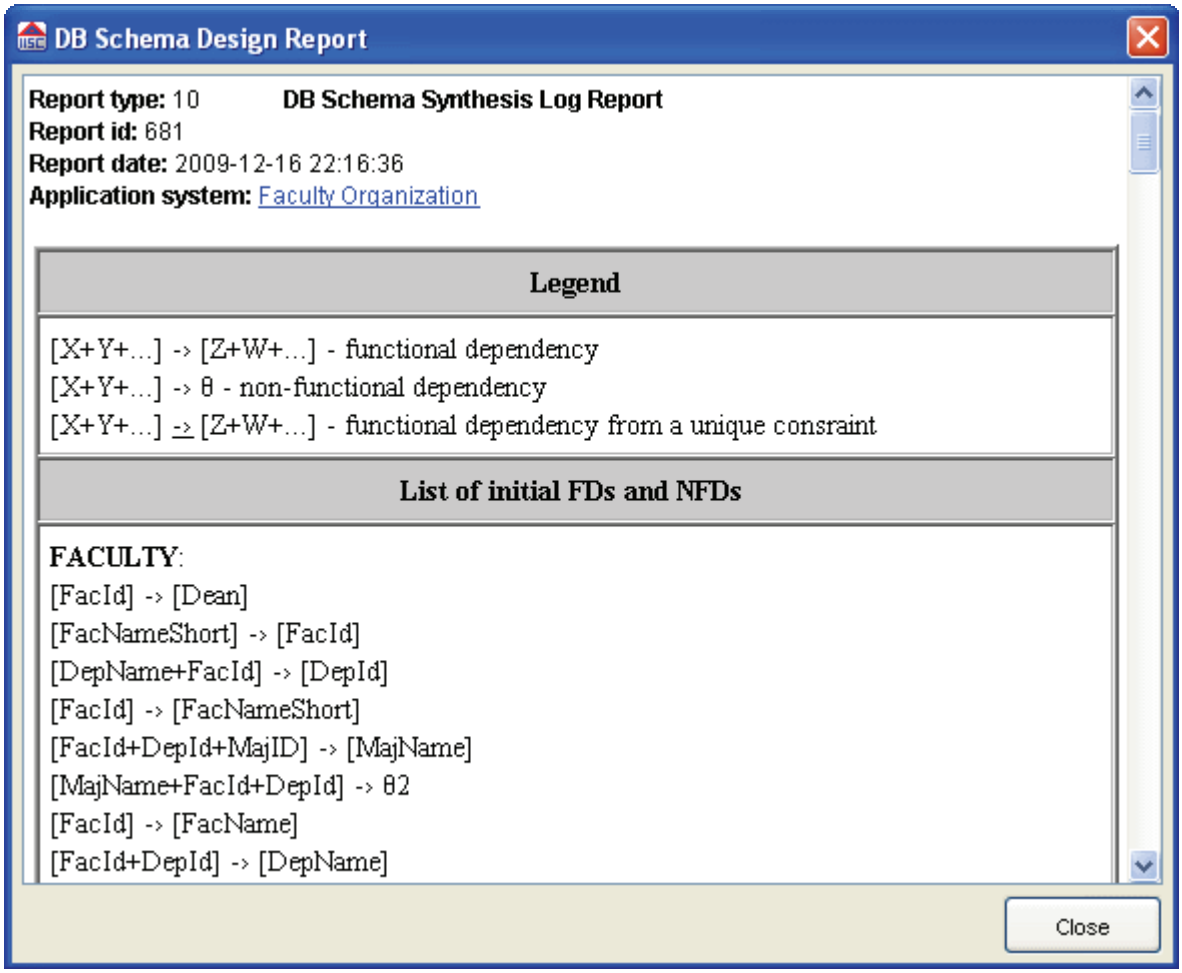
Slika 3.1.2.2. *Forma za specificiranje šema relacija*

Prvi korak prestavlja izvršenje algoritma sinteze koji kao ulazne parametre uzima polazni skup ograničenja, a kao rezultat daje skup šema relacija sa definisanim skupovima atributa, sintetizovanim ključevima i ograničenjima jedinstvene vrijednosti. Nakon svakog koraka procesa generisanja šeme baze podataka moguće je pristupiti trenutnoj specifikaciji pojedine šeme relacije kao što je prikazano na slici 3.1.2.2. Ovoj specifikaciji kao i drugim specifikacijama koje se tiču projektovanja šeme baze podataka se može pristupiti kroz podstablo koje odgovara aplikativnom sistemu u stablu projekta.

Drugi korak generiše graf zatvaranja kao grafičku reprezentaciju generisane šeme, koji je posebno važan za vizuelizaciju međurelacionih ograničenja.

U trećem koraku iz skupa ekvivalentnih ključeva, biraju se oni koji mogu biti proglašeni za primarni ključ.

U četvrtom koraku, u slučaju da više od jednog ekvivalentnog ključa šeme relacije bude proglašeno kandidatom za primarni ključ, na projektantu je da odabere koji će kandidat biti propagiran.



Slika 3.1.2.3. *Izvještaj o rezultatima i međurezultatima procesa sinteze*

Petim korakom generišu se sva međurelaciona ograničenja iz skupova:

1. ograničenja referencijalnih integriteta,
2. ograničenja referencijalnih integriteta, zasnovanih na netrivijalnim zavisnostima sadržavanja, i
3. ograničenja inverznih referencijalnih integriteta, osnovnih i onih zasnovanih na netrivijalnim zavisnostima sadržavanja.

Ukoliko projektant tako izabere, IIS*Case automatski generiše i odgovarajuće izvještaje nakon prvog i petog koraka procesa generisanja. Na slici 3.1.2.3 je prikazan izvještaj koji se generiše tokom sinteze (prvog koraka), a koji uključuje informacije o svim transformacijama koje su urađene nad polaznim skupom ograničenja.

Konsolidacija (usaglašavanje) šeme baze podataka

Svaka podšema koja odgovara podsistemu apliaktivnog sistema mora biti konzistentna sa šemom baze podataka cijelog sistema, tj. mora da zadovoljava usaglašenost svih relacionih i međurelacionih ograničenja. Ovo znači da ukoliko se podšeme integrišu u neku šemu baze podataka, neophodno je nakon integracije provjeriti konzistentnost na višem nivou. Da bi se moglo smatrati da su u implementacionoj šemi pravilno integrisane sve njene podšeme, mora biti zadovoljen uslov usaglašenosti koncepata podšeme sa konceptima potencijalne

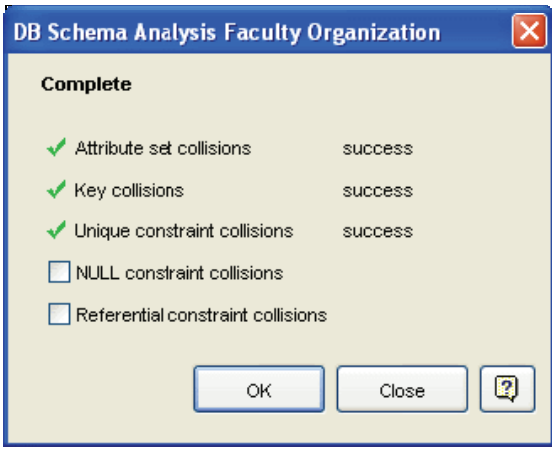
implementacione šeme. To znači da mora važiti sljedeće:

- za svaku šemu relacije podšeme, postoji šema relacije integrisane šeme, takva da je skup obilježja šeme relacije podšeme podskup skupa obilježja šeme relacije implementacione šeme i primarni ključ šeme relacije podšeme je jednak primarnom ključu korespondentne šeme relacije integrisane šeme;
- sva relevantna ograničenja iz skupa ograničenja integrisane šeme su ugrađena u skup ograničenja podšeme.

Nakon bilo kog koraka generisanja šeme buduće baze podataka aplikativnog sistema koji referencira aplikativni podsistem ili tip forme iz drugog sistema, projektant je u mogućnosti da provjeri usaglašenost integrisane šeme sa podšemama da bi se utvrdilo da li su uslovi usaglašenosti zadovoljeni, tj. da li ona može da predstavlja implementacionu šemu baze podataka. Algoritam za konsolidaciju kreiran je na osnovu postupaka i algoritama predloženih u [48]. U [45, 46] detaljno je opisno kako je dati algoritam implementiran u IIS*Case-u.

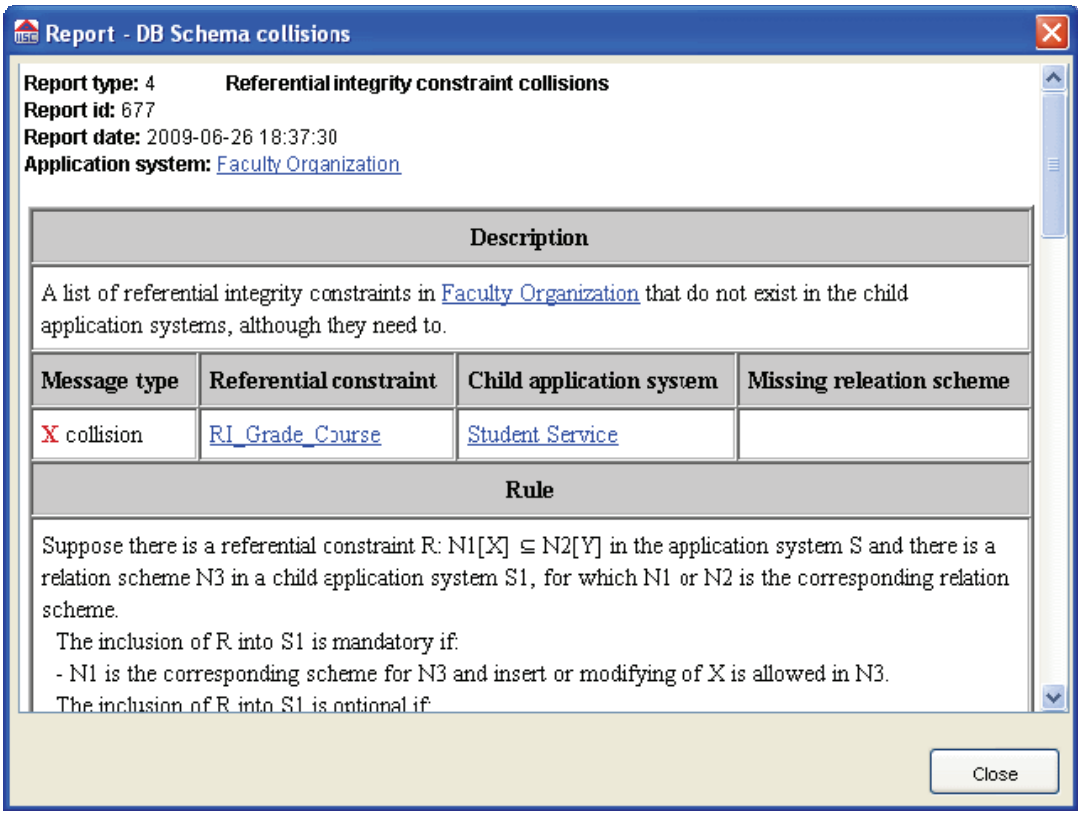
Proces konsolidacije šeme baze podataka sa podšemama podijeljen je na segmente koji usaglašavaju pojedinačne grupe ograničenja istog tipa. Proces se sastoji od sljedećih koraka:

1. usaglašenost skupova atributa podšema i potencijalne implementacione šeme baze podataka,
2. usaglašenost skupova ograničenja jedinstvene vrijednosti atributa,
3. usaglašenost skupova ograničenja ključa,
4. usaglašenost skupova ograničenja NULL vrijednosti atributa i
5. usaglašenost skupova ograničenja referencijalnih integriteta.



Slika 3.1.2.4. *Forma za analizu usaglašenosti šeme baze podataka*

Redosled izvršavanja ovih koraka je bitan, jer uspješnost provjere jedne grupe ograničenja može usloviti mogućnost provjere druge grupe. Zato korake nije moguće preskakati ili izvršavati po proizvoljnom redosledu. Na slici 3.1.2.4 prikazana je ekranska forma preko koje projektant zadaje naredbe za izvršavanje pojedine korake algoritma konsolidacije.



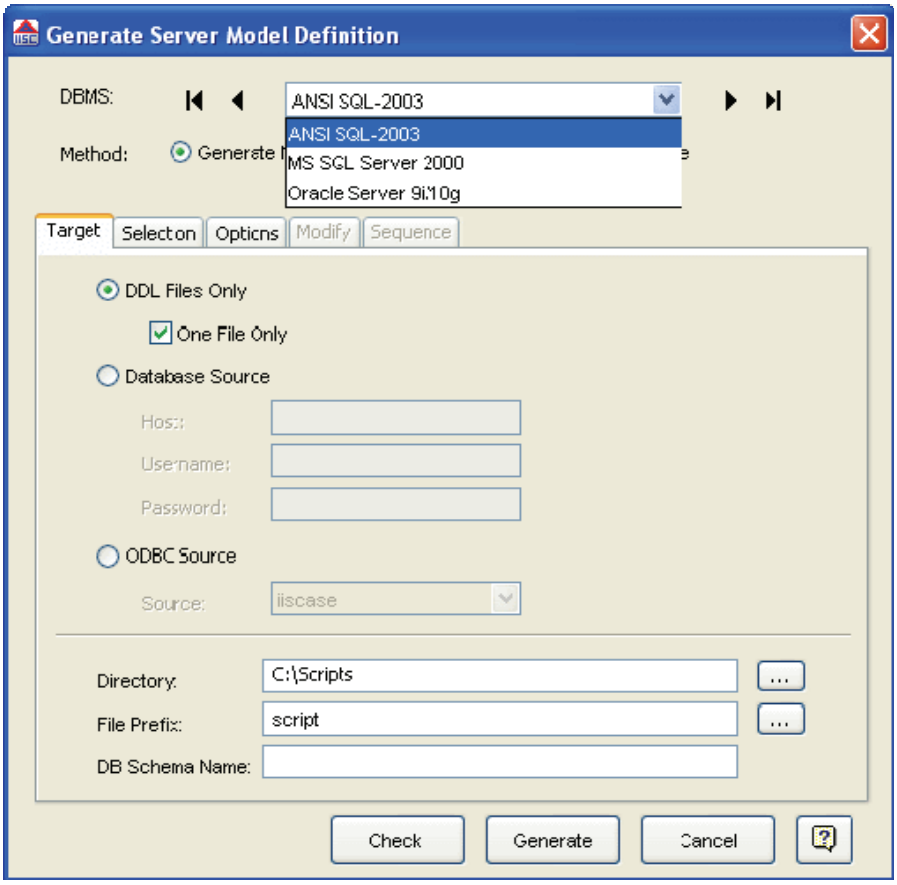
Slika 3.1.2.5. *Izvještaj o kolizijama ograničenja referencijalnog integriteta*

I ovaj proces je dobro dokumentovan kroz generisanje izvještaja u kojima se detaljno opisuju neusaglašenosti podšema, koje se odnose na skupa ograničenja koji se u datom koraku usaglašava. Svi generisani izvještaji čuvaju se, tako da projektant može da pristupi istoriji promjena. Direktan pristup izvještaju moguć je kroz podstablo stabla projekta koje odgovara datom aplikativnom sistemu. Pretraga po tipu i sadržaju izvještaja olakšava njihovo pregledanje, u slučaju velikog broja izgenerisanih izvještaja. Na slici 3.1.2.5 dat je jedan primjer izvještaja neusaglašenosti skupova ograničenja referencijalnih integriteta.

Proces projektovanja šeme baze podataka je iterativan. Svaka iteracija podrazumijeva kreiranje novog ili promjenu postojećeg skupa polaznih ograničenja kroz specifikaciju tipova formi, generisanje šeme baze i primjenu algoritma konsolidacije. Ovaj proces ponavlja se u onoliko iteracija koliko je potrebno dok se ne dobije usaglašena šema baza podataka.

SQL generator

IIS*Case podržava automatsko generisanja SQL opisa šeme baze podataka. U alat je ugrađen SQL generator (slika 3.1.2.6) putem kojeg je moguće implementirati internu šemu baze podataka u okviru izabranog sistema za upravljanje bazama podataka podržanog od strane alata. Na taj način zaokružuje se proces projektovanja šeme baze podataka. Principi rada SQL generatora opisani su u [2].



Slika 3.1.2.6. *SQL generator*

3.1.3. Automatsko generisanje transakcionih programa

U IIS*Case-u implementiran je generator transakcionih programa koji na osnovu zadatih specifikacija tipova formi i podsistema datog aplikativnog sistema, kao i izabranog šablona korisničkog interfejsa, automatski generiše prototip aplikacije. Navedena funkcionalnost predstavlja glavni rezultat istraživanja sprovedenog u ovoj doktorskoj disertaciji. Nadalje su ukratko opisani osnovni detalji funkcionalnosti generatora. Detaljni prikaz i način implementacije funkcionalnosti generisanja aplikacija informacionih sistema navedeni su u okviru poglavlja 5.

U implementaciji generatora primjenjen je pristup zasnovan na *User Interface Markup Language* (UIML) [1]. UIML je jezik baziran na XML-u koji omogućava generisanje korisničkog interfejsa za različite sistemske platforme i programska okruženja.

U [3] detaljno je diskutovan ovaj pristup i naglašeno je da se pomoću njega mogu generisati softverske specifikacije koje je moguće izvršavati na različitim platformama, uz manje izmjene, ili uopšte bez dodatnih izmjena na kodu. Generisanje izvršnih softverskih specifikacija zasnovano je na upotrebi opšteg rečnika kojeg je moguće interpretirati na različitim implementacionim platformama. Ove specifikacije predstavljene su UIML dokumentima koji se *rendering* metodom konvertuju u formu koja može biti predstavljena krajnjem korisniku i sa kojom korisnik može da komunicira. *Rendering* je moguće ostvariti na

dva načina:

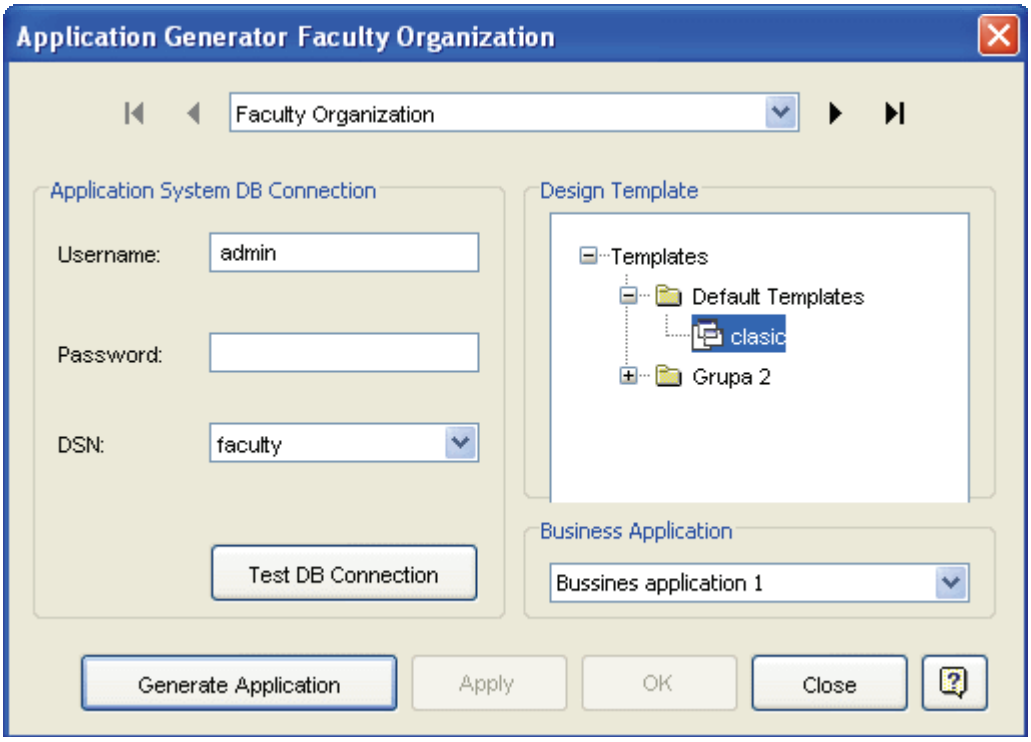
- kompajliranjem UIML-a u drugi programski jezik (npr. WML ili VoiceXML), ili
- interpretiranjem UIML-a tj. kreiranjem programa koji čita UIML dokument i poziva API-je za prikaz korisničkog interfejsa i interakciju sa krajnjim korisnikom.

U [3] su autori iznijeli stečena iskustva u radu sa UIML-om, i došli do zaključka da nije dovoljno generisanje izvršnih softverskih specifikacija zasnovano samo na opštem rečniku, pogotovo kada te specifikacije treba da se izvršavaju na tako različitim platformama kao što su VoiceXML, Palm PDA ili desktop računar. Stoga je potrebno specifikaciju korisničkog interfejsa razložiti na specifikaciju aplikacije na višem nivou apstrakcije (specifikacija zasnovana na opštem rečniku) i specifikaciju koja definiše zavisnost od razvojne platforme, koju je opet moguće, korišćenjem transformacija, automatski prilagoditi različitim platformama.

U [17, 18] predstavljen je jedan pristup generisanju prototipova aplikacija zasnovanom na XML-u. Na samom početku procesa projektovanja informacionog sistema podrazumijeva se upotreba CASE alata kojim bi se specificirali svi korisnički zahtjevi i generisali u okviru XML dokumenta. Metod predlaže generisanje XML specifikacija koje ne zavise od sistemske ili programske platforme, korišćenjem *XML Document Type Definitions* (XML DTD) dokumenata za definisanje pravila projektovanja. Primjenom *Extensible Stylesheet Language Transformations* (XSLT) metoda, ove XML specifikacije transformišu se u izvršne prototipove informacionih sistema. Prototipovi, specificirani XML dokumentima, automatski se transformišu u UIML specifikacije. Za razliku od XML specifikacija, UIML specifikacije zavise od platforme na kojoj će se prototip izvršavati. U [17, 18] opisuje se kako se ove UIML specifikacije mogu interpretirati korišćenjem *Java Render*-a. Kao izvršno i programsko okruženje izabran je *Harmonia Incorporation® Java Render*. *Java Render* implementira pravila koja transformišu UIML komponente u *Java* softverske komponente. Međutim, generisane komponente interpretiraju samo vizuelne karakteristike korisničkog interfejsa, ne i njegovo ponašanje i mogućnost interakcije (npr. komunikacija sa bazom podataka). Zato je ovaj nedostatak *Java Render*-a nadomješten kreiranjem *Java* klasa koje implementiraju ugrađeno ponašanje komponenti informacionog sistema.

Na slici 3.1.3.1 prikazana je slika ekranske forme putem koje projektant zadaje parametre koji su neophodni da bi otpočeo proces generisanja. Prirodno, ovaj postupak se izvršava nakon završenog generisanja i konsolidacije šeme baze podataka, kao i njene implementacije u okviru konkretnog sistema za upravljanje bazama podataka. Neophodno je da projektant zada sljedeće:

- pristupne parametre za komunikaciju sa bazom podataka,
- izabrani šablon korisničkog interfejsa (detaljnija specifikacija šablona korisničkog interfejsa data je u poglavljima 3.2 i 4) i
- izabranu poslovnu aplikaciju.



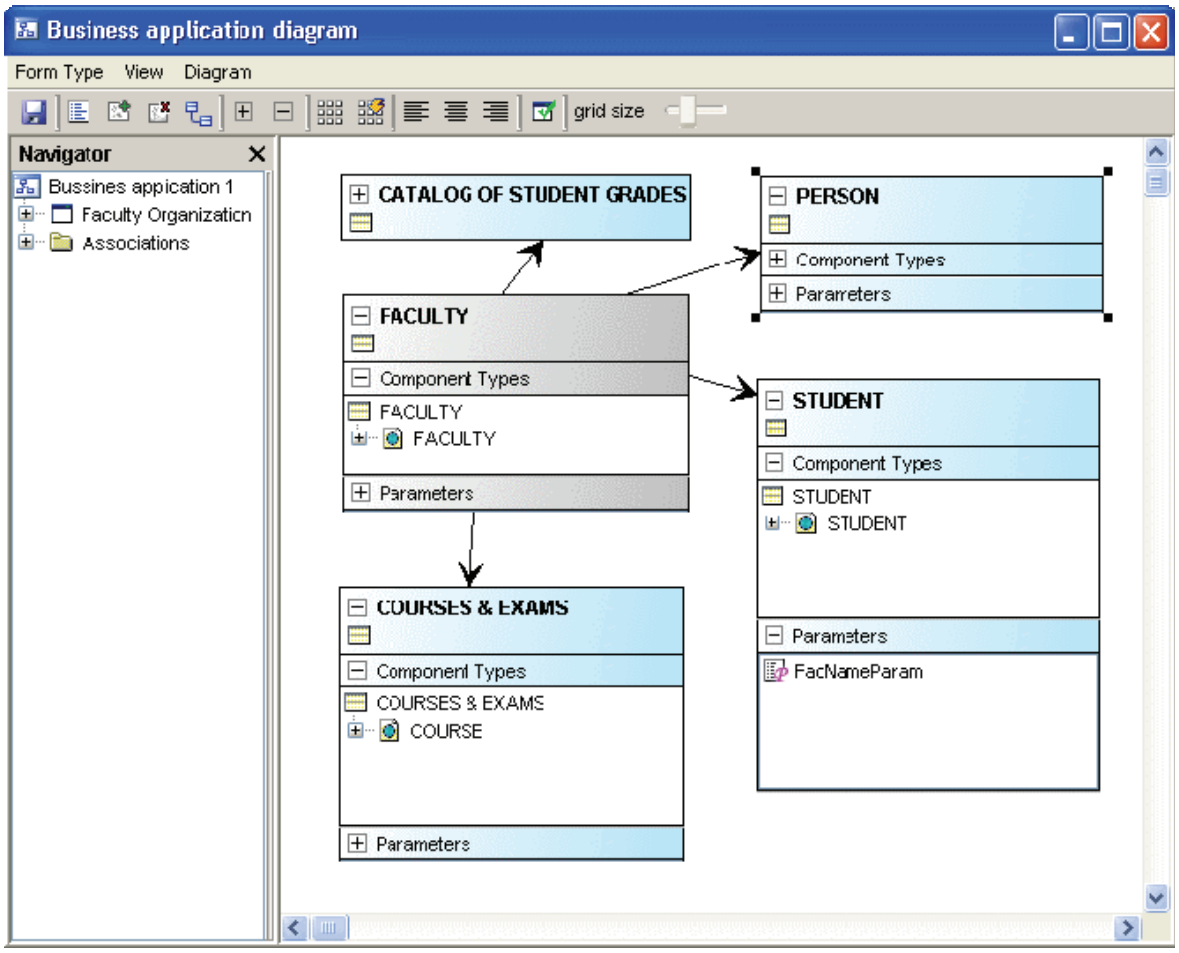
Slika 3.1.3.1. *Generator aplikacija*

Poslovne aplikacije

Koncept poslovne aplikacije uveden je i detaljno opisan u [47]. Specificiranjem poslovnih aplikacija, projektant formalno opisuje funkcionalnosti informacionog sistema, koje se odnose na uzajamne pozive generisanih ekranskih formi koje odgovaraju različitim tipovima formi. Ovaj koncept definiše se na nivou aplikativnog sistema. Pored skupa tipova formi koji učestvuje u poslovnoj aplikaciji, specificira se i kako generisane ekranske forme međusobno komuniciraju. Zato se, na nivou poslovne aplikacije, definiše koncept koji se naziva pozivajuća struktura. Jedna pozivajuća struktura predstavlja uređeni par nad skupom tipova formi koji učestvuju u poslovnoj aplikaciji. Osim ove informacije, konceptu pozivajuće strukture pridružene su sljedeće osobine:

- proslijeđene vrijednosti,
- način poziva,
- metod poziva i
- UI pozicioniranje.

Na slici 3.1.3.2. dat je primjer jedne poslovne aplikacije prikazan u okviru modula za grafičko modelovanje poslovnih aplikacija, koji se naziva *Business Aplication Designer*.

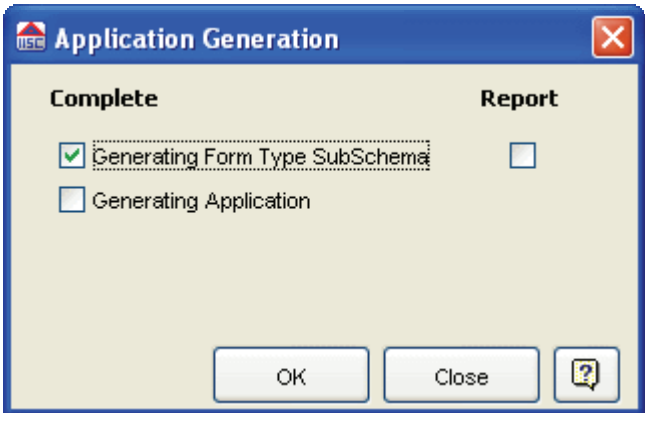


Slika 3.1.3.2. *Business Aplication Designer*

Generisanje transakcionog programa

Proces generisanja transakcionog programa sastoji se iz dva koraka (kao što je prikazano na slici 3.1.3.3):

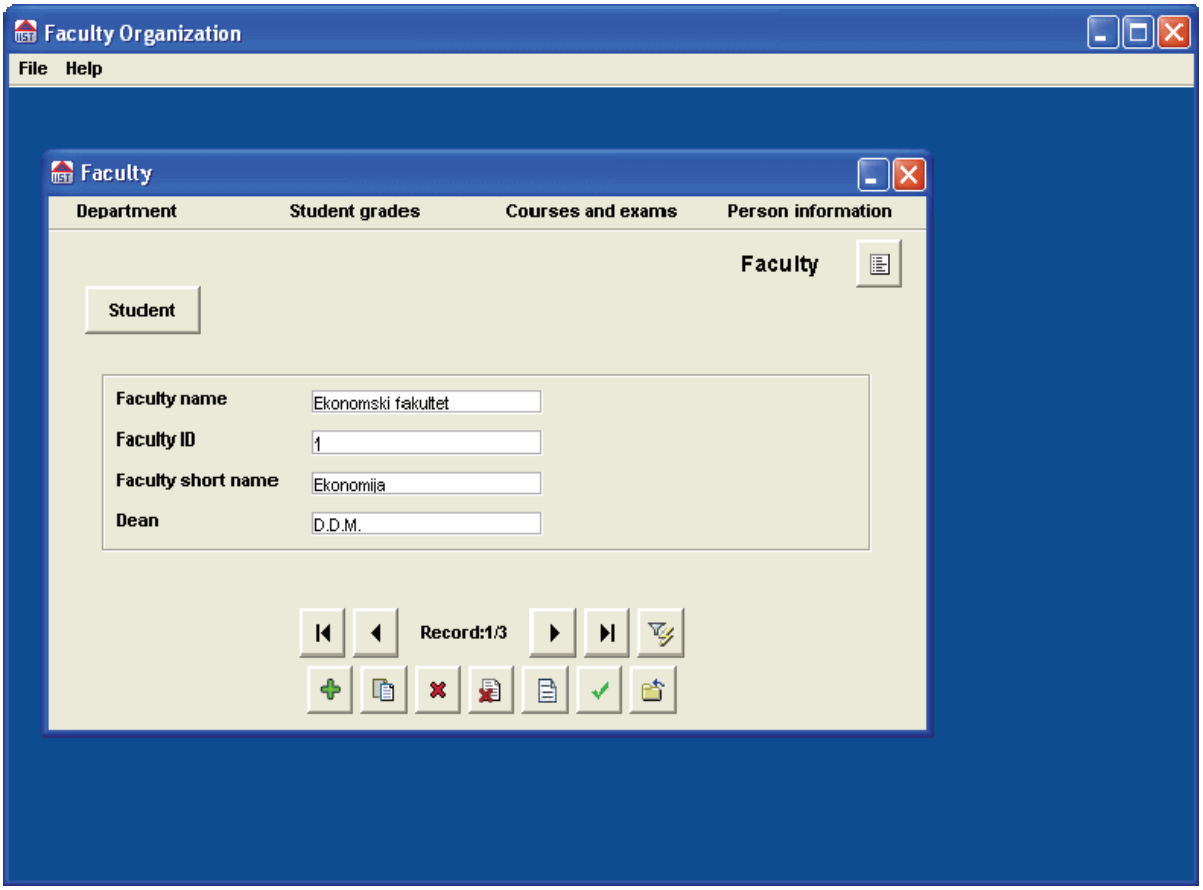
- generisanje podšema tipova formi, i
- generisanje aplikacije.



Slika 3.1.3.3. *Generisanje prototipa aplikacije*

Na slici 3.1.3.4 prikazan je primjer automatski generisane aplikacije.

Detaljna specifikacija svih postupaka generisanja i načini na koji su implementirani u generatoru transakcionih programa predstavljaju predmet istraživanja ove doktorske teze, i opisani su detaljno u poglavlju 5 ovog rada.

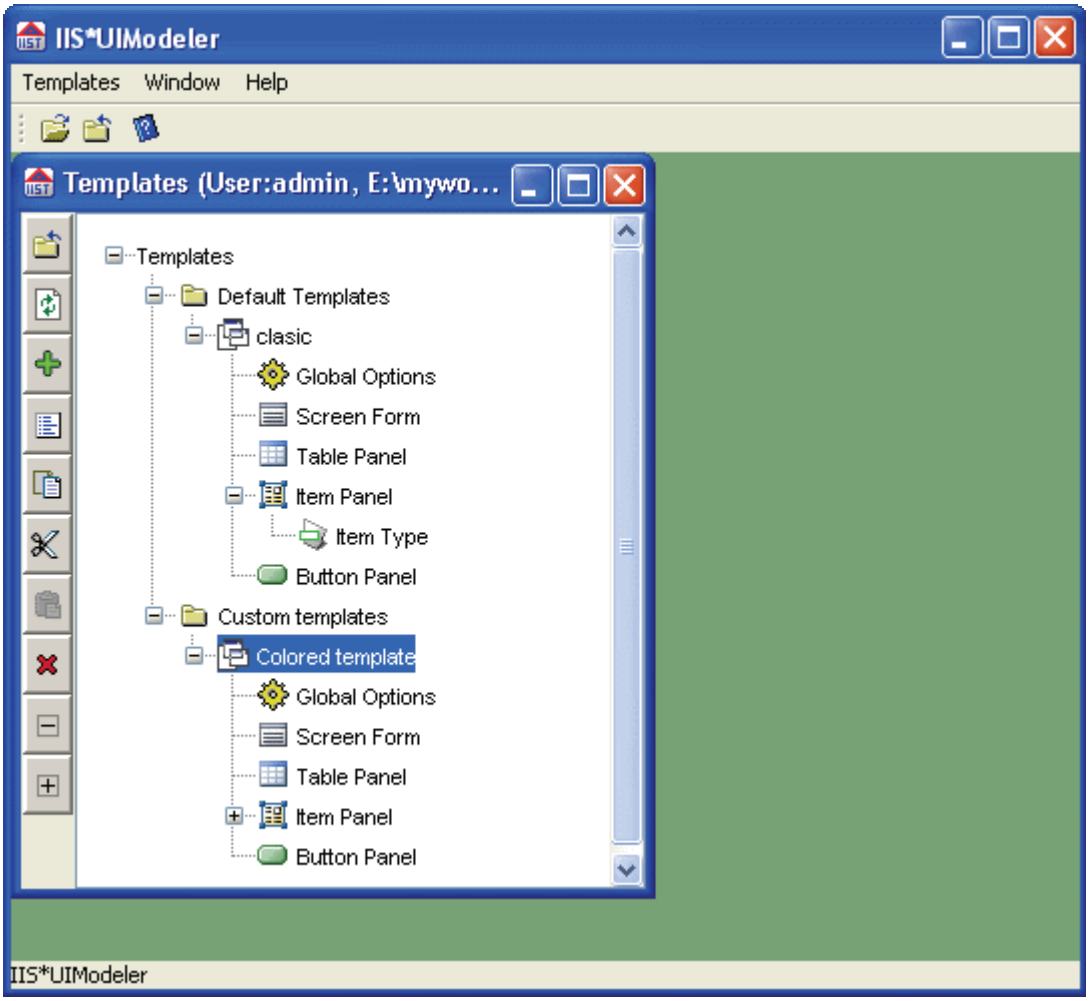


Slika 3.1.3.4. *Primjer prototipa aplikacije*

3.2. Šabloni korisničkog interfejsa informacionog sistema

IIS*UIModeler je alat koji omogućava kreiranje šablona korisničkog interfejsa. Šablon predstavlja specifikaciju opštih karakteristika korisničkog interfejsa. Izabrani šablon je neophodan ulazni parametar generatora transakcionih programa u IIS*Case-u.

Na slici 3.2.1. prikazan je osnovni prozor korisničkog interfejsa IIS*UIModeler-a. Slično kao kod IIS*Case-a, kroz IIS*UIModeler moguće je ostvariti više konekcija na različite repozitorijume i raditi paralelno sa njima. Projektant je u mogućnosti da kreira neograničen broj šablona ili koristi neki od predefinisanih. Radi lakšeg korišćenja, šabloni su organizovani u grupe i pristupa im se preko stabla šablona. Projektant može kreirati, preimenovati i brisati grupe. Jednom kreirane šablone može mijenjati, brisati ili kopirati.

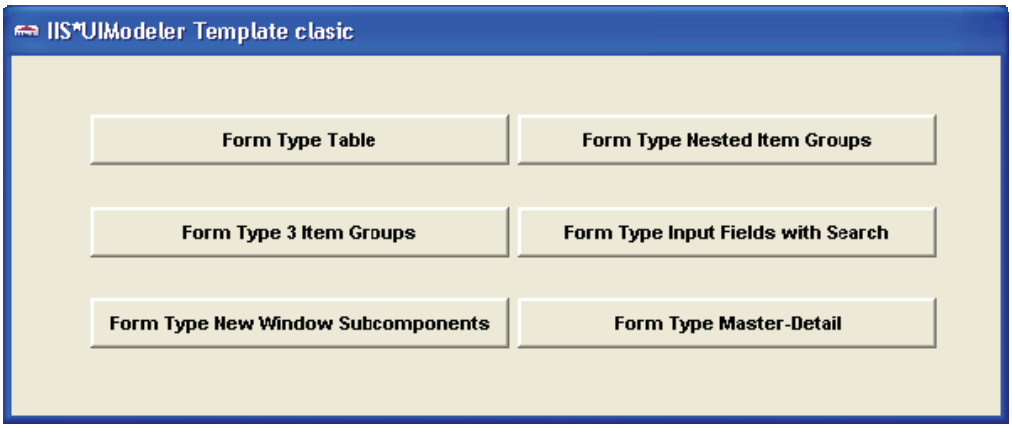


Slika 3.2.1. *Korisnički interfejs IIS*UIModeler-a*

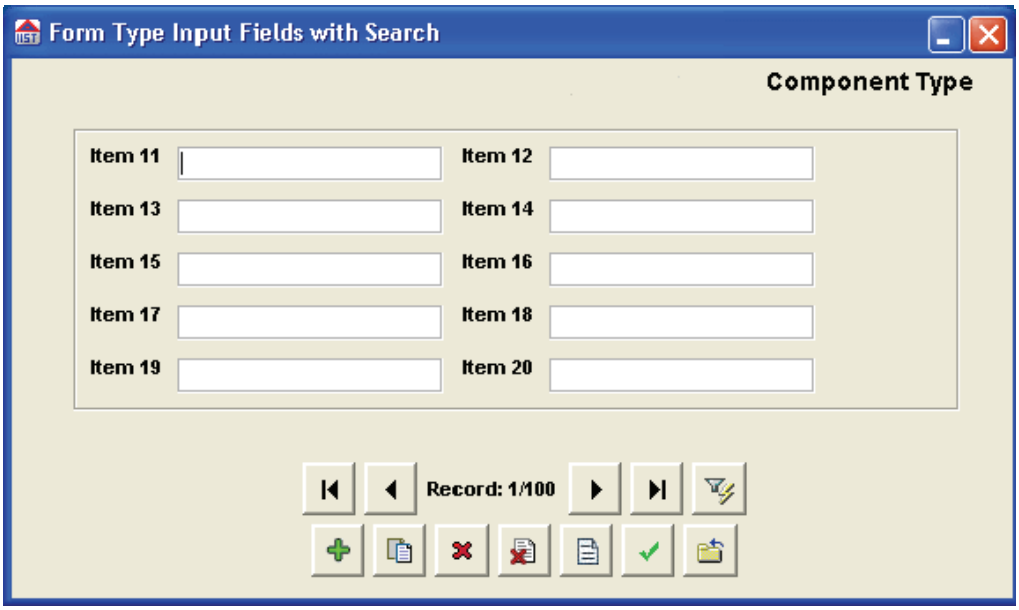
Najvažniji elementi grafičkog prikaza korisničkog interfejsa koje je moguće definisati putem šablona su:

- Inicijalni položaj, način prikaza i veličine glavnog prozora aplikacije i prozora ekranskih formi, kao i način njihovih međusobnih poziva.
- Specifikacija fontova koji se koriste za naslove i menije ekranskih formi, za tabele, panele, dugmad i polja za prikaz i ažuriranje podataka.
- Specifikacija boja slova i boja pozadina ekranskih formi, panela, tabela, dugmadi i ivica.
- Specifikacija prikaza ivica polja za prikaz i ažuriranje podataka, dugmadi, panela i tabela.
- Tip i način prikaza grupa polja. Mogući tipovi su:
    - tabelarni prikaz,
    - normalni prikaz panela, i
    - prikaz panela sa jezičcima.

- Tip i način prikaza polja za prikaz i ažuriranje podataka. Moguće je specificirati atribute sljedećih programskih kontrola:
  - tekstualno polje (*Text box*),
  - radio grupa (*Radio button*),
  - polje za čekiranje (*Check box*),
  - padajuća lista (*Combo box*), i
  - lista izbora (*List*).
- Tip i način prikaza dugmadi. Mogući tipovi su:
  - tekstualni prikaz, i
  - grafički prikaz.

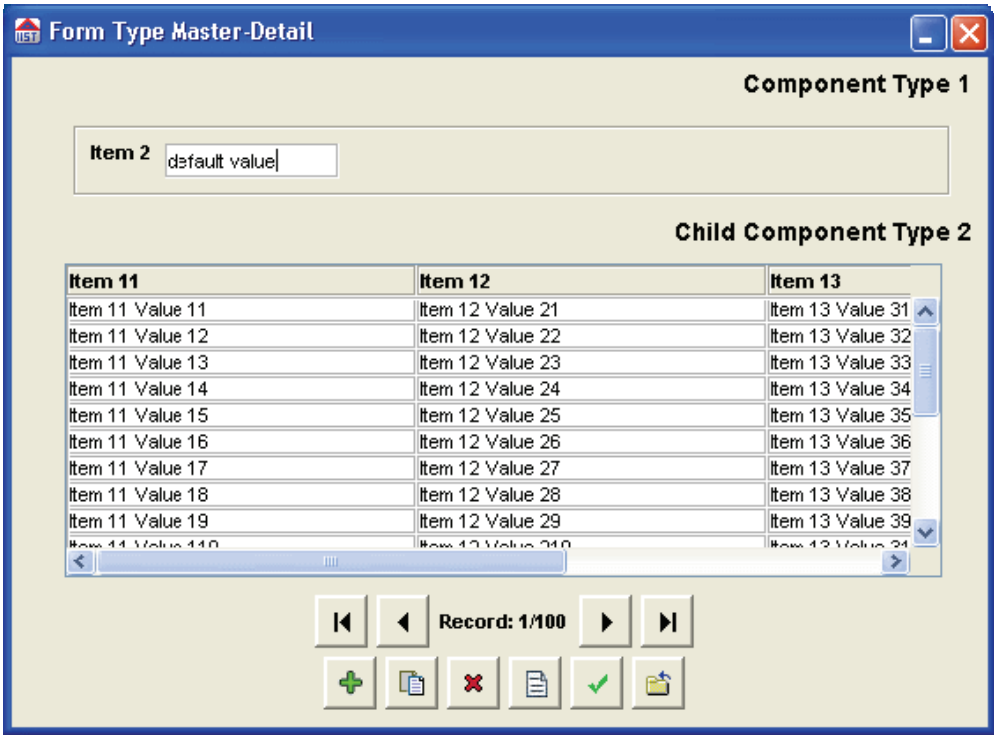


Slika 3.2.2. *Prototip korisničkog interfejsa - početna forma*



Slika 3.2.3. *Prototip korisničkog interfejsa – forma sa prikazom polja u okviru panela*

Prije nego što primjeni kreirani šablon prilikom generisanja izvršne softverske specifikacije nekog informacionog sistema, projektant ima mogućnost da se kroz IIS*UIModeler, na primjeru nekoliko različito struktuiranih ekranskih formi, uvjeri da li kreirani šablon zadovoljava potrebe krajnjeg korisnika (slika 3.2.2). Na slikama 3.2.3. i 3.2.4 dati su primjeri ekranskih formi generisani korišćenjem predefinisanog šablona.



Slika 3.2.4. *Prototip korisničkog interfejsa – forma sa tabelarnim prikazom polja*

Detaljna specifikacija šablona korisničkog interfejsa, način implementacije i generisanja prikaza primjera ekranskih formi kroz IIS*UIModeler predstavljaju predmet istraživanja ove doktorske teze i prikazani su u poglavlju 4.

4. Kreiranje šablona korisničkog interfejsa

Osim podrške projektovanja specifikacija koje opisuju funkcionalnosti konkretnih aplikacija i strukturu šeme baze podataka informacionih sistema, alati okruženja IIS*Studio pružaju mogućnost dizajniranja vizuelnih aspekata korisničkog interfejsa specificiranjem šablona, kao i generisanja prototipova aplikacija koje primjenjuju date šablone. U ovom poglavlju će biti više riječi o atributima šablona korisničkog interfejsa, kao i načinu generisanja prototipova korisničkog interfejsa primjenom UIML jezika.

Nakon kreiranja šablona korisničkog interfejsa, alat IIS*UIModeler pruža mogućnost generisanja prototipova korisničkog interfejsa koji u sebi integriše specifikacije izabranih šablona. Generator prototipova korisničkog interfejsa omogućava projektantu da u svakom trenutku tokom procesa modelovanja šablona ima vizuelnu predstavu kako će izgledati korisnički interfejs u odnosu na trenutno stanje kreirane specifikacije šablona. Na sličan način na koji se generišu prototipovi korisničkog interfejsa, biće generisani i prototipovi aplikacija upotrebom generatora aplikacija alata IIS*Case, o čemu će više biti riječi u okviru poglavlja 5.

4.1. Atributi šablona korisničkog interfejsa

Šablon specificira opšte karakteristike korisničkog interfejsa aplikacija informacionih sistema. Strukturu šablona korisničkog interfejsa čine atributi (karakteristike), koji se mogu zadati kroz alat IIS*UIModeler. Atributi šablona organizovani su u 6 grupa:

- Globalni atributi (*Global options*),
- Atributi ekranskih formi (*Screen form*),
- Atributi tabela (*Table panel*),
- Atributi panela (*Item panel*),
- Atributi polja za prikaz i ažuriranje podataka (*Item type*) i
- Atributi dugmadi (*Button panel*).

Globalni atributi

Svaki šablon ima svoje ime po kojem se razlikuje od drugih šablona u datoj grupi. Ime šablona je dio specifikacije njegovih globalnih atributa.

Prototip aplikacije informacionog sistema sastoji se od glavne ekranske forme (glavni prozor aplikacije) u okviru koje je moguće pristupiti ekranskim formama koje odgovaraju pojedinim tipovima komponenti na tipovima formi projektovanog informacionog sistema (slika 3.1.3.4). Pošto glavna ekranska forma nije dio specifikacije informacionog sistema, specifikacija njenih atributa izdvojena je u okviru grupe globalnih atributa šablona korisničkog interfejsa. Tako globalni atributi definišu osobine glavnog prozora aplikacije kao što su: veličina, pozicija, boje pozadine i desktopa, itd.

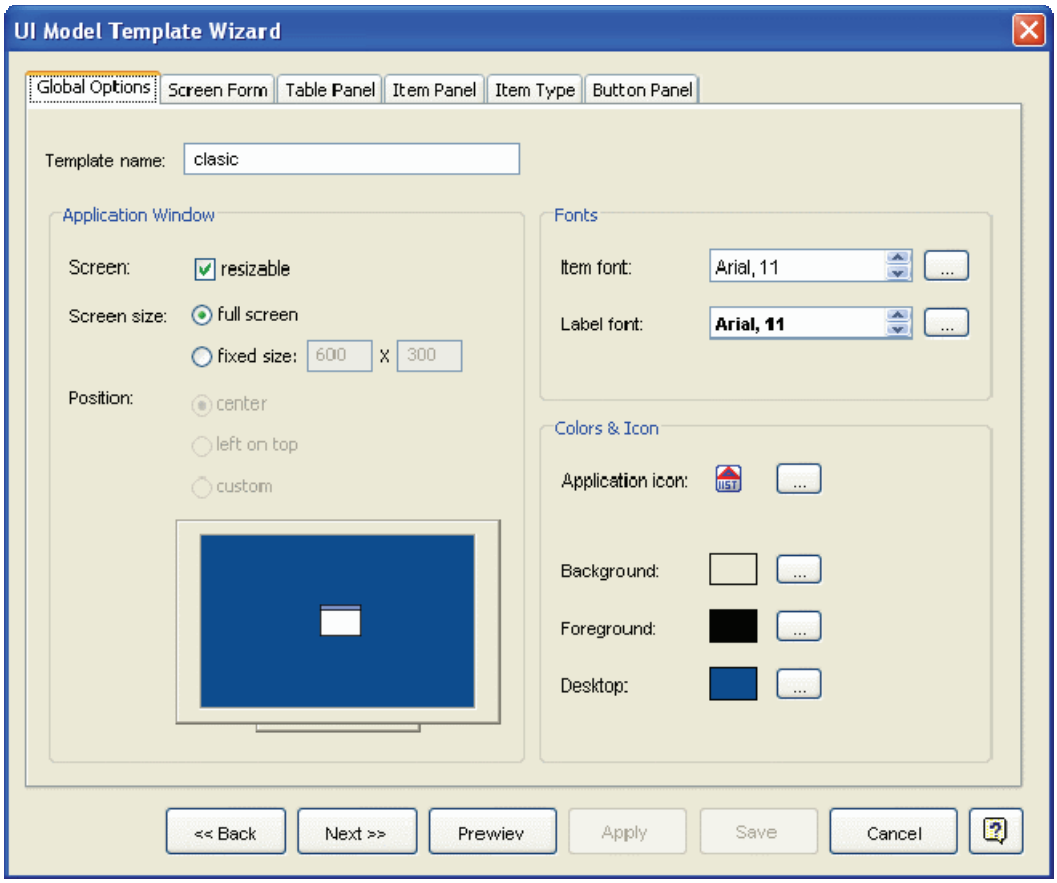
Tip, veličina i format fonta polja za prikaz i ažuriranje podataka i njihovih naziva definiše se uniformno na nivou cijele aplikacije, odnosno svih ekranskih formi. Samim tim, ove opcije definišu se u okviru grupe globalnih atributa.

Detaljna specifikacija globanih atributa šablona korisničkog interfejsa data je u tabeli 4.1.1. Forma sa panelom za specifikaciju globalnih atributa data je na slici 4.1.1.

Globalni atributi (*Global options*)	
Naziv šablona (*Template name*)	Jedinstveni naziv šablona korisničkog interfejsa.
Glavna ekranska forma (*Application window*)	
Glavni prozor aplikacije (*Screen*)	Označava da li je moguće mijenjati veličinu glavnog prozora aplikacije. Čekiranjem opcije *resizable*, korisnik može mijenjati veličinu prozora.
Veličina glavnog prozora aplikacije (*Screen size*)	Veličina glavnog prozora aplikacije može biti definisana kao maksimalna (opcija *full screen*), ili može biti definisana kao fiksna veličina (opcija *fixed size*). U slučaju izabrane fiksne veličine, projektant definiše visinu i širinu glavnog prozora aplikacije.
Pozicija glavnog prozora aplikacije (*Position*)	Pozicija glavnog prozora aplikacije u odnosu na desktop koja se specificira kada prozor ima fiksnu veličinu. Projektant izabira jednu od opcija: • centrirano (*center*), • u lijevom gornjem uglu (*left on top*), • posebno specificirana pozicija (*custom*) - projektant definiše tačne koordinate pozicije glavnog prozora aplikacije.
Fontovi (*Fonts*)	
Font polja za prikaz i ažuriranje podataka (*Item font*)	Specifikacija tipa, veličine i formata fonta polja za prikaz i ažuriranje podataka.
Font naziva polja za prikaz i ažuriranje podataka (*Label font*)	Specifikacija tipa, veličine i formata fonta naziva polja za prikaz i ažuriranje podataka.
Boje i ikone (*Colors & Icon*)	

Ikona aplikacije (*Application icon*)	Ikona glavnog prozora aplikacije.
Pozadina (*Background*)	Boja pozadine glavnog prozora aplikacije.
Boja fonta (*Foreground*)	Boja fonta glavnog prozora aplikacije.
Boja desktop-a (*Desktop*)	Boja desktop-a glavnog prozora aplikacije.

Tabela 4.1.1. *Specifikacija globalnih atributa*



Slika 4.1.1. *Forma za specifikaciju globalnih atributa*

Atributi ekranskih formi

Ova grupa atributa specificira opšte vizuelne karakteristike ekranskih formi koje odgovaraju tipovima komponenti definisanih na tipovima formi projektovanog informacionog sistema. Ono što je zajedničko za sve ekranske forme, a nije pokriveno grupom globalnih atributa, je: mogućnost i način prikaza naslova forme, boje pozadina i slova, kao i načini poziva drugih ekranskih formi iz date forme. Specifikacija šablona podržava tri načina poziva ekranskih formi:

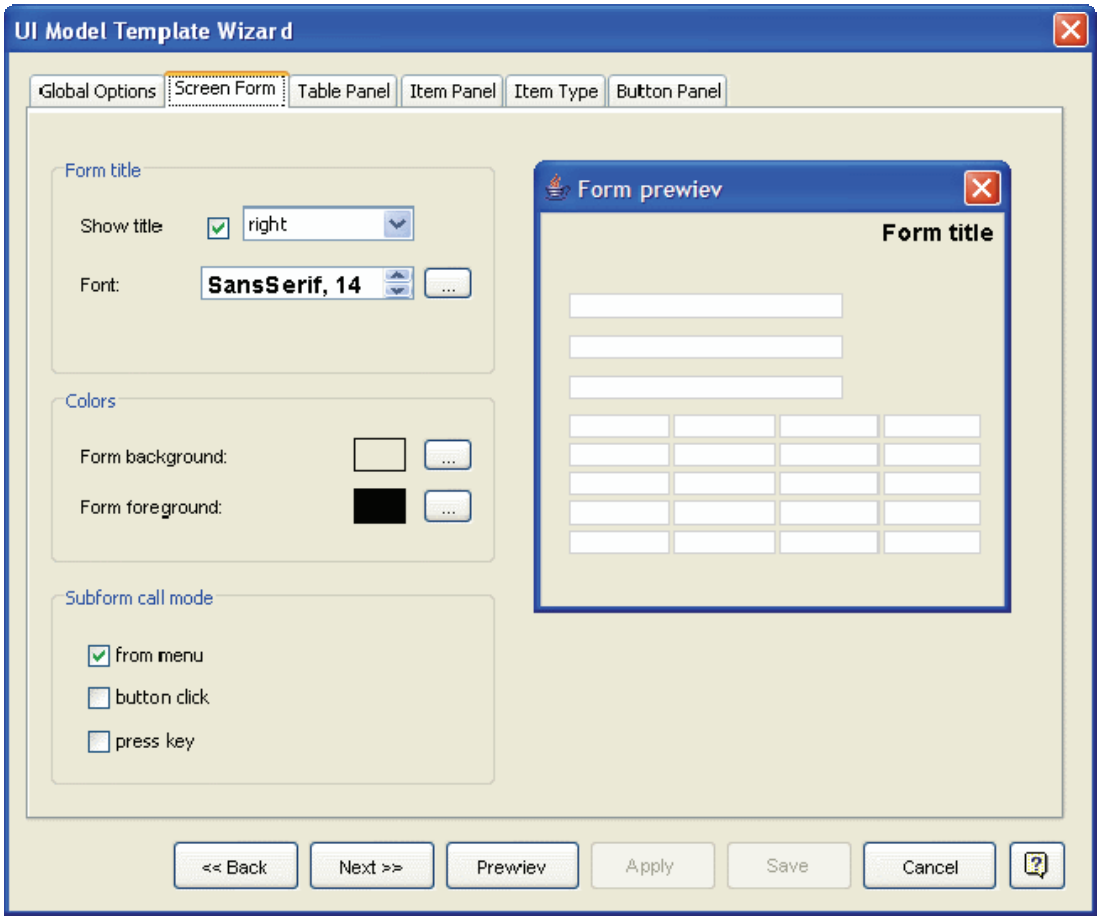
- klikom na stavku menija ekranske forme,
- klikom na dugme i
- pritiskom na odgovarajuću dugmad sa tastature.

Detaljna specifikacija atributa ekranskih formi šablona korisničkog interfejsa data je u tabeli 4.1.2. Forma za specifikaciju atributa ekranskih formi data je na slici 4.1.2. Svaka izmjena

specifikacije nekog od atributa automatski se reflektuje na *Form Prewiev* prikaz, koji projektantu daje uvid kakav efekat data promjena ima na vizuelni izgled ekranskih formi.

Atributi ekranskih formi (*Sreen form*)	
Naslov ekranske forme (*Form title*)	
Prikaži naslov (*Show title*)	Da li je na ekranskim formama prikazan naslov odgovarajućeg tipa komponente.
Font (*Font*)	Tip i veličina fonta naslova ekranskih formi.
Boje (*Colors*)	
Pozadina (*Form background*)	Boja pozadine ekranskih formi.
Boja fonta (*Form foreground*)	Boja fonta ekranskih formi.
Način poziva ekranskih formi (*Subform call mode*)	
Iz menija (*from menu*)	Poziv ekranskih formi iz menija.
Klikom na dugme (*button click*)	Poziv ekranskih formi klikom na dugme.
Pritiskom na dugmad tastature (*press key*)	Poziv ekranskih formi pritiskom na odgovarajuće dugme sa tastature.

Tabela 4.1.2. *Specifikacija atributa ekranskih formi*



Slika 4.1.2. *Forma za specifikaciju atributa ekranskih formi*

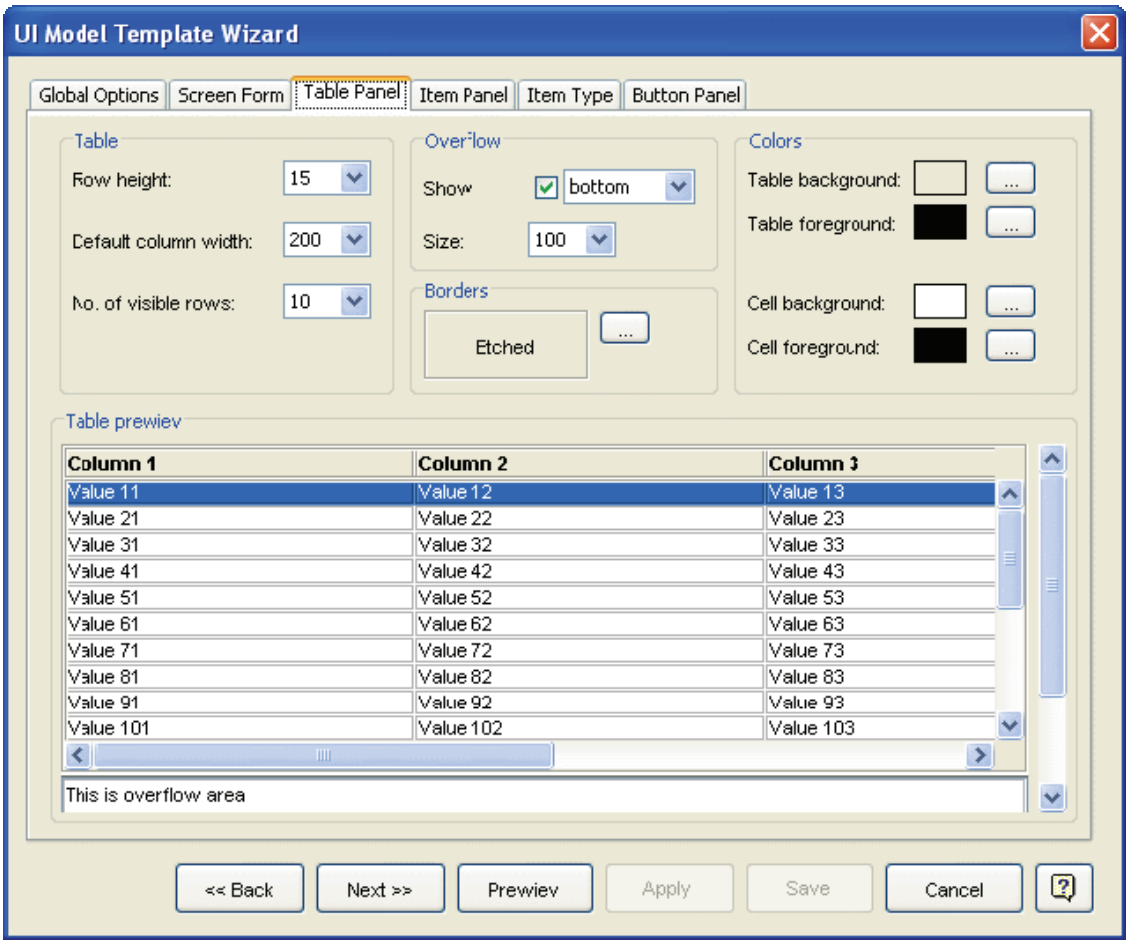
Atributi tabela

Specificiranjem ove grupe atributa projektant modeluje vizuelni izgled svih tabela koje će biti generisane kao sastavni dio korisničkog interfejsa. Ovime se obezbjeđuje uniforman izgled tabela na svim ekranskim formama. Zadavanjem vrijednosti ovih atributa specificiraju se: visina redova tabela, širina kolona tabela, broj vidljivih redova, odnosno visina tabela, format ivica i boje fonta i pozadine. Specijalno, u slučaju postojanja *overflow* grupe polja, atributi tabela specificiraju da li će data grupa polja biti prikazana u posebnoj *overflow* oblasti, kao i veličinu date oblasti.

Detaljna specifikacija atributa tabela šablona korisničkog interfejsa data je u tabeli ispod. Forma za specifikaciju atributa tabela data je na slici 4.1.3. Svaka izmjena specifikacije nekog od atributa automatski se reflektuje na *Table Prewiev* prikaz, koji projektantu daje uvid kakav efekat data promjena ima na vizuelni izgled tabela.

Atributi tabela (*Table panel*)	
Tabela (*Table*)	
Visina redova (*Row height*)	Visina redova u tabelama.
Širina kolona (*Default column width*)	Širina kolona u tabelama.
Broj vidljivih redova (*No. of visible rows*)	Broj vidljivih redova u tabelama kojim je određena visina tabela.
Overflow grupa polja (*Overflow*)	
Prikaz (*Show*)	Indikator da li se overflow grupa polja prikazuje u posebnoj *overflow* oblasti i na kojoj poziciji. Projektant izabira jednu od dvije opcije: • desno od tabele (*right*), • ispod tabele (*bottom*).
Veličina (*Size*)	Visina *overflow* grupe polja. Širina *overflow* grupe polja odgovara širine korespondentne tabele u slučaju pozicije ispod tabele.
Ivice (*Borders*)	
Specifikacija ivica	Specifikacija ivica tabela.
Boje (*Colors*)	
Pozadina tabele (*Table background*)	Boja pozadine tabela.
Boja fonta tabele (*Table foreground*)	Boja fonta tabela.
Pozadina ćelija (*Cell background*)	Boja pozadine ćelija tabela.
Boja fonta ćelija (*Cell foreground*)	Boja fonta teksta u ćelijama tabela.

Tabela 4.1.3. *Specifikacija atributa tabela*



Slika 4.1.3. *Forma za specifikaciju atributa tabela*

Atributi panela

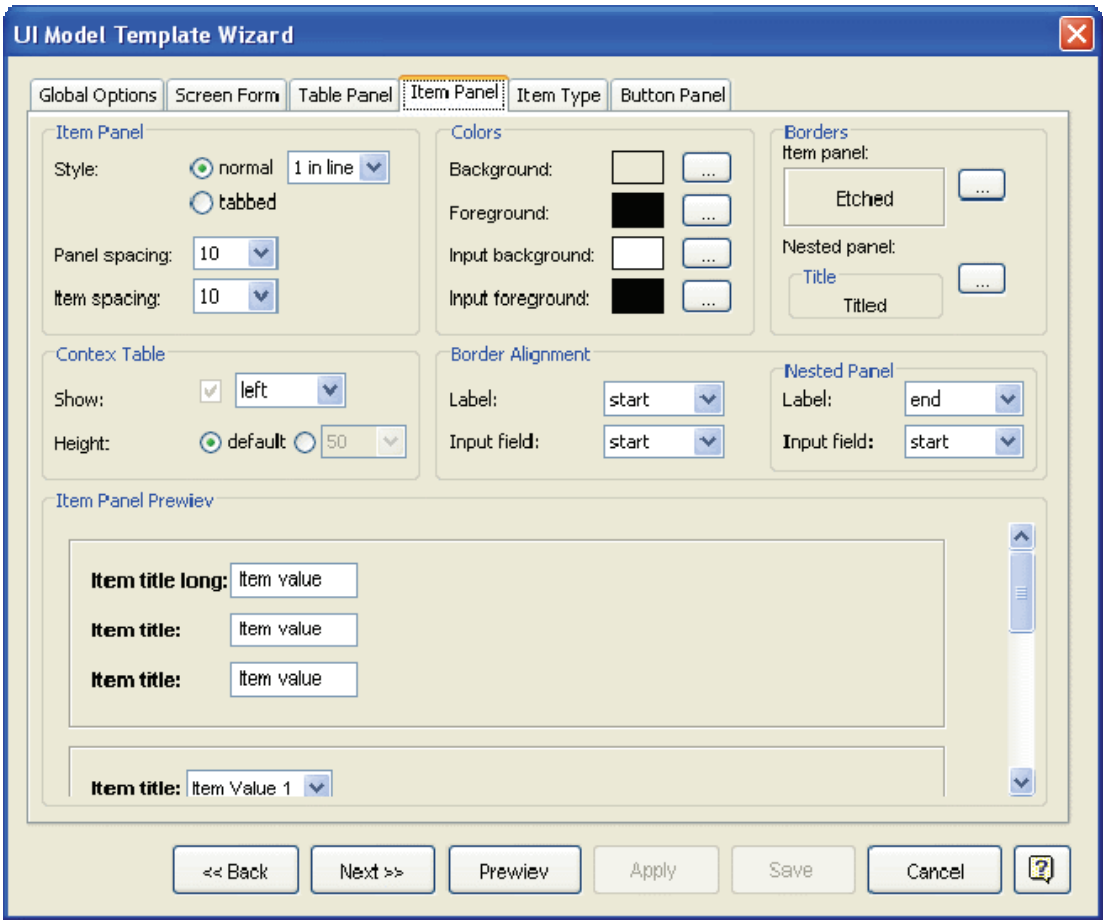
Atributi panela specificiraju način na koji će u okviru korisničkog interfejsa biti prezentirane grupe polja, kada je izabran način prikaza putem panela. Uniforman prikaz panela na svim ekranskim formama obezbjeđuje bolju preglednost podataka, kao i lakše ažuriranje podataka. Zadavanjem vrijednosti atributa panela definiše se tip prikaza panela: normalni prikaz ili prikaz panela sa jezičcima. Takođe je moguće specificirati razmak između pojedinih polja unutar panela, kao i razmak između panela (u slučaju normalnog prikaza). Atributi specificiraju boje fontova i pozadine panela, polja koja su u njima prikazana, ivice panela kao i poravnanje polja unutar panela. Specifikacija ivica i poravnanja polja specijalno se definiše za ugnježđene panele. Kontekstna tabela služi za prikaz kontekstne grupe polja u slučaju da je data grupa definisana i da je za prikaz grupa polja izabran tip panela sa jezičcima. Atributi koji specificiraju kontekstnu tabelu definišu da li će data grupa polja biti prikazana u okviru posebne tabele, kao i visinu date tabele.

Detaljna specifikacija atributa panela šablona korisničkog interfejsa data je u tabeli 4.1.4. Forma za specifikaciju atributa panela data je na slici 4.1.4. Svaka izmjena specifikacije nekog od atributa automatski se reflektuje na *Item Panel Prewiev* prikaz koji projektantu daje uvid kakav efekat data promjena ima na vizuelni izgled panela.

Atributi panela (*Item panel*)	
Panel (*Item panel*)	
Tip panela (*Style*)	Tip panela može biti normalan prikaz panel ili prikaz panela sa jezičcima.
Razmak (*Panel spacing*)	Definisani razmak između panela.
Razmak polja (*Item spacing*)	Definisani razmak između polja prikazanih u panelu.
Boje (*Colors*)	
Pozadina (*Background*)	Boja pozadine panela.
Boja fonta (*Foreground*)	Boja fonta panela.
Pozadina polja (*Input background*)	Boja pozadine polja prikazanih u panelu.
Boja fonta polja (*Input foreground*)	Boja fonta polja prikazanih u panelu.
Specifikacija ivica (*Borders*)	
Panel (*Item panel*)	Tip ivice panela.
Ugnježdeni panel (*Nested panel*)	Tip ivice ugnježdenog panela.
Kontekstna tabela (*Contex table*)	
Prikaz (*Show*)	Način prikaza kontekstne tabele. Projektant izabira jednu od dvije opcije: • lijevo od panela (*left*), • iznad panela (*top*).
Visina (*Height*)	Visina kontekstne tabele.
Poravnanje polja panela (*Item alignment*)	
Naziv polja (*Label*)	Poravnanje naziva polja za prikaz i ažuriranje podataka unutar panela. Projektant izabira jednu od tri opcije: • na početku (*start*), • na kraju (*end*), • nije definisano poravnanje (*none*).
Polje za prikaz i ažuriranje podataka (*Input field*)	Poravnanje polja za prikaz i ažuriranje podataka unutar panela. Projektant izabira jednu od tri opcije: • na početku (*start*), • na kraju (*end*), • nije definisano poravnanje (*none*).
Poravnanje polja ugnježdenog panela (*Nested alignment*)	
Naziv polja (*Label*)	Poravnanje naziva polja za prikaz i ažuriranje podataka unutar ugnježdenih panela. Projektant izabira jednu od tri opcije: • na početku (*start*), • na kraju (*end*), • nije definisano poravnanje (*none*).
Polje za prikaz i ažuriranje podataka (*Input field*)	Poravnanje polja za prikaz i ažuriranje podataka unutar ugnježdenih panela. Projektant izabira jednu

	od tri opcije: • na početku (*start*), • na kraju (*end*), • nije definisano poravnanje (*none*).

Tabela 4.1.4. *Specifikacija atributa panela*



Slika 4.1.4. *Forma za specifikaciju atributa panela*

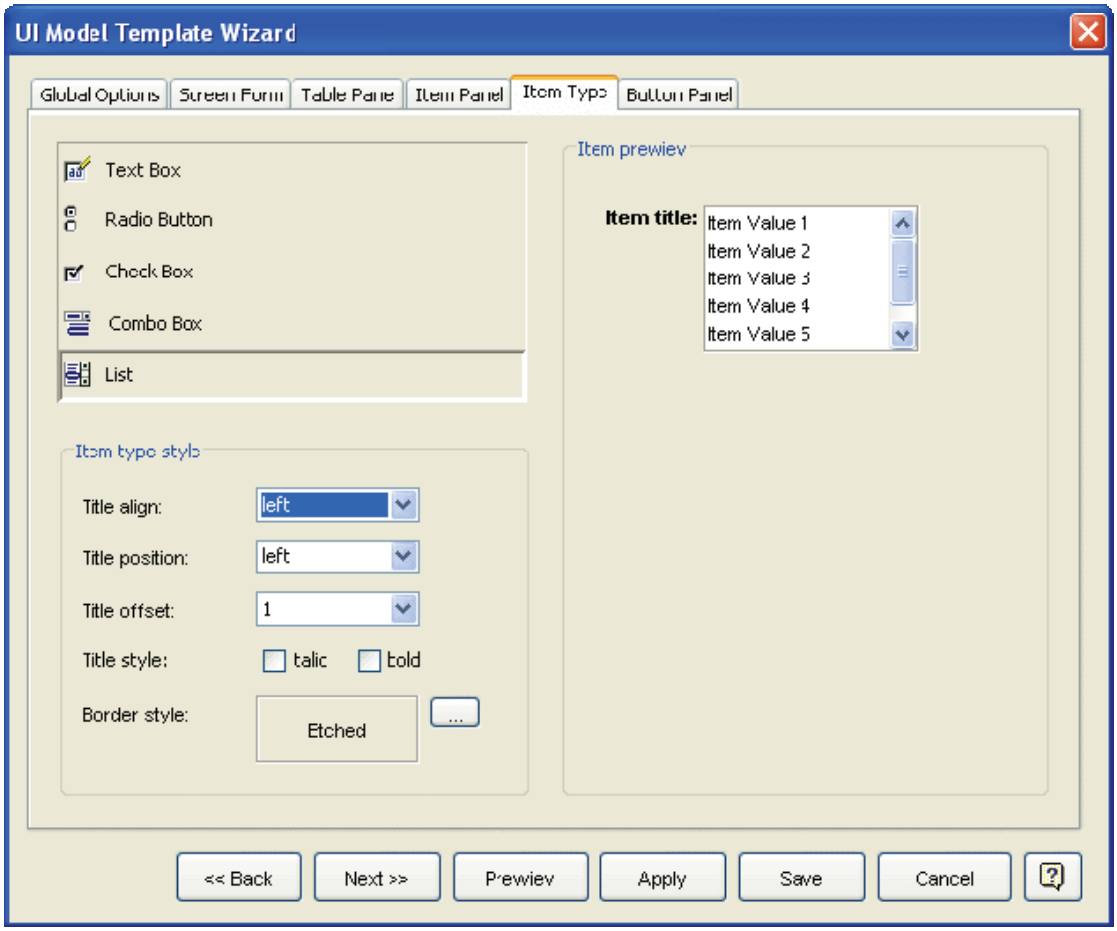
Atributi polja panela

Atributi polja panela specificiraju način na koji će u okviru korisničkog interfejsa biti prikazana polja za prikaz i ažuriranje podataka, u slučaju da je izabran način prikaza putem panela. Šablon podržava 5 tipova prikaza polja: tekstualno polje (*Text box*), radio grupa (*Radio button*), polje za čekiranje (*Check box*), padajuća lista (*Combo box*) i lista izbora (*List*). Uvođenjem polja koja nisu samo tekstualna, omogućeno je da korisnik prilikom ažuriranja podataka, može zadati vrijednosti iz predefinisanih lista.

Specifikacija boja i fontova polja, kao i poravnanja, data je na nivou globalnih atributa i atributa panela. Na ovom nivou atributi specificiraju ivice polja kao i osobine naslova polja (pozicija, poravnanje, razmak između polja i naslova, i format prikaza).

Forma za specifikaciju atributa panela data je na slici 4.1.5. Svaka izmjena specifikacije nekog od atributa automatski se reflektuje na *Item Prewiev* prikaz koji projektantu daje uvid

kakav efekat data promjena ima na vizuelni izgled polja za prikaz i ažuriranje podataka. Detaljna specifikacija atributa polja panela šablona korisničkog interfejsa data je u tabeli 4.1.5.



Slika 4.1.5. *Forma za specifikaciju atributa polja panela*

Atributi polja panela (*Item type*)	
Tip polja panela	Mogući tipovi polja panela su: *Text box*, *Radio button*, *Check box*, *Combo box*, *List*.
Stil polja panela (*Item type style*)	
Poravnanje naziva (*Title align*)	Poravnanje naziva polja panela. Projektant izabira jednu od tri opcije: • lijevo (*left*), • centrirano (*center*), • desno (*right*).
Pozicija naziva (*Title position*)	Pozicija naziva u odnosu na polje za prikaz i ažuriranje podataka. Projektant izabira jednu od tri opcije: • lijevo (*left*), • iznad (*top*), • desno (*right*), • ispod (*bottom*).
Razmak naziva (*Title offset*)	Razmak naziva od polja za prikaz i ažuriranje podataka.

Stil naziva (*Title style*)	Format naslova. Projektant izabira jednu od dvije opcije: • boldovani format (*bold*), • *italic* format.
Tip ivica (*Border style*)	Tip ivica polja za prikaz i ažuriranje podataka.

Tabela 4.1.5. *Specifikacija atributa polja panela*

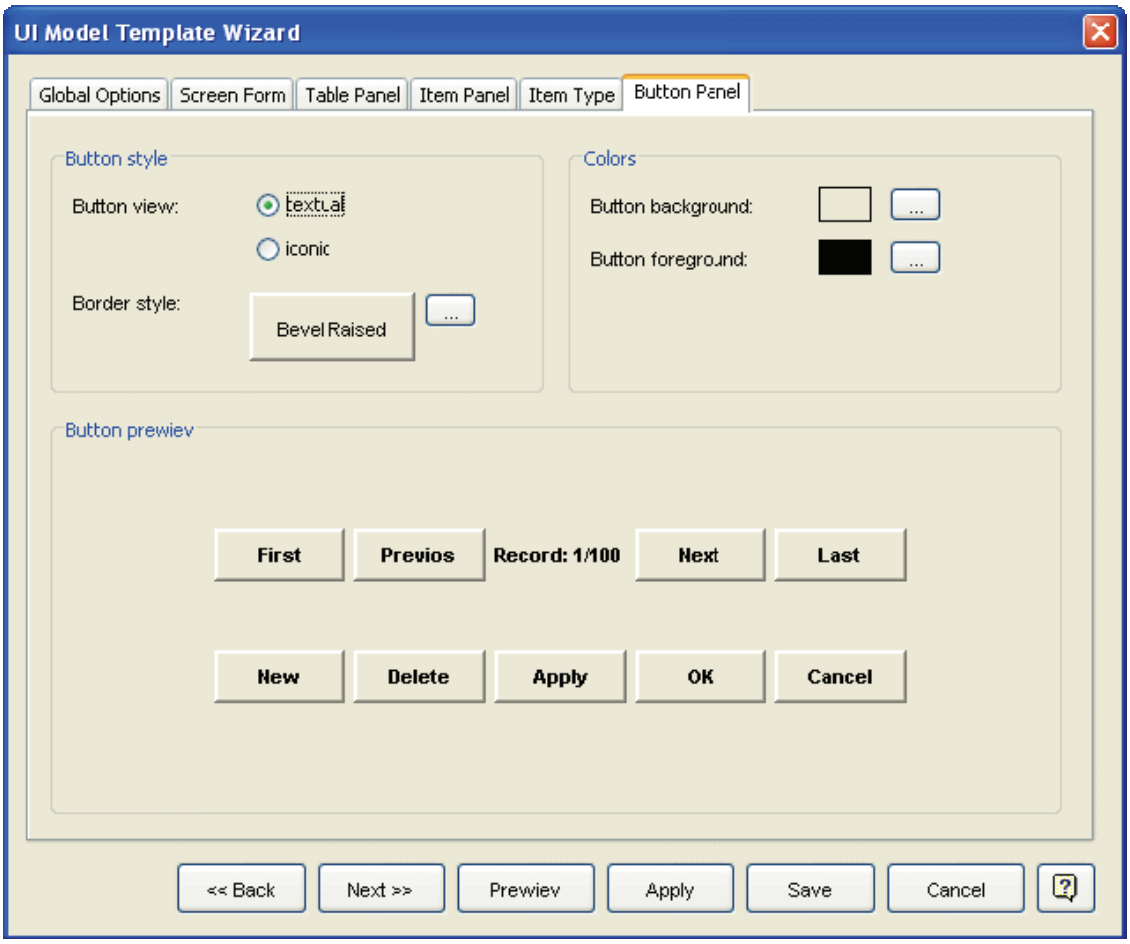
Atributi dugmadi

Šablon korisničkog interfejsa podržava dva tipa prikaza dugmadi na ekranskim formama generisanog korisničkog interfejsa: tekstualni i grafički prikaz. Grafički prikaz podrazumijeva prikaz dugmadi fiksirane veličine sa unaprijed zadatim ikonama koje odgovaraju pojedinim funkcionalnostima dugmadi. Tekstualni prikaz (kao što je dato na slici 4.1.6) podrazumijeva prikaz dugmeta sa tekstualnim ispisom akcije koja odgovara funkcionalnosti dugmeta. Dodatno se putem atributa dugmadi definišu ivice, boje pozadine i fonta dugmadi. Uniforman prikaz i grafička prezentacija dugmadi čini korisnički interfejs intuitivnijim i jednostavnijim za upotrebu.

Detaljna specifikacija atributa dugmadi šablona korisničkog interfejsa data je u tabeli 4.1.6. Forma za specifikaciju atributa dugmadi data je na slici 4.1.6. Svaka izmjena specifikacije nekog od atributa automatski se reflektuje na *Button Prewiev* prikaz koji projektantu daje uvid kakav efekat data promjena ima na vizuelni izgled dugmadi.

Atributi dugmadi (*Button panel*)	
Format dugmadi (*Button style*)	
Prikaz dugmadi (*Button view*)	Tip prikaza dugmadi. Projektant izabira jednu od dvije opcije: • tekstualni prikaz (*textaul*), • grafički prikaz (*iconic*),
Tip ivica (*Border style*)	Tip ivica dugmadi.
Boje (*Colors*)	
Pozadina dugmeta (*Button background*)	Boja pozadine dugmadi.
Boja fonta dugmeta (*Button foreground*)	Boja fonta dugmadi.

Tabela 4.1.6. *Specifikacija atributa dugmadi*



Slika 4.1.6. *Forma za specifikaciju dugmadi*

Specifikacija dijela repozitorijuma koji se odnosi na šablone korisničkog interfejsa data je u prilogu B (slika B.1).

Alatom IIS*UIModeler omogućeno je modelovanje vizuelnog izgleda aplikacija informacionih sistema nezavisno od kreiranja njihovih projektnih specifikacija. Integrisanjem specifikacije izabranog šablona korisničkog interfejsa i projektne specifikacije informacionog sistema, dobija se finalna specifikacija koju je moguće transformisati u aplikaciju koja zadovoljava sva projektovana pravila poslovanja i vizuelne zahtjeve koje je projektant definisao.

Osim što pruža jednostavan interfejs za kreiranje šablona, automatsko generisanje prototipova korisničkog interfejsa, IIS*UIModeler čuva sve kreirane šablone organizovano po grupama. Na taj način isti šablon može biti primjenjen prilikom generisanja prototipova aplikacija različitih informacionih sistema, i obrnuto, ista projektna specifikacija može biti transformisana na više načina, primjenom različitih šablona, pri čemu će se generisati prototipovi aplikacija istih funkcionalnosti a različitog vizuelnog dizajna.

4.2. UIML specifikacija korisničkog interfejsa

Glavna motivacija primjene UIML-a u cilju implementiranja generatora korisničkog interfejsa u okruženju IIS*Studio, jeste mogućnost reprezentacije korisničkog interfejsa u formi

pogodnoj za integrisanje sa konkretnim programskim jezikom, u ovom slučaju programskim jezikom *Java*.

Kompletna specifikacija UIML jezika data je u [1]. UIML je deklarativan, XML kompatibilan meta-jezik za opis korisničkog interfejsa. Prije svega, UIML je kreiran kao rezultat potrebe da se kreira jezik za implementaciju aplikacija na različitim uređajima. Između ostalih, razlozi za uvođenje UIML-a bilu su:

- Omogućavanje implementacije korisničkog interfejsa za bilo koji uređaj bez potrebe za učenjem programskih jezika i upoznavanje sa programskim interfejsima specifičnih za dati uređaj.
- Smanjenje vremena razvoja korisničkog interfejsa.
- Omogućavanje razvoja korisničkog interfejsa i onima koji nisu programeri.
- Mogućnost brzog kreiranja prototipova korisničkog interfejsa.
- Mogućnost lakog proširenja u cilju podrške novim tehnologijama koje će biti razvijene u budućnosti.

UIML je kompatibilan sa W3C XML 1.0 specifikacijom. UIML specifikacija može biti transformisana u HTML, CSS i XSL specifikacije. XSLT transformacije mogu biti korištene za transformaciju UIML specifikacija u XML kompatibilne jezike.

Za potrebe implementacije korisničkog interfejsa danas se koriste različiti programski jezici: jezici bazirani na XML-u (npr. HTML, XHTML, VoiceXML), *JavaScript*, *Java*, C++, itd. Ovi jezici imaju potpuno različite sintakse. Postavlja se pitanje mogućnosti specificiranja korisničkog interfejsa korišćenjem jedne zajedničke sintakse. Tako bi alati za kreiranje korisničkog interfejsa koristili UIML za čuvanje specifikacija, a zatim koristili izabrani programski jezik (npr. HTML ili *Java* koji su danas podržani, ili bilo koji programski jezik koji će biti podržan u budućnosti) za transformisanje UIML specifikacija. Na taj način poboljšava se efikasnost procesa razvoja korisničkog interfejsa, prije svega kroz skraćivanje potrebnog vremena. Kreiranje alata koji podržavaju transformacije specifikacija korisničkog interfejsa u UIML, ili UIML specifikacija u konkretan programski jezik, postaje mnogo jednostavnije. Zahvaljujući ovome, nastaju alati, koji korišćenjem zajedničkog jezika za razmjenu podataka postaju visoko interoperabilni.

UIML, kao meta-jezik, služi za kreiranje UIML specifikacija koje su iskazane putem XML šema jezika. UIML definiše mali skup "moćnih" strukturnih elemenata kao što je `<part>` koji služi za opis elemenata korisničkog interfejsa, ili `<property>` koji služi za opis osobina datih elemenata. UIML strukturni elementi su nezavisni od vrste korisničkog interfejsa, izabrane platforme ili izabranog jezika kojim će biti transformisani.

Da bi se mogle koristiti UIML specifikacije, neophodno je specificirati rečnik (originalni naziv na engleskom: *toolkit vocabulary*). Rečnik suštinski predstavlja DTD šemu UIML specifikacije koja specificira klase UIML elemenata kao i osobine datih klasa. U zavisnosti od potreba razvoja, primjenjuju se različiti pristupi za definisanje rečnika. Tako, u jednom pristupu rečnik može da definiše klase koje su u "jedan na jedan" korespondenciji sa programskim modulima izabranog programskog jezika. Npr. klase UIML elemenata mogu da korespondiraju sa *Java Swing* API klasama. U drugom pristupu, rečnik može da se definiše na apstraktnijem nivou, tako da klase rečnika definišu apstrakcije koje je specificirao dizajner korisničkog interfejsa, kao npr. *Naslov*, *Apstrakt*, *Tekst* za korisničke interfejse za pregled

dokumenata. Na ovaj način, kreiraju se UIML specifikacije na standardizovan način, nezavisno od specifikacije rečnika. Ista UIML specifikacija na taj način može biti interpretirana u različitim programskim okruženjima, primjenom odgovarajućih rečnika.

Zahvaljujući navedenim konceptima, UIML posebno je koristan za kreiranje višejezičkih, dinamičkih korisničkih interfejsa, namjenjenih različitim platformama.

Struktura korisničkog interfejsa pomoću UIML-a logički je predstavljena kao virtuelno drvo elemenata. Ona može da se mijenja tokom vremena izvršavanja generisanog korisničkog interfejsa. Virtuelno drvo elemenata predstavlja se krajnjem korisniku prilikom inicijalnog generisanja korisničkog interfejsa. UIML rečnik definiše element `<structure>` za definisanje inicijalne strukture drveta i element `<restructure>` za dinamičku izmjenu inicijalne strukture. Element `<restructure>` omogućava da se tokom izvršavanja korisničkog interfejsa, u zavisnosti od toga da li su odgovarajući uslovi zadovoljeni, mijenja struktura korisničkog interfejsa brisanjem ili dodavanjem novih elemenata. Npr. dodavanje novog panela na ekransku formu kao posledica klika na dugme, gdje dati panel nije dio inicijalne strukture korisničkog interfejsa.

UIML elemenat `<behavior>` opisuje akcije koje se dešavaju prilikom interakcije krajnjeg korisnika sa korisničkim interfejsom. `<behavior>` element definiše pravilo koje sadrži uslov i niz akcija koje treba da budu izvršene u slučaju da je uslov zadovoljen. Svaki put kada krajnji korisnik interaguje sa korisničkim interfejsom, "okidaju" se događaji koji mogu za posledicu da imaju izvršavanje određenih akcija. Akcije se izvršavaju samo u slučaju da je specificiran uslov zadovoljen. Svaka akcija može da ima za posledicu:

- promjenu osobine elementa korisničkog interfejsa,
- poziv funkcije skript jezika,
- poziv funkcije ili metode pozadinskog objekta ili
- kreiranje događaja.

Na ovaj način UIML obezbjeđuje dovoljno informacija da se omogući razvoj i implementacija takve aplikativne logike koja može dinamički da mijenja korisnički interfejs. Ovo je bitno jer omogućava generisanje funkcionalnih prototipova aplikacija, odnosno prototipova aplikativnih sistema.

Na slici 4.2.1 dat je jedan najjednostavniji primjer UIML specifikacije. Ova specifikacija generiše korisnički interfejs koji sadrži tekst "*Hello World*". Definicija strukture, sadržaja, osobina i ponašanja elemenata korisničkog interfejsa data je u okviru UIML elementa `<interface>`. On definiše inicijalnu strukturu korisničkog interfejsa u kojoj su elementi predstavljeni UIML elementima `<part>`. Elemenat `<style>` definiše osobine pojedinih elemenata strukture korisničkog interfejsa. Pomenuti UIML elementi detaljno su opisani dalje u tekstu.

```
<?xml version="1.0"?>
<!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 3.0 Draft//EN"
"http://uiml.org/dtds/UIML3_0a.dtd">
<uiml>
  <interface>
    <structure>
      <part id="TopHello">
        <part id="hello" class="helloC"/>
      </part>
    </structure>
    <style>
      <property part-name="TopHello" name="rendering">Container
      </property>
      <property part-name="TopHello" name="content">Hello </property>
      <property part-class="helloC" name="rendering">String </property>
      <property part-name="hello" name="content">Hello World!
      </property>
    </style>
  </interface>
  <peers> ... </peers>
</uiml>
```

Slika 4.2.1. *Primjer UIML specifikacije*

Struktura UIML dokumenta

Tipičan UIML dokument sastoji se od sljedećih elemenata:

1. Uvodni element identifikujete verziju XML jezika, kao i lokaciju UIML DTD šeme (primjer je prikazan na slici 4.2.2).

```
<?xml version="1.0"?>
<!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 3.0 Draft//EN"
http://uiml.org/dtds/UIML3_0a.dtd">
```

Slika 4.2.2. *Uvodni UIML element*

2. Korijenski element UIML specifikacije `<uiml>` (slika 4.2.3) sadrži svi druge UIML elemente.

```
<uiml>... </uiml>
```

Slika 4.2.3. *Element* `<uiml>`

Na slici 4.2.4 prikazana je DTD specifikacija `<uiml>` elementa.

```
<!ELEMENT uiml (head?, (template|interface|peers)*)>
```

Slika 4.2.4. *DTD specifikacija elementa* `<uiml>`

Mogući podelementi korijenskog elementa `<uiml>` su:

a) Opcioni `<head>` element koji obezbjeđuje metapodatke o UIML dokumentu (slika 4.2.5). Specifikacija elemenata data u okviru `<head>` elementa ne smatra se dijelom specifikacije korisničkog interfejsa i nema nikakvog uticaja na sam proces generisanja. Glavna svrha `<head>` elementa je da čuva podatke o dokumentu, kao što su podaci o autoru, verziji, datumu kreiranja itd. Element `<head>` sastoji se od jednog ili više `<meta>` elemenata.

```
<head>
      <meta name="Author" content="Jelena Banović"/>
      <meta name="Date" content="February 15, 2010"/>
      <meta name="Description" content= "UIML specifikacija prototipa
                                          korisničkog interfejsa"/>
</head>
```

Slika 4.2.5. *Element* `<head>`

Na slici 4.2.6 prikazana je DTD specifikacija `<head>` elemenata.

```
<!ELEMENT head (meta)*>
```

Slika 4.2.6. *DTD specifikacija elementa* `<head>`

b) Opcioni element `<template>` omogućava ponovnu upotrebu fragmenata UIML koda. Ovim se smanjuje količina UIML koda potrebna za specificiranje korisničkog interfejsa i omogućava standardizovana prezentacija generisanih interfejsa prema krajnjem korisniku. Poznato je da se korisnici bolje snalaze u radu sa korisničkim interfejsima koji su im bliski. Upotrebom `<template>` elementa moguće je jedan fragment UIML koda uključiti na više mjesta u okviru istog UIML dokumenta, kao i fragment iz jednog UIML dokument uključiti u drugi UIML dokument. Primjer elementa `<template>` dat je na slici 4.2.7.

```
<template id="vocab">
      <presentation base="Java_1.3_Harmonia_1.0">
            <d-class id="Frame" used-in-tag="part"
                  maps-type="class" maps-to="java.awt.Frame">
                  ...
            </d-class>
            ...
      </presentation>
</template>
```

Slika 4.2.7. *Element* `<template>`

Na slici 4.2.8 prikazana je DTD specifikacija `<template>` elementa. Detaljna specifikacija podelemenata elementa `<template>` data je u [1].

```
<!ELEMENT template (behavior|d-class|d-component|
constant|content|interface|logic|part|peers|presentation|proper
ty|restructure|rule|script|structure|style)>
<!ATTLIST template
            id NMTOKEN #IMPLIED>
```

Slika 4.2.8. *DTD specifikacija elementa* `<template>`

c) `Element <interface>` sadrži sve druge UIML elemente koji opisuju korisnički interfejs, zajedno sa njihovom strukturom, sadržajem, stilom i ponašanjem (primjer je prikazan na slici 4.2.9).

Mogući podelementi elementa `<interface>` su: `<structure>`, `<style>`, `<content>` i `<behavior>`. Element `<structure>` uključuje skup elemenata koji definišu korisnički interfejs i njihovu organizaciju u odnosu na različite platforme. On se sastoji od jednog ili više podelemenata `<part>` koji specificiraju pojedine elemente korisničkog interfejsa i predstavljaju instance UIML klasa. Putem `<style>` elementa definisane su vrijednosti različitih osobina elemenata interfejsa, analogno *stylesheet* funkcionalnosti u HTML-u. Element `<content>` definiše pojedine riječi, slike i audio sadržaje koji su pridruženi pojedinim elementima u cilju lakšeg prilagođavanja i internacionalizacije korisničkog interfejsa. U okviru elementa `<behavior>` definisano je ponašanje elemenata korisničkog interfejsa tokom interakcije sa krajnjim korisnikom.

```
<interface>
      <structure>... </structure>
      <style> ... </style>
      <content> ... </content>
      <behavior> ... </behavior>
</interface>
```

Slika 4.2.9. *Element* `<interface>`

Na slici 4.2.10. prikazana je DTD specifikacija `<interface>` elemenata.

```
<!ELEMENT interface (structure|style|content|behavior)*>
<!ATTLIST interface
            id NMTOKEN #IMPLIED
            source CDATA #IMPLIED
            how (append|cascade|replace) "replace"
            export (hidden|optional|required) "optional">
```

Slika 4.2.10. *DTD specifikacija elementa* `<interface>`

e) Opcioni element `<peers>` opisuje transformisanje klasa, osobina, događaja i poziva korišćenih u UIML dokumentu u odgovarajuće entitete izabranog programskog okruženja (primjer je prikazan na slici 4.2.11). Element `<peers>` može imati dva podelementa, kojima odgovaraju dva tipa transformacija:

- `<presentation>`, element koji definiše transformisanje klasa i imena elemenata i događaja u odgovarajuće entitete izvan UIML dokumenta, specificiranjem UIML rečnika.

- `<logic>`, element koji definiše transformisanje odgovarajućih UIML elemenata u poslovnu logiku aplikacije.

```
<peers> ... </peers>
```

Slika 4.2.11. *Element* `<peers>`

Na slici 4.2.12 prikazana je DTD specifikacija `<peers>` elemenata.

```
<!ELEMENT peers (presentation|logic)*>
<!ATTLIST peers
            id NMTOKEN #IMPLIED
            source CDATA #IMPLIED
            how (append|cascade|replace) "replace"
            export (hidden|optional|required) "optional">
```

Slika 4.2.12. *DTD specifikacija elementa* `<peers>`

XML komentari, nove linije i prazna polja mogu biti navedeni u specifikacije prije ili poslije svakog od navedenih elemenata.

Skelet jednog UIML dokumenta je prikazan na slici 4.2.13.

```
<?xml version="1.0"?>
<!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 3.0 Draft//EN"
"http://uiml.org/dtds/UIML3_0a.dtd">
<uiml xmlns='http://uiml.org/dtds/UIML3_0a.dtd'>
      <head> ... </head>
      <template> ... </template>
      <interface> ... </interface>
      <peers> ... </peers>
</uiml>
```

Slika 4.2.13. *Skelet UIML dokumenta*

Ovdje su pomenuti najvažniji elementi UIML specifikacije. Definicija svih UIML elemenata data je u [1].

4.3. Generisanje prototipa korisničkog interfejsa

Nakon specificiranja atributa šablona korisničkog interfejsa, projektant ima mogućnost generisanja prototipa korisničkog interfejsa (opcija *Prewiev* na slici 4.1.1). Ova funkcionalnost pruža projektantu veću fleksibilnost prilikom modelovanja vizuelnih aspekata korisničkog interfejsa, u odnosu na neke druge alate. Prototip predstavlja primjer korisničkog interfejsa aplikativnog sistema koji primjenjuje zadatu specifikaciju izabranog šablona, i pri tom obuhvata moguće primjere struktura ekranskih formi i njihovih podelemenata (panela, tabela, različitih polja za prikaz i ažuriranje podataka, dugmadi itd.). Nakon pregleda prototipa, projektant se može vratiti natrag na specifikaciju šablona, napraviti izmjene, i ponovo pokrenuti proces generisanja prototipa korisničkog interfejsa. Ovaj proces je, slično projektovanju informacionog sistema kroz IIS*Case, iterativan, i završava se kad projektant dođe do željenih rezultata.

Algoritam generisanja prototipa korisničkog interfejsa sastoji se iz dva koraka:

- generisanje UIML specifikacije i
- generisanje korisničkog interfejsa.

Na osnovu specifikacije date u [1] automatski se generiše UIML dokument koji specificira korisnički interfejs prototipa koji se sastoji od šest ekranskih formi različitih formata i struktura (slika 3.2.2) kojima je moguće pristupiti iz glavne ekranske forme. UIML specifikacija generiše se na osnovu primjera korisničkog interfejsa aplikativnog sistema i definisanih vrijednosti atributa izabranog šablona. Primjer UIML specifikacije prototipa korisničkog interfejsa dat je u Prilogu C.

Primjenom *rendering* metode, UIML specifikacija se konvertuje u formu prototipa korisničkog interfejsa koji se prezentuje krajnjem korisniku. Izabrano programsko okruženje je *Harmonia Incorporation® Java Renderer*. U pitanju je interpreter koji elemente UIML specifikacije transformiše u *Java AWT/Swing* komponente. U okviru `<presentation>` elementa koji je definisan u okviru `<peers>` segmenta UIML dokumenta definiše se UIML rečnik. Vrijednost atributa `base` (slika 4.3.1) znači da UIML dokument koristi rečnik definisan u dokumentu *Java_1.5_Harmonia_1.0.uiml*. Zadavanjem vrijednosti ovog atributa postiže se da većina *Java AWT*/*Swing* klasa može biti korišćena u funkciji imena elemenata UIML specifikacije.

```
<presentation base="Java 1.5 Harmonia 1.0">
```

Slika 4.3.1. *UIML specifikacija rečnika*

Na slici 4.3.2 prikazan je vrlo jednostavan *Java* kod koji generiše prototip korisničkog interfejsa. U kodu se poziva *Java Renderer* koji specificirani UIML dokument `Specification.uiml` interpretira kao *Java applet*. Nakon kreiranja objekta klase `Renderer`, i poziva njegove metode `renderUIML` sa ulaznim parametrom koji specificira UIML dokument, generiše se prototip.

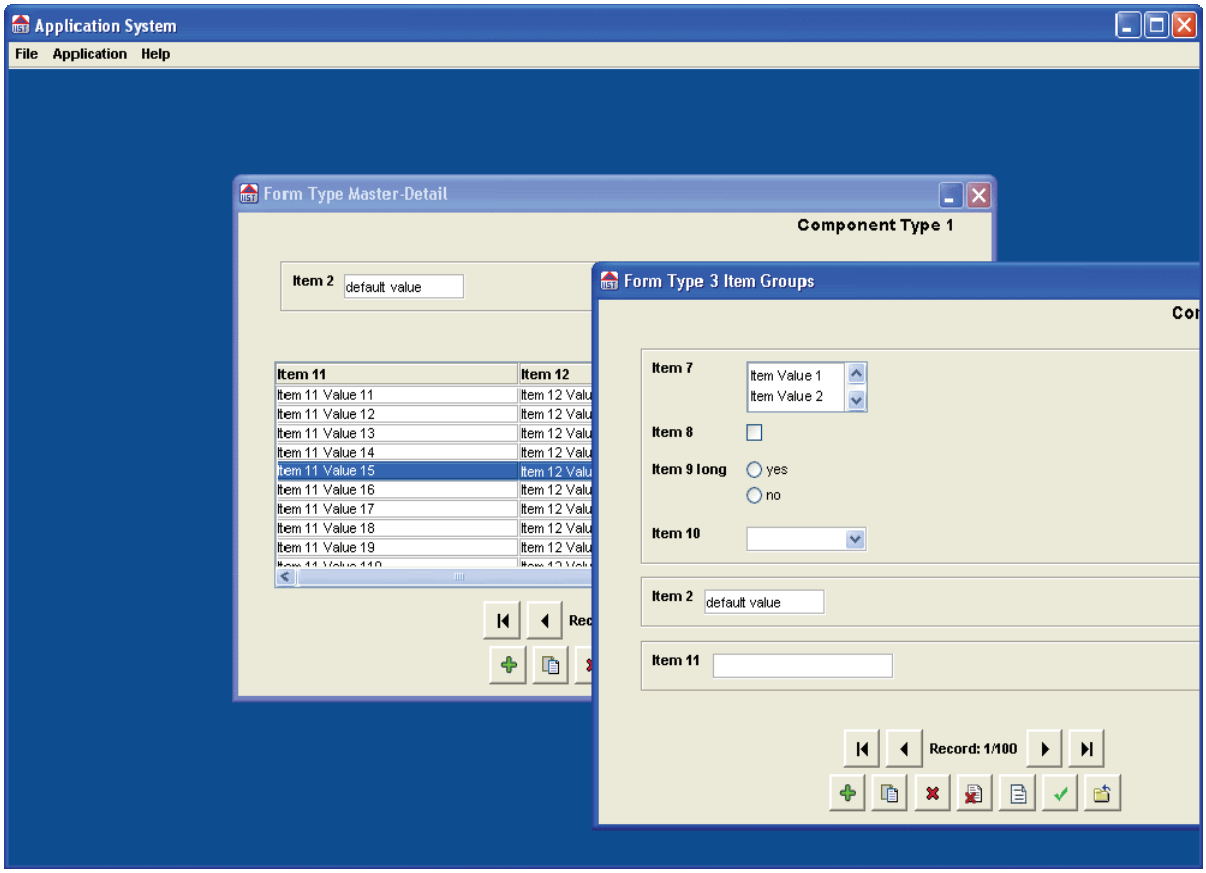
```
public void renderUIMLSpecification()
{
  Renderer r = new Renderer();
  r.runningAsApplet=true;
  String uimlFileName = "Specification.uiml";
  boolean renderOK = r.renderUIML(uimlFileName);
  if(!renderOK) System.out.print("Error!");
}
```

Slika 4.3.2. *Java kod koji generiše prototip na osnovu UIML specifikacije*

Na slici 4.3.3 prikazan je generisani prototip korisničkog interfejsa. Generisanjem prototipa projektant dobija uvid u vizuelni dizajn izabranog šablona i na taj način stiče utisak kako će izgledati prototip aplikacije projektovanog informacionog sistema, bez napuštanja radnog okruženja alata IIS*UIModeler.

Na slici 4.3.3 prikazana su dva tipa ekranske forme na kojima su primjenjeni atributi izabranog šablona: *Form Type Master Detail* koja prikazuje ekransku formu sa *master-detail* strukturom i *Form Type 3 Item Groups* koja prikazuje ekransku formu na kojoj su atributi

grupisani u tri grupe polja koje su predstavljene zasebnim panelima. Kao što je prikazano na slici 3.2.2, osim ovih tipova ekranskih formi, prototip korisničkog interfejsa sadrži i forme: *Form Type Table* koja predstavlja ekransku formu na kojoj su podaci prikazani u tabeli, *Form Type Nested Item Groups* koja daje način prikaza ugnježdenih grupa polja, *Form Type Input Fields with Search* koja prikazuje ekransku formu sa prikazom putem panela sa funkcionalnošću pretrage i *Form Type New Window Subcomponents* koja prikazuje situaciju kada se podkomponente datog tipa komponente prikazuju u okviru nove ekranske forme (suprotno od *master-detail* tipa).



Slika 4.3.3. *Prototip korisničkog interfejsa*

Isti pristup koji se primjenjuje za integraciju specifikacije izabranog šablona u specifikacije prototipova korisničkog interfejsa i njihovo generisanje, primjenjuje se i prilikom generisanja prototipova aplikacija informacionih sistema o čemu će više biti riječi u poglavlju 5.

5. Automatizovano generisanje izvršnih softverskih specifikacija

U procesu projektovanja informacionog sistema, nakon zadavanja projektnih specifikacija i generisanja implementacione šeme baze podataka, IIS*Case, u finalnom koraku, projektantu pruža mogućnost generisanja transakcionih programa, odnosno izvršnih softverskih specifikacija aplikativnih sistema. Postupak generisanja realizovan je implementacijom većeg broja, po obimu složenih algoritama. Rad generatora transakcionih programa svodi se na "pozadinsko" izvršenje tih algoritama, a projektantu se pruža uvid samo u krajnji rezultat – prototip aplikativnog sistema, eventualno sa pridruženim "dijagnostičkim" izvještajem. Ovaj prototip je funkcionalna aplikacija koja zadovoljava sve polazne korisničke funkcionalne i vizuelne zahtjeve, kao i definisana poslovna pravila data kroz specifikaciju projekta i korisničkog šablona. Aplikacija koristi ODBC konekciju za komunikaciju sa prethodno generisanom implementacionom šemom baze podataka i podržava sve funkcionalnosti prikaza i ažuriranja podataka.

Kao što je prikazano na slici 3.1.3.2, proces generisanja transakcionog programa sastoji se iz dva koraka:

1. *Generisanje podšema tipova formi*, implementirano putem algoritama definisanih u [22]. Ovaj korak je preduslov za izvršavanje sljedećeg koraka, jer kao rezultat daje neophodne ulazne informacije na osnovu kojih će biti kreirani SQL upiti, putem kojih će generisana aplikacija komunicirati sa bazom podataka.
2. *Generisanje aplikacije*. Ovaj korak uključuje generisanje UIML dokumenta sa specifikacijom vizuelnih i funkcionalnih aspekata aplikativnog sistema. Generator pokreće *Java Renderer* koji generisanu specifikaciju interpretira kao Java aplikaciju.

Kao i ostali automatizovani postupci, i ovaj je propraćen generisanjem odgovarajućeg izvještaja. U alatu IIS*Case omogućen je pregled arhive izvještaja kreiranih prilikom ranijih postupaka generisanja.

U nastavku teksta ovog poglavlja opisani su postupci koji se primjenjuju prilikom generisanja podšema tipova formi, kao i algoritama koji podržavaju funkcionalnosti prikaza i ažuriranja podataka generisanog transakcionog programa. Takođe je predstavljeno na koji način se specificiraju vizuelne osobine transakcionih programa putem izbora šablona korisničkog interfejsa, specificiranja poslovne aplikacije, kao i vizuelnih atributa, grupe polja i lista

vrijednosti. Specifikacije vizuelnih osobina i funkcionalnosti za prikaz i ažuriranje podataka integriše se sa specifikacijama aplikativnih sistema, i kao rezultat generišu se UIML specifikacije prototipova aplikacija informacionih sisteme, koje se primjenom *rendering* metode izvršavaju u izabranom programskog okruženju.

5.1. Generisanje podšema tipova formi

U [22, 24, 25, 35] predložena je metodologija projektovanja baze podataka informacionog sistema zasnovana na definiciji tipova formi. U okviru ove metodologije, definisani su i opisani u [22, 24] postupci koji obezbjeđuju automatizovanu analizu upotrebljivosti i generisanje podšeme za bilo koji tip forme. Podšema je predstavljena kao par (P, I), gdje je P skup šema relacija dobijen na osnovu skupa šema relacija šeme baze podataka, a I je skup međurelacionih ograničenja dobijen na osnovu skupa međurelacionih ograničenja šeme baze podataka i skupa P.

Za dalje praćenje materije izložene u ovoj tački, neophodno da je čitalac upoznat sa definicijama i značenjima koncepata tipa forme i njenih strukturnih elemenata, na kojima se pristup izložen u ovom radu zasniva, a koji su detaljno opisani u [22, 35, 36, 46], kao i opštim pojmovima i konceptima iz oblasti relacionog modela podataka definisanih u [10, 11, 12, 13, 35, 36, 55], pa se detaljniji opisi tih pojmova ovde izostavljaju.

Definicija 5.1.1. Tip forme $\mathcal{F}$ je primjenljiv (upotrebljiv, primjenljiv za upite) nad šemom baze podataka (S, O), gdje je S skup šema relacija, a O skup međurelacionih ograničenja, ako se za $\mathcal{F}$ može izgenerisati podšema za upite, takva da obezbjeđuje spojivost bez gubitka informacija. □ [22]

Postupak automatizacije analize tipova formi daće projektantu odgovor da li je dati tip forme primjenljiv ili ne. U slučaju da je tip forme primjenljiv daljim postupkom biće određeno da li je primjenljiv samo za upite, ili je primjenljiv i za upite i za ažuriranje. U slučaju identifikacije tipa forme kao upotrebljivog za ažuriranje, postupkom se utvrđuje: (i) koji tipovi komponenti tipa forme su raspoloživi za brisanje torki iz odgovarajuće relacije, (ii) koja obilježja tipa forme se mogu modifikovati i (iii) nad kojim tipovima komponenti tipa forme je moguće dodavati nove torke u odgovarajuće relacije.

U pottačkama 5.1.1- 5.1.5 detaljnije su opisani formalni postupci za generisanje podšema tipa forme i utvrđivanje primjenjivosti datih tipova formi za potrebe čitanja i ažuriranja podataka. U prilogu A navedeni su algoritmi preuzeti iz [22] koji automatizuju navedene postupke. Ovi algoritmi su kompletno implementirani u okviru generatora transakcionih programa alata IIS*Case. U pottačkama se navode i pojmovi formalno definisani i uvedeni u [22] koji su neophodni za dalje praćenje ovog rada. Više informacija o datim pojmovima i o primjerima koji ilustruju date pojmove mogu se naći u [22].

5.1.1. Podšema tipa forme iz klase za ažuriranje

Prilikom kreiranja tipa forme u okviru alata IIS*Case, projektant deklariše da li dati tip forme pripada klasi za ažuriranje $A_{\mathcal{F}}(a)$ ili klasi za upite $A_{\mathcal{F}}(u)$. Tipovi forme iz klase za ažuriranje mogu da se koriste i za čitanje podataka i za ažuriranje relacija baze podataka. Ukoliko se tip

forme $\mathcal{F}$ iz klase $A_{\mathcal{F}}(a)$ koristi samo za upite, za generisanje podšeme koristi se sljedeći skup šema relacija šeme baze podataka:

$$P_u^S = \{(R_i, K_i) \in S \mid Min(R_i) \vee Pod(R_i)\},$$

a ukoliko se tip forme $\mathcal{F}$ koristi i za ažuriranje podataka, podšema će biti formirana na osnovu sljedećeg skupa šema relacija šeme baze podataka:

$$P_a^S = \{(R_i, K_i) \in S \mid Min(R_i) \vee Pod(R_i) \vee Ref(R_i)\},$$

pri čemu predikati $Min(R_i)$, $Pod(R_i)$ i $Ref(R_i)$ predstavljaju uslove, na osnovu kojih će biti određivane šeme relacija koje treba da uđu u sastav skupa P_u^S, odnosno P_a^S. Treba primijetiti da važi $P_u^S \subseteq P_a^S$ [22]. Dalje u tekstu, date su definicije navedenih predikata.

Neka je sa $\mathcal{X}(O)$, gdje je O tip komponente tipa forme $\mathcal{F}$, skup unija po jednog ključa od korijenskog do tipa komponente O.

Proizvoljna šema relacije $(R_i, K_i) \in S$ ispunjava svojstvo $Min(R_i)$ akko je ispunjen uslov:

$$(\exists X_j \in T)(X_j \subseteq (R_i)_\Gamma^+) \wedge$$
$$(\forall (R_k, K_k) \in S)\,(k \neq i \wedge R_k \subseteq (R_i)_\Gamma^+ \Rightarrow X_j \not\subset (R_k)_\Gamma^+),$$

gdje Γ predstavlja skup funkcionalnih zavisnosti zadatih definicijom skupa svih tipova formi, a T skup nosećih grupa tipa forme $\mathcal{F}$. Skup nosećih grupa T definisan je formulom:

$$T(\mathcal{F}) = \{X_i \in \mathcal{X}(O_j) \mid \neg(\exists O_j \in O)\,((O_i, O_j) \in \psi) \wedge Jkl(X_i, O_i)\}$$

gdje je, $\mathcal{X}(O_j)$ skup unija po jednog ključa od korijenskog do tipa komponente O_j, a značenje predikata $Jkl(X_i, O_i)$ je sljedeće: "jedina izabrana unija ključeva lista O_i", odnosno:

$$(\forall X \in \mathcal{X}(O_i))\,(X \neq X_i \Rightarrow X \notin T(\mathcal{F})),$$

a ψ je relacija koja nad skupom tipova komponenti definiše strukturu stabla: $\psi \subseteq O \times O$. [22]

Graf zatvaranja dijagramski predstavlja strukturu šeme baze podataka u relacionom modelu.

Definicija 5.1.1.1. Osnovni graf zatvaranja relacionih šema je graf $\mathcal{G} = (\mathcal{V}, \rho)$, gdje je skupu čvorova $\mathcal{V}$ bijektivno pridružen skup šema relacija $S = \{(R_i, K_i) \mid i = 1,.., m\}$, a relacija $\rho \subseteq \mathcal{V}^2$ definisana na sljedeći način:

$$\rho = \{(R_i, R_j) \in \mathcal{V}^2 \mid R_j \subseteq (R_i)^+_\Gamma \wedge i \neq j \wedge$$
$$\neg(\exists R_k \in \mathcal{V})(k \neq ji \wedge k \neq j \Rightarrow R_j \subseteq (R_k)^+_\Gamma \wedge R_k \subseteq (R_i)^+_\Gamma\}. \square \text{ [22]}$$

Tranzitivni zatvarač relacije ρ bila bi relacija $\tilde{\rho}$, definisana na sljedeći način:

$$\tilde{\rho} = \{(R_i, R_j) \in \mathcal{V}^2 \mid R_j \subseteq (R_i)^+_\Gamma\}.$$

Struktura $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\rho})$ je tranzitivni graf zatvaranja relacionih šema. [22]

Svakom čvoru grafova $\mathcal{G}$ i $\tilde{\mathcal{G}}$ pridružena su dva skupa: skup njemu podređenih čvorova i skup njemu nadređenih čvorova. Uvode se sljedeće funkcije:

- funkcija direktno podređenih čvorova $\Pi : S \to P(S)$:

$$(\forall R_i \in S)(\Pi(R_i) = \{R_j \in S \mid (R_i, R_j) \in \rho\}),$$

- funkcija direktno nadređenih čvorova H: $S \to P(S)$:

$$(\forall R_i \in S)(H(R_i) = \{R_j \in S \mid (R_j, R_i) \in \rho\}),$$

- funkcija direktno podređenih čvorova $\tilde{\Pi} : S \to P(S)$:

$$(\forall R_i \in S)(\tilde{\Pi}(R_i) = \{R_j \in S \mid (R_i, R_j) \in \tilde{\rho}\}),$$

- funkcija direktno nadređenih čvorova $\tilde{H} : S \to P(S)$:

$$(\forall R_i \in S)(\tilde{H}(R_i) = \{R_j \in S \mid (R_j, R_i) \in \tilde{\rho}\}). \text{ [22]}$$

Definicija 5.1.1.2. Skup minimalnih čvorova tipa forme $\mathcal{F}$, u odnosu na graf zatvaranja $\mathcal{G}$, je skup šema relacija:

$$P^S_{min}(\mathcal{F}) = \{(R_i, K_i) \in S \mid Min(R_i)\}. \square \text{ [22]}$$

Definicija 5.1.1.3. Neka je dat graf zatvaranja $\mathcal{G} = (S, \tilde{\Pi}, \tilde{H})$ i skup P^S_{min} minimalnih čvorova tipa forme $\mathcal{F}$. Neke je $K^p_j \in K_j$ primarni ključ šeme (R_j, K_j), $W(\mathcal{F})$ je skup obilježja tipa forme $\mathcal{F}$, $\overline{T}(\mathcal{F}) = \{X_i \subseteq W(\mathcal{F}) \mid (\exists O_j \in O)(X_i \in \mathcal{X}(O_j))\}$, a $\mathcal{X}(O_j)$ skup unija po jednog ključa od korijenskog do objekta O_j. Funkcija relevantnih čvorova datog tipa forme $\Omega : P^S_{min} \to P(S)$ definiše se izrazom:

$$(\forall\, R_i \in P^S_{min})\,(\Omega(R_i) = \{R_j \in \tilde{\Pi}(R_i) \setminus R_i \mid$$

$$(R_j \cap W(\mathcal{F})) \setminus K^p_j \neq \emptyset \vee (\exists\, X_k \in \bar{T}(\mathcal{F}))(\exists\, K_j \in K_j)\,(X_k = K_j)\}).\ \square\ [22]$$

Skup relevantnih čvorova za tip forme $\mathcal{F}$ je skup:

$$R_c(\mathcal{F}) = P^S_{min} \cup P^S_{rlp},$$

gdje je P^S_{rlp} skup relevantnih podređenih čvorova:

$$P^S_{rlp} = \{(R_j, K_j) \in S \mid (\exists\, R_i \in P^S_{min})(R_j \in \Omega(R_i))\}.[22]$$

Proizvoljna šema relacije $(R_j, K_j) \in S$ ispunjava svojstvo $Spd(R_j)$ akko je ispunjen uslov:

$$(\exists\, R_i \in P^S_{min}(\mathcal{F}))(R_j \in \tilde{\Pi}(R_i) \setminus \{R_i\} \wedge (\exists\, R_k \in \Omega(R_i))\,(R_k \subset (R_j)^+_\Gamma)).$$

Skup podređenih čvorova tipa forme $\mathcal{F}$ je skup: $P^S_{spd}(\mathcal{F}) = \{(R_j, K_j) \in S \mid Spd(R_j)\}.[22]$

Proizvoljna šema relacije $(R_j, K_j) \in S$ ispunjava svojstvo $Pod(R_j)$ akko je ispunjen uslov:

$$(R_j \in P^S_{rlp}) \vee\ [(R_j \in P^S_{spd}) \wedge$$

$$(\exists\, R_i \in (P^S_{rlp} \cup P^S_{spd}) \cap \Pi(R_j))(K^p_i \not\subset K^p_j) \wedge$$

$$(\exists\, R_i \in P^S_{min})(\exists\, R_k \in \Omega(R_i))].$$

Definicija 5.1.1.4. Skup potrebnih podređenih čvorova tipa forme $\mathcal{F}$, u odnosu na graf zatvaranja $\mathcal{G}$, je skup šema relacija:

$$P^S_{pod}(\mathcal{F}) = \{(R_j, K_j) \in S \mid Pod(R_j)\}.[22]$$

Definicija 5.1.1.5. Skup šema relacija $P^S_u(\mathcal{F})$ šeme baze podataka, na osnovu kojeg će se generisati podšema za upite datog tipa forme $\mathcal{F}$ iz klase $A_{\mathcal{F}}(a)$, predstavlja uniju skupa minimalnih i skupa potrebnih podređenih čvorova tipa forme:

$$P^S_u(\mathcal{F}) = P^S_{min}(\mathcal{F}) \cup P^S_{pod}(\mathcal{F}).\ \square\ [22]$$

5.1.2. Analiza primjenljivosti za ažuriranje tipa forme iz klase $A_{\mathcal{F}}$ (a)

Neka je dat tip forme $\mathcal{F}$, sa skupom tipova komponenti O i strukturom stabla nad skupom tipova komponenti: $\psi \subseteq O \times O$. Dalje u tekstu su definisani INSERT, DELETE i UPDATE kriterijumi koji redom definišu sljedeće skupove:

- $OIns(\mathcal{F})$ - skup tipova komponenti tipa forme $\mathcal{F}$ koji se mogu koristiti za dodavanje novih podataka,
- $ODel(\mathcal{F})$ - skup tipova komponenti tipa forme $\mathcal{F}$ koji se mogu koristiti za brisanje podataka,
- $WUpd(\mathcal{F})$ - skup obilježja tipa forme $\mathcal{F}$ čije se vrijednosti mogu modifikovati.

A) INSERT kriterijum

Neka je dat tip komponente $O_i \in O$. Tip komponente O_i zadovoljava INSERT kriterijum $Ins(O_i, R_j)$, u odnosu na šemu relacije $(R_j, K_j) \in P_u^S(\mathcal{F})$, akko je ispunjen uslov:

$$(\forall K_j \in K_j)(\exists X_i \in \mathcal{X}(O_i))(X_i = K_j) \wedge (\forall A \in R_j \setminus W(\mathcal{F}))(w \in dom(A)),$$

gdje je simbolom *w* označena nula (nepoznata) vrijednost.

Prvim članom konjukcije, zahtjeva se da svi ekvivalentni ključevi šeme R_j odgovaraju nekoj od unija po jednog ključa od korijenskog do datog tipa komponente O_i. Drugi član konjukcije zahtjeva da za sva obilježja iz skupa $R_j \setminus W$, budu dozvoljene nula-vrijednosti, inače insertovanje ne bi moglo da se sprovede.

Skup tipova komponenti na osnovu kojih se može vršiti dodavanje novih torki u odgovarajuće relacije će biti:

$$OIns(\mathcal{F}) = \{(O_i, R_j) \mid O_i \in O(\mathcal{F}) \wedge R_j \in P_u^S(\mathcal{F}) \wedge Ins(O_i, R_j)\}.[22]$$

B) DELETE kriterijum

Neka je dat tip komponente $O_i \in O$, tipa forme $\mathcal{F}$. Tip komponente O_i zadovoljava DELETE kriterijum $Del(O_i, R_i)$, u odnosu na šemu relacije $(R_j, K_j) \in P_u^S(\mathcal{F})$, akko je ispunjen uslov:

$$(\exists X_i \in \mathcal{X}(O_i))(\exists K_j \in K_j)(X_i = K_j).$$

Skup tipova objekata na osnovu kojih se može vršiti brisanje torki iz odgovarajuće relacije će biti:

$$ODel(\mathcal{F}) = \{(O_i, R_j) \mid O_i \in O(\mathcal{F}) \wedge R_j \in P_u^S(\mathcal{F}) \wedge Del(O_i, R_j)\}.[22]$$

C) UPDATE kriterijum

Neka je

$$K_p(\mathcal{F}) = \{A \in W(\mathcal{F}) \mid (\exists O_i \in O)(\exists K_i \in K_o^i)(A \in K_i)\},$$

i neka je $W_B(\mathcal{F})$ skup elementarnih obilježje tipa forme:

$$W_B(\mathcal{F}) = \{A \in W(\mathcal{F}) \mid f(A) = \emptyset\},$$

gdje je $f: U \to P(U)$ funkcija izvođenja, a $f(A)$ predstavlja podskup univerzalnog skupa obilježja iz kojeg se izvodi uloga obilježja A.

Dato obilježje $A \in W(\mathcal{F})$ će zadovoljavati UPDATE kriterijum $Upd(A)$ akko:

$$A \in W_B(\mathcal{F}) \setminus K_p(\mathcal{F}) \wedge$$

$$(\exists O_i \in O)(\exists R_j \in P_u^S(\mathcal{F}))(Del(O_i, R_j) \wedge A \in N_i \cap R_j),$$

gdje je N_i skup obilježja tipa komponente O_i.

Na osnovu predikata $Upd(A)$ formiramo skup svih obilježja raspoloživih za modifikovanje

$$WUpd(\mathcal{F}) = \{A \in W(\mathcal{F}) \mid Upd(A)\}.$$ [22]

Proizvoljna šema relacije $(R_k, K_k) \in S$ (odnosno čvor grafa zatvaranja $\mathcal{G}$) ispunjava svojstvo $Ref(R_k)$ akko je ispunjen uslov:

$$[(\exists (O_i, R_i) \in ODel(\mathcal{F}))(R_k \in H(R_i)) \vee$$

$$(R_k \in \Pi(R_i) \wedge WUpd(\mathcal{F}) \cap K_k^p \neq \emptyset))] \vee$$

$$[(\exists (O_i, R_i) \in OIns(\mathcal{F}))(R_k \in \Pi(R_i) \wedge K_k^p \subseteq R_i)].$$

Definicija 5.1.2.1. Skup referentnih čvorova tipa forme $\mathcal{F}$, u odnosu na graf zatvaranja $\mathcal{G}$, je skup šema relacija:

$$P_{ref}^S(\mathcal{F}) = \{(R_j, K_j) \in S \mid Ref(R_j)\}.$$

Definicija 5.1.2.2. Skup šema relacija $P_a^S(\mathcal{F})$, šeme baze podataka na osnovu kojeg će se generisati podšema za ažuriranje datog tipa forme $\mathcal{F}$, predstavlja uniju minimalnih, skupa potrebnih podređenih i skupa referentnih čvorova tipa forme:

$$P_a^S(\mathcal{F}) = P_{min}^S(\mathcal{F}) \cup P_{pod}^S(\mathcal{F}) \cup P_{ref}^S(\mathcal{F}).$$ [22]

5.1.3. Analiza primjenljivosti i načina upotrebe tipa forme iz klase $A_{\mathcal{F}}$(u)

A) ***Primjenljivost za upite***

U [22] nabrojani su i detaljno objašnjeni sljedeći kriterijumi koji moraju da budu zadovoljeni da bi dati tip forme iz klase $A_{\mathcal{F}}$(u) bio primjenljiv za upite (svi navedeni kriterijumi preuzeti su sa originalnom numeracijom iz [22]):

1. $(\forall A \in W(\mathcal{F}) \setminus U)\,(f(A) = \emptyset))$ (4.16)

   Svako obilježje definisano na tipu forme, koje ne pripada skupu obilježja šeme baze podataka nije elementarno obilježje već mu se vrijednost izračunava na osnovu obilježja od kojih je izveden.

2. $\Gamma \models F(\mathcal{F})|_U$ (4.17)

   Projekcija skupa funkcionalnih zavisnosti definisanih na tipu forme $\mathcal{F}$ na skup obilježja šeme baze podataka mora biti logička posledica skupa funkcionalnih zavisnosti Γ.

3. $K_p(\mathcal{F}) \setminus U = \emptyset \Rightarrow$

   $[(\forall X_i \in T(\mathcal{F}))\,(\exists R_j \in S) \wedge (X_i \subseteq (R_i)^+_\Gamma)] \vee$ (4.18)

   $[(\exists J_y \in SJ)\,(\bigcup_{i=1}^{k} X_i = \bigcup_{i=1}^{m} Y_i) \wedge (\forall Y_i \in T_y)\,(\exists X_j \in T)\,(Y_i \subseteq X_j)]$ (4.20)

   Uslov da na tipu forme $\mathcal{F}$ nisu definisana primarna obilježja koja nisu sadržana u skupu obilježja šeme baze podataka, implicira uslov 4.18 ili 4.20. Uslov 4.18 podrazumijeva da za svaku noseću grupu obilježja definisanu na tipu forme $\mathcal{F}$, postoji šema relacije, takva da je data noseća grupa obilježja sadržana u zatvaraču skupa obilježja date šeme relacije u odnosu na skup Γ. Neka je tipom forme $\mathcal{F}$ definisana zavisnost spoja $J(\mathcal{F}) = \{X_1, \ldots, X_k\}$. Ispunjenošću uslova 4.20 konstatuje se da u skupu netrivijalnih zavisnosti spoja *SJ*, definisanih tipovima formi iz klase $A_{\mathcal{F}}$(a), postoji zavisnost spoja $J_y = \bowtie (Y_1, \ldots, Y_m)$, gdje je $T_y = \{Y_1, \ldots, Y_m\}$, koja za logičku posledicu ima zavisnost spoja $J(\mathcal{F})$.

4. $K_p(\mathcal{F}) \setminus U \neq \emptyset \Rightarrow$

   $(\forall A \in K_p(\mathcal{F}) \setminus U)\,(\exists A^r \in W_{rek}(A) \cup \{A\})$

   $(\exists A_B \in f(A^r) \cap U)\,(dom(A) = dom(A_B))$. (4.22)

   Uslov da na tipu forme $\mathcal{F}$ postoje definisana primarna obilježja koja nisu sadržana u skupu obilježja šeme baze podataka, implicira uslov 4.22. $W_{rek}(A)$ je skup svih izvedenih obilježja, koja su učestvovala u rekurzivnoj primjeni funkcije izvođenja na primarno obilježje A definisano na tipu forme $\mathcal{F}$ koje nije sadržano u skupu obilježja šeme baze podataka. Prema ograničenju zadatom uslovom 4.22 svako obilježje

$A \in K_p(\mathcal{F}) \setminus U$ treba zamijeniti odgovarajućim skupom $f(A^r) \cap U$.

5. $K_p(\mathcal{F}) \setminus U \neq \emptyset \Rightarrow$

$$[(\forall X_i^r \in T^r(\mathcal{F}))(\exists R_j \in S)(X_i^r \subseteq (R_i)_\Gamma^+)] \vee \tag{4.24}$$

$$[(\exists J_y \in SJ)(\bigcup_{i=1}^{k} X_i^r = \bigcup_{i=1}^{m} Y_i) \wedge (\forall Y_i \in T_y)(\exists X_j^r \in T^r)(Y_i \subseteq X_j^r)] \tag{4.26}$$

Uslov da na tipu forme postoje definisana primarna obilježja koja nisu sadržana u skupu obilježja šeme baze podataka implicira uslov 4.24 ili 4.26.

Skup $T^r(\mathcal{F})$ definisan je na sljedeći način:

$$T^r(\mathcal{F}) = \{X_i^r \mid (\exists X_i \in T(\mathcal{F}))(X_i^r = (X_i \cap U) \cup \left[\bigcup_{A \in X_i \setminus U} (f(A^r) \cap U)\right])\},$$

gdje je $X_i^r = \{X_i \cap U\} \cup \left[\bigcup_{A \in X_i \setminus U} (f(A^r) \cap U)\right]$.

Uslov 4.24 podrazumijeva da za svaki elemenat X_i^r skupa $T^r(\mathcal{F})$ postoji šema relacije, takva da je X_i^r sadržan u zatvaraču skupa obilježja date šeme relacije u odnosu na skup Γ. Ispunjenošću uslova 4.26 konstatuje se da u skupu netrivijalnih zavisnosti spoja *SJ*, definisanih tipovima formi iz klase $A_{\mathcal{F}}(a)$, postoji zavisnost spoja $J_y = (Y_1, \ldots, Y_m)$, gdje je $T_y = \{Y_1, \ldots, Y_m\}$, koja za logičku posledicu ima zavisnost spoja $J^r(\mathcal{F}) = \{X_1^r, \ldots, X_k^r\}$.

B) ***Primjenljivost za ažuriranje***

U [22] nabrojani su i detaljno objašnjeni sljedeći kriterijumi koji moraju da budu zadovoljeni da bi dati tip forme iz klase $A_{\mathcal{F}}(u)$ bio primjenljiv za ažuriranje (svi navedeni kriterijumi preuzeti su sa originalnom numeracijom iz [22]):

1. $(\forall A \in W(\mathcal{F}) \setminus U)(f(A) = \emptyset))$ (4.16)

   Svako obilježje definisano na tipu forme, koje ne pripada skupu obilježja šeme baze podataka nije elementarno obilježje već mu se vrijednost izračunava na osnovu obilježja od kojih je izveden.

2. $\Gamma \models F(\mathcal{F})|_U$ (4.17)

   Projekcija skupa funkcionalnih zavisnosti definisanih na tipu forme $\mathcal{F}$ na skup obilježja šeme baze podataka mora biti logička posledica skupa funkcionalnih zavisnosti Γ.

3. $K_p(\mathcal{F}) \setminus U \neq \emptyset$ (4.28)

   Na tipu forme $\mathcal{F}$ nisu definisana primarna obilježja koja nisu sadržana u skupu obilježja šeme baze podataka.

4. $(\forall A \in W(\mathcal{F}) \setminus W_B(\mathcal{F}))(f(A) \subseteq W(\mathcal{F}))$ (4.29)

   $W_B(\mathcal{F}) = \{A \in W(\mathcal{F}) \mid f(A) = \emptyset\}$ predstavlja skup elementarnih obilježja tipa forme $\mathcal{F}$. Uslov 4.29 ispunjen je ako je izvedeno obilježje nastalo izvođenjem od skupa elementarnih obilježja.

5. $(\forall X_i \in T(\mathcal{F}))(\exists R_j \in S)(X_i \subseteq (R_j)^+_\Gamma)$ (4.18)

   Uslov 4.18 podrazumijeva da za svaku noseću grupu obilježja definisanu na tipu forme $\mathcal{F}$, postoji šema relacije, takva da je data noseća grupa obilježja sadržana u zatvaraču skupa obilježja date šeme relacije u odnosu na skup Γ.

6. $\neg (X_i \rightarrow \emptyset \in NF(\mathcal{F}))(\exists P \rightarrow Q \in \Gamma^+)(QP \subseteq X_i \wedge Q \not\subset P)$ (4.30)

   Uslovom 4.30 provjerava se da li je testiranim tipom forme definisan nefunkcionalni odnos nad skupom obilježja nad kojim već postoji funkcionalna zavisnosti iz skupa Γ^+.

5.1.4. Tabela analize primjenljivosti i načina upotrebe tipa forme

Sistematizovani prikaz analize primjenljivosti i načina upotrebe datog tipa forme dat je u tabeli odlučivanja 5.1.4.1.

(4.16)	F	-	T	T	T	T	T	T	T	T	T
(4.17)	-	F	T	T	T	T	T	T	T	T	T
(4.28)	-	-	F	F	F	F	T	T	T	T	T
(4.22)	-	-	F	T	T	T	-	-	-	-	-
(4.24)	-	-	-	F	F	T	-	-	-	-	-
(4.26)	-	-	-	F	T	-	-	-	-	-	-
(4.18)	-	-	-	-	-	-	T	T	T	F	F
(4.30)	-	-	-	-	-	-	T	T	F	-	-
(4.29)	-	-	-	-	-	-	T	F	-	-	-
(4.20)	-	-	-	-	-	-	-	-	-	F	T
Z00	X	X	X	X						X	
Z01					X						
Z02						X					
Z03							X				
Z04								X	X		
Z05											X

Tabela 5.1.4.1. *Tabela analize primjenljivosti i načina upotrebe tipa forme*

Značenje zaključaka, datih u tabeli 5.1.4.1:

- **Z00** – tip forme proglašava se neprimjenljivim.
- **Z01** – tip forme je primjenljiv za upite. Podšema se formira na osnovu izraza:

$$P_u^S(\mathcal{F}) = P_{min}^{sr}{}'(\mathcal{F}) \cup P_{pod}^{sr}{}'(\mathcal{F}) \cup P_{rek}^{S}(\mathcal{F}),$$

gdje se $P_{min}^{sr}{}'(\mathcal{F})$ formira prema predikatu $Min'(\mathrm{R_i})$, na osnovu skupa $P_{min}^{sr}{}'(\mathcal{F})$ formira se skup $P_{pod}^{sr}{}'(\mathcal{F})$, a skup $P_{rek}^{S}(\mathcal{F})$ formira se po predikatu $Rek(\mathrm{R_i})$. Predikat $Rek(\mathrm{R_i})$ definisan je na sljedeći način:

$$(\exists\, \mathrm{A}' \in \mathrm{K_p}(\mathcal{F}) \setminus \mathrm{U})(\exists\, \mathrm{A}'' \in W_{rek}(\mathrm{A}'))(f(\mathrm{A}') \cap \mathrm{U} = \mathrm{K_i} \wedge \mathrm{K_i} \in K_i),$$

dok je predikat $Min'(\mathrm{R_i})$ definisan sa:

$$(\exists\, \mathrm{Y_j} \in \mathrm{T_y})(\mathrm{Y_j} \subseteq (\mathrm{R_i})_\Gamma^+) \wedge$$
$$(\forall (\mathrm{R_k}, K_k) \in S)(\mathrm{k} \neq \mathrm{i} \wedge \mathrm{R_k} \subseteq (\mathrm{R_i})_\Gamma^+ \Rightarrow \mathrm{Y_j} \not\subset (\mathrm{R_k})_\Gamma^+).$$

- **Z02** – tip forme je primjenljiv za upite. Podšema se formira na osnovu izraza:

$$P_u^S(\mathcal{F}) = P_{min}^{sr}(\mathcal{F}) \cup P_{pod}^{sr}(\mathcal{F}) \cup P_{rek}^{S}(\mathcal{F}),$$

gdje se $P_{min}^{sr}(\mathcal{F})$ formira prema predikatu $Min^{\mathrm{r}}(\mathrm{R_i})$, na osnovu skupa $P_{min}^{sr}(\mathcal{F})$ formira se skup $P_{pod}^{sr}(\mathcal{F})$, a skup $P_{rek}^{S}(\mathcal{F})$ formira se prema predikatu $Rek(\mathrm{R_i})$. Predikat $Min^{\mathrm{r}}(\mathrm{R_i})$ definisan je na sljedeći način:

$$(\exists\, \mathrm{X_j^r} \in \mathrm{T^r})(\mathrm{X_j^r} \subseteq (\mathrm{R_i})_\Gamma^+) \wedge$$
$$(\forall (\mathrm{R_k}, K_k) \in S)(\mathrm{k} \neq \mathrm{i} \wedge \mathrm{R_k} \subseteq (\mathrm{R_i})_\Gamma^+ \Rightarrow \mathrm{X_j^r} \not\subset (\mathrm{R_k})_\Gamma^+).$$

- **Z03** – tip forme je primjenljiv za ažuriranje. Podšema se formira na osnovu izraza:

$$P_a^S(\mathcal{F}) = P_{min}^{S}(\mathcal{F}) \cup P_{pod}^{S}(\mathcal{F}) \cup P_{ref}^{S}(\mathcal{F}),$$

gdje se $P_{min}^{S}(\mathcal{F})$ formira prema predikatu $Min(\mathrm{R_i})$, $P_{pod}^{S}(\mathcal{F})$ prema predikatu $Pod(\mathrm{R_i})$, a $P_{ref}^{S}(\mathcal{F})$ prema predikatu $Ref(\mathrm{R_i})$.

- **Z04** – tip forme je primjenljiv za upite. Podšema se formira na osnovu izraza:

$$P_u^S(\mathcal{F}) = P_{min}^{S}(\mathcal{F}) \cup P_{pod}^{S}(\mathcal{F}),$$

gdje se $P^S_{min}(\mathcal{F})$ formira prema predikatu $Min(\mathrm{R_i})$, a $P^S_{pod}(\mathcal{F})$ formira se prema predikatu $Pod(\mathrm{R_i})$.

- **Z05** – tip forme je primjenljiv za upite. Podšema se formira na osnovu izraza:

$$P^S_u(\mathcal{F}) = P^S_{min}'(\mathcal{F}) \cup P^S_{pod}'(\mathcal{F}),$$

gdje se $P^S_{min}'(\mathcal{F})$ formira prema predikatu $Min'(\mathrm{R_i})$, na osnovu skupa $P^S_{min}'(\mathcal{F})$ formira se skup $P^S_{pod}'(\mathcal{F})$.

5.1.5. Algoritmi za generisanje podšema tipova formi

Algoritam za generisanje podšeme tipa forme iz klase za ažuriranje

Algoritam za generisanje skupa šema relacija podšeme datog tipa forme iz klase $\mathrm{A}_{\mathcal{F}}$(a) sastoji se od 5 koraka. Algoritmi koji odgovaraju ovim koracima definisani su u [22], a dati su u prilogu A.

1. Generisanje skupa minimalnih čvorova podšeme $P^S_{min}(\mathcal{F})$ (slika A.1).
2. Generisanje skupa relevantnih čvorova $R_c(\mathcal{F})$ (Slika A.3).
3. Generisanje skupa potrebnih podređenih čvorova $P^S_{pod}(\mathcal{F})$ (slika A.8).
4. Analiza primjenljivosti tipa forme iz $\mathrm{A}_{\mathcal{F}}$(a) i generisanje skupa referentnih čvorova $P^S_{ref}(\mathcal{F})$ (slike A.9 i A.10).
5. Uklanjanje suvišnih (nerelevantnih) obilježja iz skupa P^S_u i formiranje skupa šema relacija podšeme $P_a(\mathcal{F})$ (slika A.11).

Algoritam za generisanje podšeme tipa forme iz klase za upite

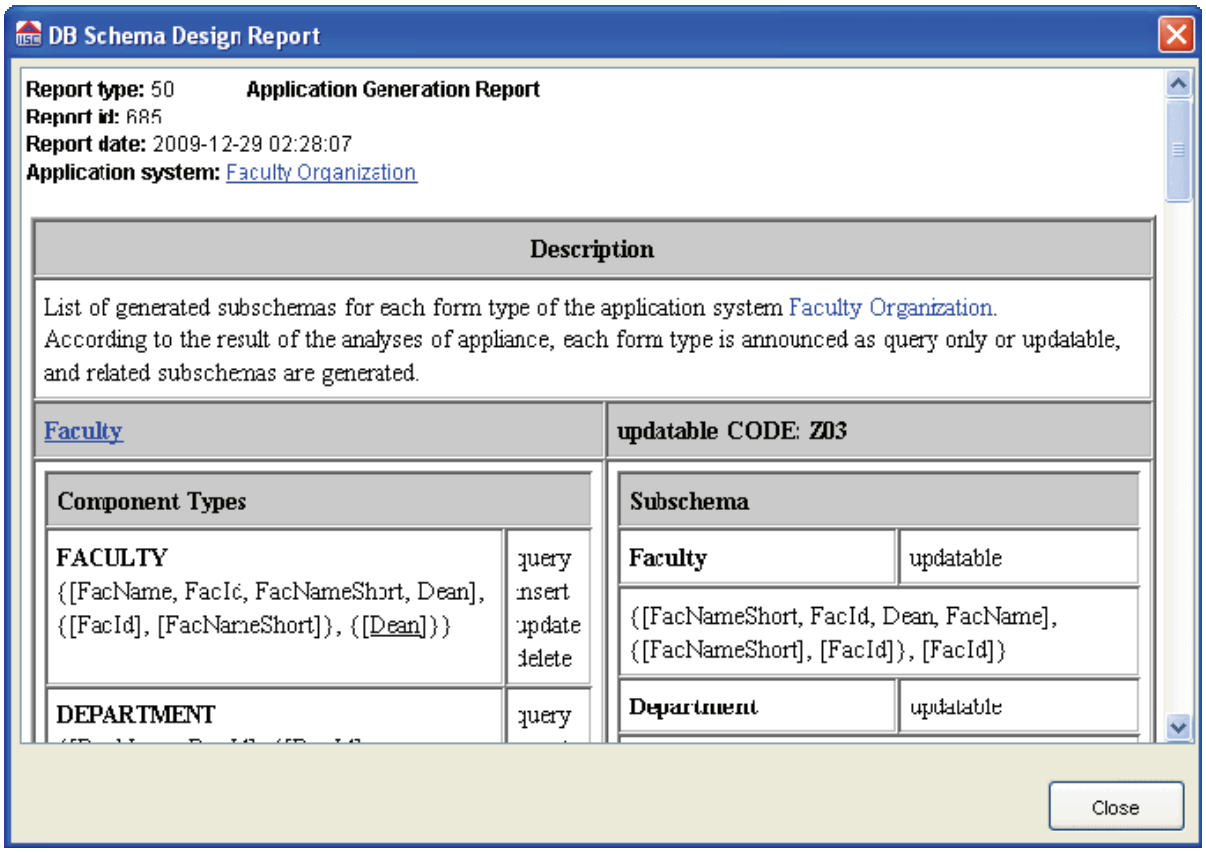
Algoritmi za generisanje podšeme tipa forme za iz klase upite definisani su u [22], a dati su u prilogu A. Svi algoritmi navedeni na slikama A.12 - A.22, sastavni su dio algoritma analize primjenljivosti i načina upotrebe tipa forme iz klase $\mathrm{A}_{\mathcal{F}}$(u) datog na slici A.23.

Generisanje podšeme tipa forme u IIS*Case-u

U prvom koraku automatizovanog postupka za generisanje transakcionih programa generišu se podšeme tipova formi. U okviru ovog koraka implementirani su navedeni algoritmi za generisanje podšema tipova formi iz klase za ažuriranje i upite. Kao rezultat, dobijaju se

skupovi šema relacija koje pripadaju podšemama pojedinih tipova formi, kao i rezultat analize primjenljivosti tipova formi u skladu sa zaključcima datim u tabeli 5.1.4.1.

Ukoliko želi, projektant može izabrati da se tokom ovog koraka generiše odgovarajući izvještaj prikazan na slici 5.1.5.1, koji daje informaciju o krajnjim rezultatima primjene ovih algoritama. Dati izvještaj sadrži spisak svih tipova formi sa informacijama o dozvoljenim operacijama na datom tipu forme: *query*, *insert*, *update*, i *delete*. Dozvoljene operacije na tipovima komponenti tipa forme projektant zadaje putem forme za specifikaciju tipova komponenti. S druge strane, tipovima formi pridruženi su odgovarajući indikatori (prema tabeli 5.1.4.1) koji su rezultat analize primjenljivosti i načina njihove upotrebe. U slučaju da su vrijednosti indikatora analize primjenljivosti i načina upotrebe tipova formi Z01-Z05, za svaki od tipova formi generisana je i prikazana podšema. Vrijednost indikatora implicira način upotrebe tipa forme, tako da samo vrijednost Z03 dozvoljava upotrebu datog tipa forme za čitanje i ažuriranje podataka, dok ostale vrijednosti, u slučaju da je podšema generisana, dozvoljavaju samo čitanje podataka. Uz svaku od šema relacija generisanih podšema prikazana je vrijednost: *updatable* ili *read only*, u zavisnosti od pomenutog indikatora. Prilikom generisanja prototipa aplikacija, a u zavisnosti od dozvoljenih operacija na samim tipovima komponenti, kao i rezultatima analize primjenljivosti i načina upotrebe tipova formi, generišu se ekranske forme koje će podržavati odgovarajuće operacije za čitanje i ažuriranje podataka.



Slika 5.1.5.1. *Izvještaj o generisanim podšemama tipova formi*

5.2. Specifikacija vizuelnih osobina transakcionog programa

Prilikom generisanja transakcionog programa generiše se odgovarajuća UIML specifikacija koja uključuje sve prethodno definisane vizuelne osobine aplikativnog sistema. Ove osobine projektant zadaje kroz:

- izbor šablona korisničkog interfejsa,
- specificiranje vizuelnih atributa, grupa polja i lista vrijednosti i
- izbor poslovne aplikacije.

5.2.1. Šablon korisničkog interfejsa

Specifikacija šablona korisničkog interfejsa i njegova UIML interpretacija detaljno su izloženi u poglavlju 4.

5.2.2. Vizuelni atributi, grupe polja i liste vrijednosti

Vizuelna svojstva generisanog transakcionog programa u smislu prikaza polja za pregled i ažuriranje podataka određena su ne samo izabranim korisničkim šablonom već i specifikacijama vizuelnih atributima, grupa polja i lista vrijednosti.

Vizuelni atributi

Vizuelni atributi uvedeni su u [47] i direktno se zadaju kroz interfejs alata IIS*Case kroz specifikacije:

- domena,
- atributa,
- atributa na tipovima komponenti i
- tipova komponenti.

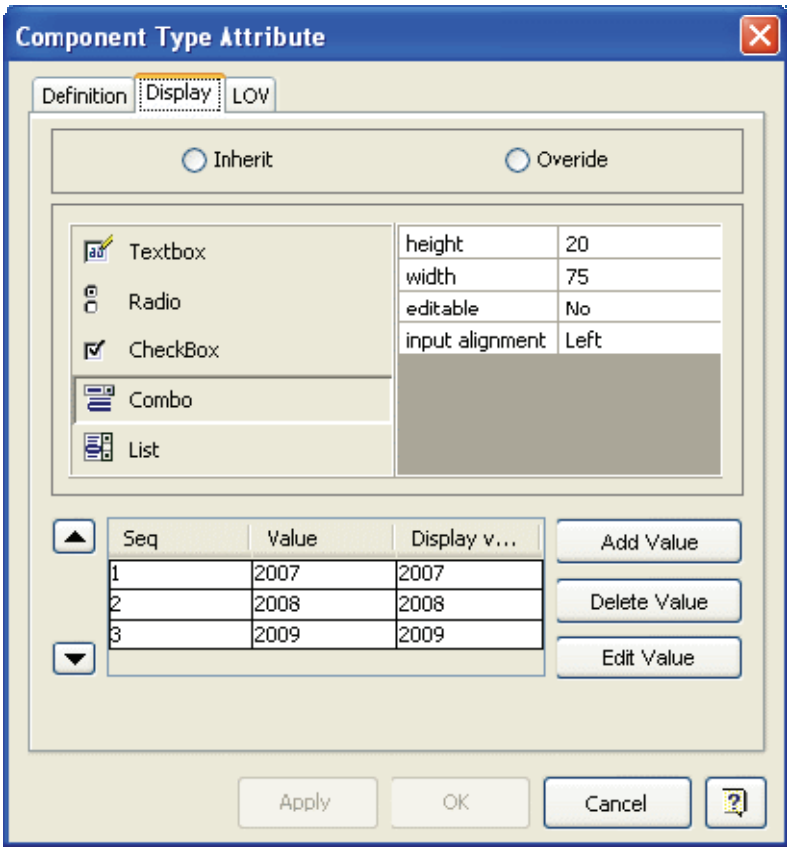
Atribut nasljeđuje vizuelne atribute pridruženog domena. Za neki atribut moguće je definisati drugačija vizuelna svojstva, nezavisno od odgovarajućeg domena. Takođe, atribut na tipu komponente nasljeđuje specifikaciju vizuelnih osobina pridruženog atributa na nivou osnovnih koncepata. Na analogan način i vrijednosti nasljeđenih vizuelnih osobina atributa je moguće dodatno redefinisati na atributu tipa komponente.

Slika 5.2.2.1 prikazuje formu putem koje se definišu vizuelni atributi atributa na tipu komponente. Za svaki domen može se definisati tip polja (programske kontrole), kojim će dati atribut biti prikazan na ekranskoj formi. Moguće vrste polja su:

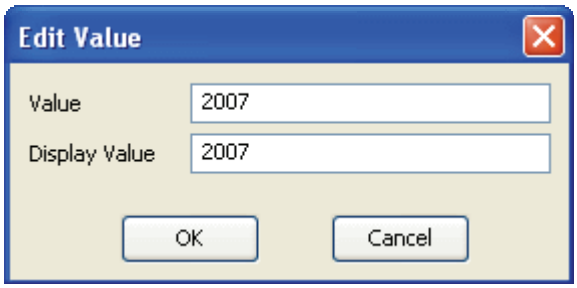
- tekstualno polje (*Text box*),
- radio grupa (*Radio button*),
- polje za čekiranje (*Check box*),

- padajuća lista (*Combo box*), i
- lista izbora (*List*).

Za programske kontrole: radio grupa, polje za čekiranje, padajuća lista i lista izbora; može se definisati skup ponuđenih vrijednosti koje se putem te kontrole mogu unositi kao što je prikazano na slici 5.2.2.2.

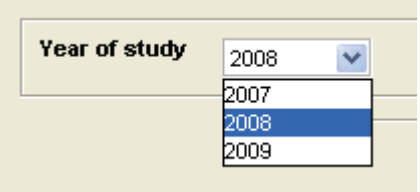


Slika 5.2.2.1. *Forma za specificiranje vizuelnih atributa na tipu komponente*



Slika 5.2.2.2. *Forma za specificiranje ponuđenih vrijednosti*

Na slici 5.2.2.3 prikazano je kako izgleda atribut sa vizuelnim atributima prikazanim na slici 5.2.2.1, na ekranskoj formi generisanog transakcionog programa. Na slici 5.2.2.4 dat je fragment UIML specifikacije koji se odnosi na ovaj atribut tipa komponente.



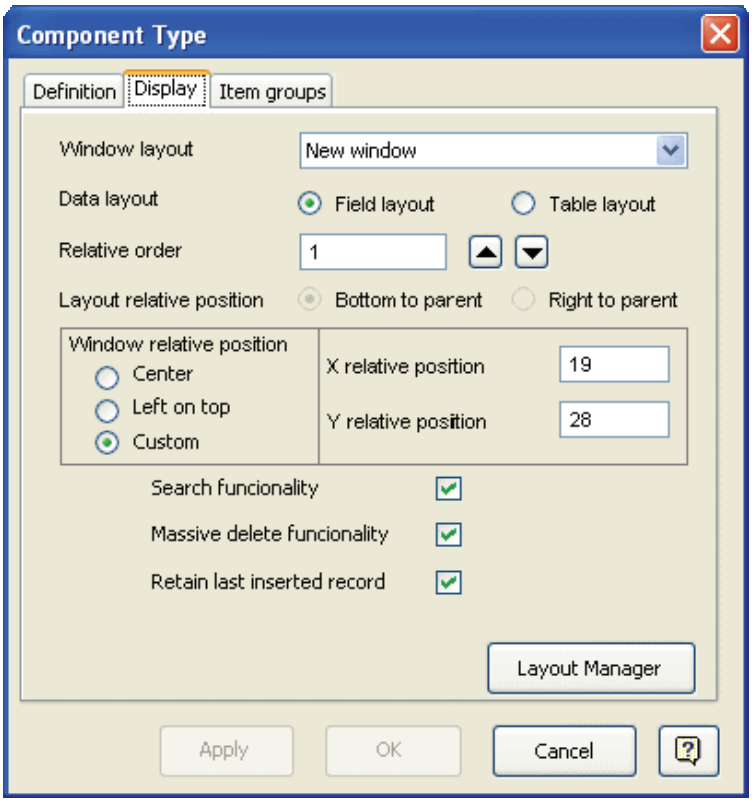
Slika 5.2.2.3. *Prototip korisničkog interfejsa – atribut sa vizuelnim atributima sa slike* 5.2.2.1.

```
<part id="629itemGroup" class="JPanel">
<style>
<property name="bounds">10,10,500,40</property>
<property name="preferredSize">500,40</property>
<property name="border">EtchedBorder</property>
<property name="name">Item Group 6</property>
<property name="background">ece9d8</property>
<property  name="layout">null</property>
</style>
<part id="6029InputLabel" class="JLabel">
<style>
<property name="bounds">10,10,91,11</property>
<property name="horizontalAlignment">LEFT</property>
<property name="text">Year of study</property>
<property name="font">Arial Bold-bold-11</property>
</style>
</part>
<part id="6029InputField" class="JComboBox">
<style>
<property name="content"><constant model="list">
<constant value="2007"/>
<constant value="2008"/>
<constant value="2009"/>
</constant>
</property>
<property name="bounds">101,10,75,20</property>
<property name="editable">false</property>
</style>
</part>
</part>
```

Slika 5.2.2.4. *UIML specifikacija atributa sa vizuelnim atributima sa slike* 5.2.2.1.

Forme za specificiranje vizuelnih osobina atributa i domena imaju gotovo identične funkcionalnosti kao forma prikazana na slici 5.2.2.1.

Na 5.2.2.5 data je forma za specificiranje vizuelnih atributa tipova komponenti putem koje je moguće zadati: položaj tipa komponente u generisanoj formi, način prikaza podataka na ekranskoj formi, poredak tipa komponente na ekranskoj formi u odnosu na druge tipove komponenti i njegov položaj, funkcionalnosti masovnog brisanja, pretrage, itd.



Slika 5.2.2.5. *Forma za specificiranje vizuelnih atributa tipova komponenti*

Na slici 5.2.2.6 dat je fragment UIML specifikacije koji se odnosi na tip komponente *Faculty* sa slike 3.1.1.3, čiji su vizuelni atributi specificirani na slici 5.2.2.5.

```
- <part id="25compType" class="JInternalFrame">
+ <style>
+ <part id="25compTypeMenu" class="JMenuBar">
+ <part id="25Parameters" class="JPanel">
+ <part id="25Title" class="JLabel">
+ <part id="25Toolbar" class="JPanel">
+ <part id="25masterDetail" class="JPanel">
- <part id="25DataPanel" class="DataPanel">
+ <style>
- <part id="2525itemGroup" class="JPanel">
+ <style>
- <part id="4525InputLabel" class="JLabel">
- <style>
  <property name="bounds">10,10,126,11</property>
  <property name="horizontalAlignment">LEFT</property>
  <property name="text">Faculty name</property>
  <property name="font">Arial Bold-bold-11</property>
  </style>
  </part>
- <part id="4525InputField" class="JTextField">
- <style>
  <property name="text">Faculty name 1</property>
  <property name="bounds">136,10,150,15</property>
  </style>
  </part>
- <part id="4425InputLabel" class="JLabel">
- <style>
  <property name="bounds">10,35,126,11</property>
```

```
  <property name="horizontalAlignment">LEFT</property>
  <property name="text">Faculty ID</property>
  <property name="font">Arial Bold-bold-11</property>
  </style>
  </part>
- <part id="4425InputField" class="JTextField">
- <style>
  <property name="text" />
  <property name="bounds">136,35,150,15</property>
  </style>
  </part>
- <part id="4625InputLabel" class="JLabel">
- <style>
  <property name="bounds">10,60,126,11</property>
  <property name="horizontalAlignment">LEFT</property>
  <property name="text">Faculty short name</property>
  <property name="font">Arial Bold-bold-11</property>
  </style>
  </part>
- <part id="4625InputField" class="JTextField">
- <style>
  <property name="text">default text</property>
  <property name="bounds">136,60,150,15</property>
  </style>
  </part>
- <part id="5825InputLabel" class="JLabel">
- <style>
  <property name="bounds">10,85,126,11</property>
  <property name="horizontalAlignment">LEFT</property>
  <property name="text">Dean</property>
  <property name="font">Arial Bold-bold-11</property>
  </style>
  </part>
- <part id="5825InputField" class="JTextField">
- <style>
  <property name="text">default text</property>
  <property name="bounds">136,85,150,15</property>
  </style>
  </part>
  </part>
  </part>
+ <part id="25NavigationPanel" class="JPanel">
+ <part id="25ButtonPanel" class="JPanel">
  </part>
```

Slika 5.2.2.6. *UIML specifikacija tipa komponente sa vizuelnim atributima sa slike* 5.2.2.5.

Specifikacija dijela repozitorijuma koji se odnosi na vizuelne atribute data je u prilogu B (slika B.2).

Grupe polja

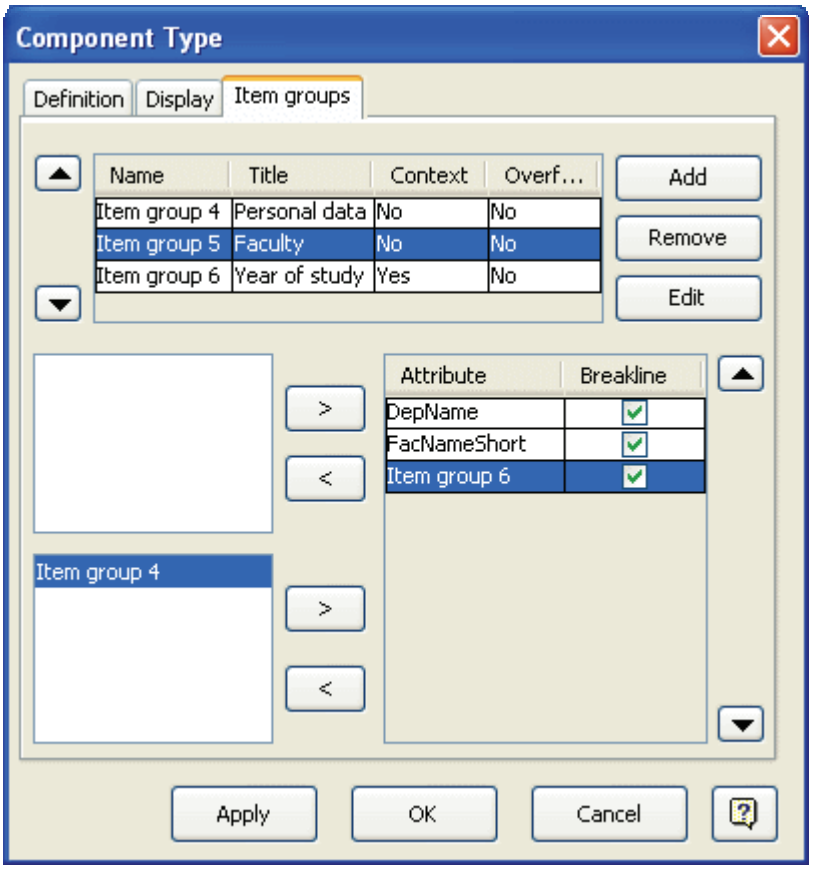
Pojam grupe polja uveden je u [47] i definiše se na nivou koncepta tipa komponente. Motivacija za uvođenje ovog pojma je poboljšanje preglednosti ekranskih formi. Polja srodne funkcionalnosti ili semantike se na ovaj način vizuelno izdvajaju i grupišu u panele, a uvodi se i redosled njihovog prikazivanja. Jednu grupu polja čini skup atributa na istom tipu komponente. Svaki tip komponente mora da ima definisanu bar jednu grupu polja. Svi atributi na tipovima komponenti moraju biti sadržani u tačno jednoj grupi polja, tj. jedan atribut na

tipu komponente mora da pripada jednoj grupi polja. Istoj grupi polja ne mogu pripadati dva atributa sa različitih tipova komponenti. U jednu grupu polja može biti uključena druga grupa polja kao podgrupa polja i kao takva ne može sadržati druge grupe polja.

Na slici 5.2.2.7 prikazana je forma za zadavanje grupa polja na tipu komponente. Pomoću ove forme projektant može da dodaje nove, briše i modifikuje postojeće grupe polja.

Osim naziva i naslova, grupi polja zadaju se njene podgrupe, kao i indikator da li je u pitanju grupa kontekstnog ili *overflow* tipa. Čekiranjem opcije *Overflow* označava da će ta grupa polja na ekranskoj formi biti smještena u tzv. *overflow* oblast – oblast koja prikazuje dodatne podatke, tekućeg, trenutno prikazanog sloga (torke) na formi u slučaju tabelarnog prikaza podataka. Čekiranjem opcije *Context*, definiše se da li će ta grupa polja, u slučaju prikaza podataka u okviru panela biti prikazana u posebnoj kontekstnoj tabeli.

Za svaki atribut u okviru neke grupe polja, definisan je njegov poredak i relativni položaj u odnosu na prethodni atribut u poretku (opcijom *Breakline* definiše se da li će atribut biti prikazan desno od prethodnog atributa, ili ispod tj. u novom redu).

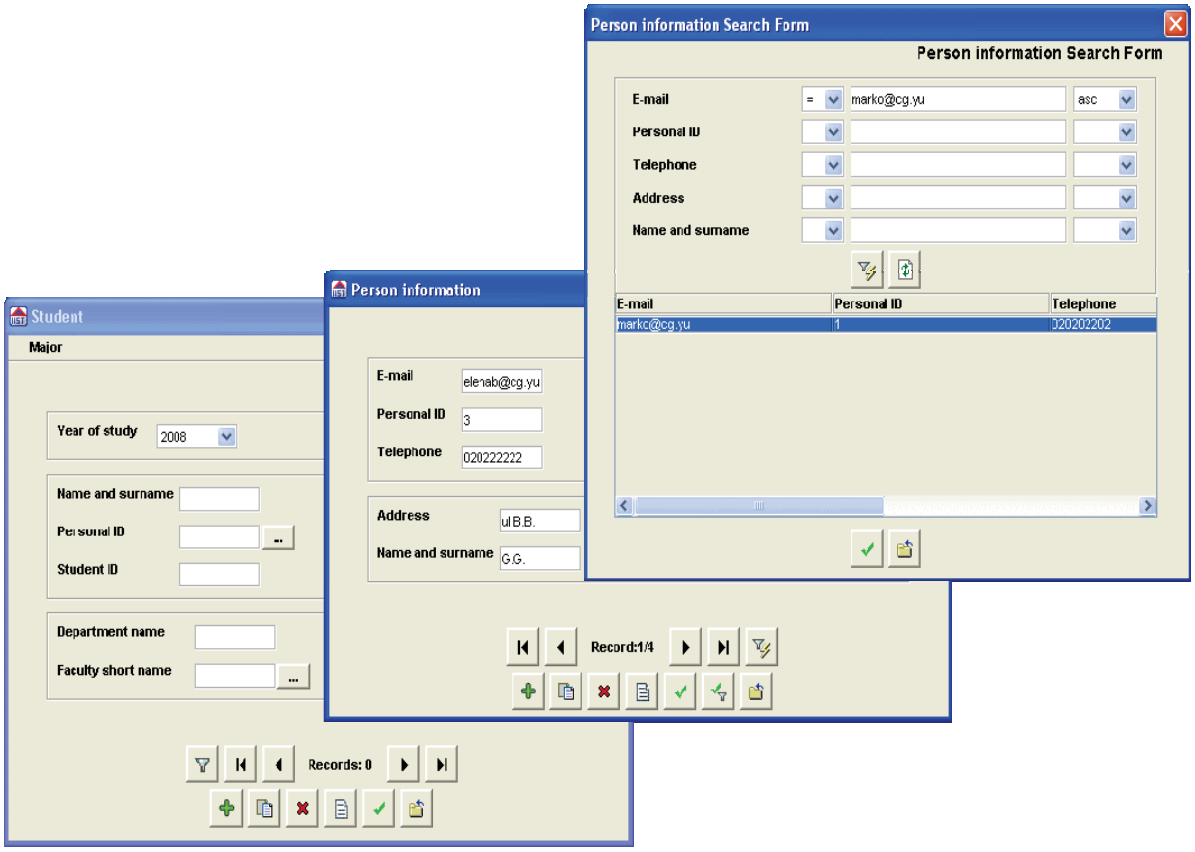


Slika 5.2.2.7. *Forma za specificiranje grupa polja tipa komponente*

Specifikacija dijela repozitorijuma koji se odnosi na grupe polja data je u prilogu B (slika B.2).

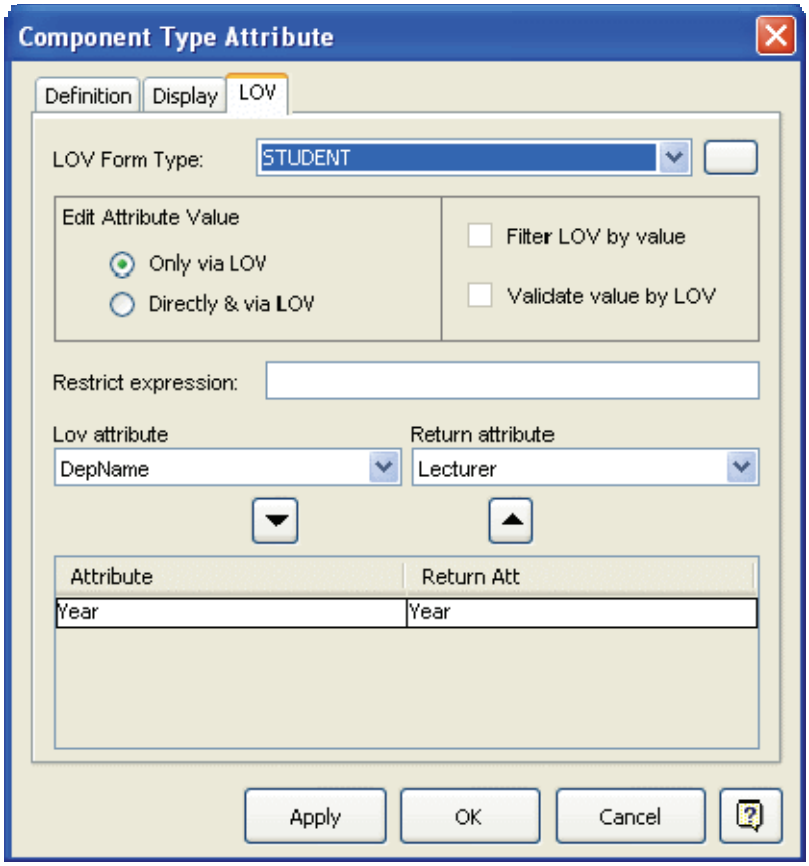
Liste vrijednosti

Pojam liste vrijednosti uveden je kako bi se unaprijedila funkcionalnost unosa podataka generisanog transakcionog programa. Specifikacija liste vrijednosti zadaje se na nivou atributa tipa komponente. Zadavanjem liste vrijednosti za određeni atribut na tipu komponente, omogućeno je da krajnji korisnik, u okviru transakcionog programa, zadaje vrijednost za dati atribut putem posebne ekranske forme koja predstavlja listu vrijednosti. Ova ekranska forma odgovara nekoj od definisanih tipova formi, i njena struktura, dozvoljene funkcionalnosti, vizuelni i drugi atributi definisani su specifikacijom datog tipa forme. Kao što je prikazano na slici 5.2.2.8, listu vrijednosti *Person Information* korisnik aktivira klikom na posebno dugme pridruženo polju koje odgovara atributu *Personal Id* i putem nje je u stanju da izvrši pretragu vrijednost zadajući više parametara pretrage. Nakon toga, izabrana vrijednost atributa vraća se natrag početnoj formi, i odgovarajuće polje dobija izabranu vrijednost. Moguće je putem iste liste vrijednosti vratiti vrijednost više atributa na tipu komponente.



Slika 5.2.2.8. *Primjer liste vrijednosti*

Na slici 5.2.2.9 prikazana je forma za zadavanje liste vrijednosti okviru specifikacije atributa na tipu komponente. Putem ove forme zadaje se tip forme liste vrijednosti, lista atributa kojima se vraćaju izabrane vrijednosti, lista uparenih atributa sa zadatog tipa forme, editabilnost polja, način filtriranja podataka dostupnih kroz listu vrijednosti, kao i validiranje ručno unešene vrijednosti. Elementi specifikacije liste vrijednosti detaljno su specificirani u tabeli 5.2.2.1.



Slika 5.2.2.9. *Forma za specificiranje liste vrijednosti*

Atributi liste vrijednosti (*List of Value - LOV*)	
Tip forme liste vrijednosti (*LOV Form Type*)	Tip forme koji specificira ekransku formu liste vrijednosti izabira se iz ponuđene liste.
Editabilnost polja (*Edit Attribute Value*)	Mogući načini zadavanja vrijednosti atributa kojem je pridružena lista vrijednosti. Moguće vrijednosti su: • putem liste vrijednosti (*Only via LOV*), • direktnim unosom i putem liste vrijednosti (*Directly & Only via LOV*).
Filtriranje podataka dostupnih kroz listu vrijednosti (*Filter LOV by value*)	Čekiranjem ove opcije, u slučaju da je dozvoljeno zadavanje vrijednosti atributa direktno ukucavanjem vrijednosti u odgovarajuće polje, definiše se da se unesena vrijednost koristi kao filter prilikom izlistavanja rekorda u listi vrijednosti.
Validiranje listom vrijednosti (*Validate value by LOV*)	Čekiranjem ove opcije, u slučaju da je dozvoljeno zadavanje vrijednosti atributa direktno ukucavanjem vrijednosti u odgovarajuće polje, definiše se da se unesena vrijednost validira pri napuštanju polja. U slučaju da vrijednost nije validna ispisuje se odgovarajuća poruka i nemoguće je napustiti polje dok se ne unese validna vrijednost.
Restriktivni izraz (*Restriction expression*)	U listi vrijednosti će biti prikazane samo one vrijednosti koje zadovoljavaju dati restriktivni izraz.

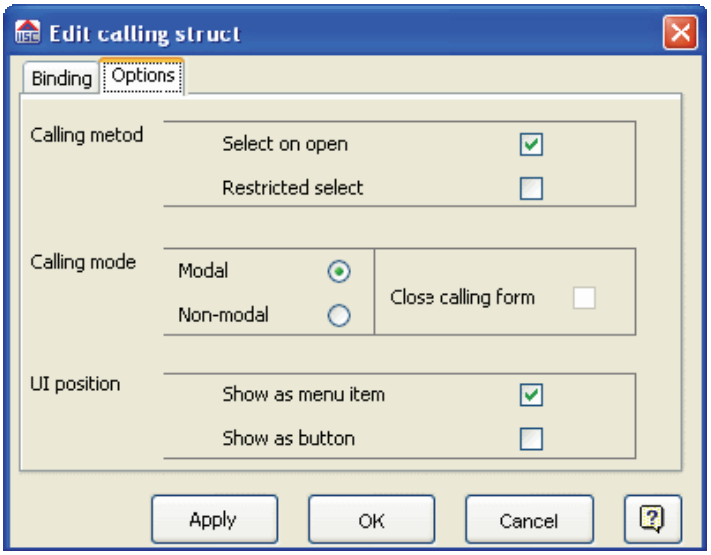
Tabela uparenih atributa	Specifikacijom ove tabele definiše se lista atributa kojima se vraća vrijednost putem liste vrijednosti (kolona *Return Att*) i lista odgovarajućih atributa sa forme liste vrijednosti (kolona *Attribute*).

Tabela 5.2.2.1. *Specifikacija atributa liste vrijednosti*

Specifikacija dijela repozitorijuma koji se odnosi na liste vrijednosti data je u prilogu B (slika B.2).

5.2.3. Poslovne aplikacije

U skladu sa specifikacijom poslovne aplikacije date na slici 3.1.3.2 generiše se ekranska forma data na slici 3.1.3.3. Na slici 5.2.3.1. data je forma IIS*Case-a za specificiranje pozivajućih struktura putem koje se definiše način poziva drugih ekranskih formi sa date ekranske forme u okviru specifikacije poslovne aplikacije. Pojam poslovnih aplikacija i pozivajućih struktura, kao i njihova specifikacija uvedeni su u [47]. Generator aplikacija, prilikom generisanja prototipa integriše elemente ovih specifikacija u okviru generisane UIML specifikacije.



Slika 5.2.3.1. *Forma za specificiranje pozivajućih struktura*

Na slici 5.2.3.2 dat je fragment generisane UIML specifikacije koji se odnosi na strukturu menija ekranske forme koji u sebi uključuje funkcionalnost poziva drugih ekranskih formi onako kako je to zadato kroz specifikaciju odgovarajuće pozivajuće strukture.

```
- <part id="25compTypeMenu" class="JMenuBar">
- <part id="47compTypeMenu" class="JMenuItem">
- <style>
  <property name="text">Department</property>
  <property name="minimumSize">70,10</property>
  <property name="maximumSize">80,10</property>
  <property name="mnemonic">68</property>
  </style>
```

```
- <behavior>
- <rule>
- <condition>
  <event class="actionPerformed" />
  </condition>
- <action>
  <property part-name="47compType" name="visible">true</property>
  <property part-name="47LoVButton" name="visible">false</property>
  <property part-name="25DataPanel" name="caller">-1</property>
  </action>
  </rule>
  </behavior>
  </part>
- <part id="30compTypeMenu" class="JMenuItem">
- <style>
  <property name="text">Student grades</property>
  <property name="background">ece9d8</property>
  <property name="foreground">000000</property>
  <property name="font">Arial Bold-bold-11</property>
  <property name="minimumSize">94,10</property>
  <property name="minimumSize">104,10</property>
  <property name="mnemonic">83</property>
  </style>
- <behavior>
- <rule>
- <condition>
  <event class="actionPerformed" />
  </condition>
- <action>
  <property part-name="50compType" name="visible">true</property>
  <property part-name="50DataPanel" name="caller">12</property>
  <property part-name="50LoVButton" name="visible">false</property>
  <property part-name="25compType" name="visible">true</property>
  </action>
  </rule>
  </behavior>
  </part>
- <part id="15compTypeMenu" class="JMenuItem">
- <style>
  <property name="text">Courses and exams</property>
  <property name="background">ece9d8</property>
  <property name="foreground">000000</property>
  <property name="font">Arial Bold-bold-11</property>
  <property name="minimumSize">112,10</property>
  <property name="minimumSize">122,10</property>
  <property name="mnemonic">67</property>
  </style>
- <behavior>
- <rule>
- <condition>
  <event class="actionPerformed" />
  </condition>
- <action>
  <property part-name="34compType" name="visible">true</property>
  <property part-name="34DataPanel" name="caller">12</property>
  <property part-name="34LoVButton" name="visible">false</property>
  <property part-name="25compType" name="visible">true</property>
  </action>
  </rule>
  </behavior>
  </part>
```

```
- <part id="13compTypeMenu" class="JMenuItem">
- <style>
  <property name="text">Person information</property>
  <property name="background">ece9d8</property>
  <property name="foreground">000000</property>
  <property name="font">Arial Bold-bold-11</property>
  <property name="minimumSize">118,10</property>
  <property name="minimumSize">128,10</property>
  <property name="mnemonic">80</property>
  </style>
- <behavior>
- <rule>
- <condition>
  <event class="actionPerformed" />
  </condition>
- <action>
  <property part-name="28compType" name="visible">true</property>
  <property part-name="28DataPanel" name="caller">12</property>
  <property part-name="28LoVButton" name="visible">false</property>
  <property part-name="25compType" name="visible">true</property>
  </action>
  </rule>
  </behavior>
  </part>
  </part>
```

Slika 5.2.3.2. *UIML specifikacija strukture menija forme Faculty Organization*

Analogno, na slici 5.2.3.3 dat je fragment generisane UIML specifikacije koji se odnosi na strukturu *toolbar*-a ekranske forme koji u sebi uključuje funkcionalnost poziva drugih ekranskih formi klikom na dugme.

```
- <part id="25Toolbar" class="JPanel">
- <style>
  <property name="opaque">false</property>
  <property name="bounds">20,29,83,40</property>
  </style>
- <part id="12ToolbarButton" class="JButton">
- <style>
  <property name="text">Student</property>
  <property name="preferredSize">75,30</property>
  </style>
- <behavior>
- <rule>
- <condition>
  <event class="actionPerformed" />
  </condition>
- <action>
  <property part-name="29compType" name="visible">true</property>
  <property part-name="29DataPanel" name="caller">12</property>
  <property part-name="29LoVButton" name="visible">false</property>
  <property part-name="25compType" name="visible">false</property>
  </action>
  </rule>
  </behavior>
  </part>
  </part>
```

Slika 5.2.3.3. *UIML specifikacija toolbar-a ekranske forme Faculty Organization*

5.3. Generisanje izvršnih softverskih specifikacija

U [17, 18] predloženo je generisanje izvršnih softverskih specifikacija informacionih sistema putem generisanja UIML specifikacija koje se mogu interpretirati korišćenjem *Java Render*-a. Ovaj postupak interpretira samo vizuelne karakteristike korisničkog interfejsa, ne i složene funkcionalnosti kao što su npr. funkcionalnosti koje podržavaju komunikaciju sa bazom podataka. U [17, 18] pokazano je kako je ovaj nedostatak *Java Render*-a moguće nadomjestiti implementacijom *Java* klasa koje podržavaju složene funkcionalnosti generisanih prototipova.

Predloženi postupak je dalje unaprijeđen i implementiran u okviru generatora aplikacija alata IIS*Case, što je jedan od glavnih rezultata istraživanja sprovedenog u ovoj doktorskoj disertaciji. Generisani prototipovi aplikacija su funkcionalni programi.

UIML specifikacije prototipova aplikacija sadrže sve što je potrebno da bi se elementi specifikacije korisničkog interfejsa i funkcionalnosti projektovanog aplikativnog sistema na pravilan način interpretirali, odnosno transformisali u odgovarajuće programske komponente izabranog programskom okruženja. Definisanje preslikavanja elemenata projektantske specifikacije u programske komponente obuhvata:

- specificiranje programskih komponenti transakcionog programa i njihovih metoda koje implementiraju korisnički interfejs i
- specificiranje programskih komponenti transakcionog programa i njihovih metoda koje implementiraju osobine i ponašanja svih komponenti korisničkog interfejsa.

5.3.1. Preslikavanje UIML elemenata u programske komponente

Tokom interpretacije, elementi UIML dokumenta preslikavaju se u odgovarajuće programske komponente:

- *Java AWT/Swing komponente* koje služe za interpretaciju svih elemenata korisničkog interfejsa i osnovnih funkcionalnosti koje su pokrivene metodama *Java AWT/Swing* klasa.
- *Custom Java komponente*, odnosno objekte, za ove potrebe, kreiranih *Java* klasa koje implementiraju napredne funkcionalnosti kao što su npr. one koje podržavaju komunikaciju sa bazom podataka, prenos parametara i njihovih vrijednosti između ekranskih formi, itd.

U okviru UIML specifikacije veza između odgovarajućih UIML komponenti i odgovarajućih Java klasa ostvarena je kroz strukturu `<peers>`. Ova struktura omogućava definisanje preslikavanje klasa i njihovih metoda, osobina i događaja u eksterne objekte koji nisu sastavni dio UIML specifikacije. U okviru njene podstrukture `<presentation>` definisan je rečnik dozvoljenih imena klasa (specificiranih kroz UIML elemenat `<d-class>`) koje mogu biti korišćene za specificiranje elemenata, osobina i događaja definisanih u UIML dokumentu. Za potrebe implementacije generatora aplikacije kreiran je poseban rečnik koji proširuje osnovni rečnik *Java 1.5 Harmonia 1.0*. kao što je prikazano na slici 5.3.1.

Za potrebe implementacije složenih funkcionalnosti koje podržavaju komunikaciju sa bazom podataka razvijene su, u okviru ovog istraživanja, *Java* klase `ui.DBTable` i `ui.DBPanel`. Kroz specifikaciju datu na slici 5.3.1, UIML elementi klasa `DataTable` i `DataPanel` transformišu se u objekte ovih klasa. Specifikacija ove transformacija definisana je u okviru UIML elemenata `<d-class>`. Putem ovih elementa zadaje se lista dozvoljenih imena UIML klasa. Dodatno se definišu i klase izabranog programskog okruženja (u ovom slučaju *Java* klase) u čije objekte se, specificiranjem opcije `maps-to`, odgovarajuće UIML komponente preslikavaju. Specifikacijom podelementa `<d-property>` i `<d-param>` ostvaruje se veza između metoda preslikanih *Java* klasa i odgovarajućih UIML klasa, tako da je omogućeno da se pozivom datih *Java* metoda mijenjaju osobine objekata datih UIML klasa.

UIML elementi `DataTable` i `DataPanel` su najvažniji strukturni elementi UIML specifikacije na nivou tipa komponente. Prilikom generisanja UIML specifikacije, u zavisnosti od specifikacije vizuelnih atributa tipa komponenti (tačnije specifikacije načina prikaza podataka – tabelarni ili putem panela), u okviru UIML fragmenta koji se odnosi na tip komponente generiše se UIML element klase `DataTable` ili `DataPanel`. U toku izvršavanja, ovi elementi se interpretiraju kao odgovarajuće programske komponente, tj. objekti klase `ui.DBTable` ili `ui.DBPanel`.

```
- <peers>
- <logic>
- <d-component id="MainFrame" name="MainFrame" maps-to="ui.UIFrame">
- <d-method id="Close" maps-to="closeFrame">
  </d-method>
  </d-component>
  </logic>
- <presentation base="Java_1.5_Harmonia_1.0" how="union" export="optional">
- <d-class id="DataTable" used-in-tag="part" maps-type="class" maps-
to="ui.DBTable" how="replace" export="optional">
- <d-property id="preferredScrollableViewportSize" maps-type="setMethod"
maps-to="setPreferredScrollableViewportSize">
  <d-param type="java.awt.Dimension" />
  </d-property>
- <d-property id="visible" maps-type="setMethod" maps-to="setVisible">
  <d-param type="boolean" />
  </d-property>
- <d-property id="LoV" maps-type="setMethod" maps-to="setL0Vs">
  <d-param type="int" />
  </d-property>
- <d-property id="caller" maps-type="setMethod" maps-to="setCaller">
  <d-param type="int" />
  </d-property>
- <d-property id="setSearch" maps-type="setMethod" maps-to="setSearchData">
  <d-param type="int" />
  </d-property>
- <d-property id="setSearchChoice" maps-type="setMethod" maps-
to="setSearchChoiceData">
  <d-param type="int" />
  </d-property>
- <d-property id="reset" maps-type="setMethod" maps-to="resetData">
  <d-param type="int" />
  </d-property>
- <d-property id="setAction" maps-type="setMethod" maps-to="setAction">
  <d-param type="java.lang.String" />
  </d-property>
- <d-property id="url" maps-type="setMethod" maps-to="setUrl">
```

```
  <d-param type="java.lang.String" />
  </d-property>
- <d-property id="comp" maps-type="setMethod" maps-to="setComp">
  <d-param type="int" />
  </d-property>
- <d-property id="pr" maps-type="setMethod" maps-to="setPR">
  <d-param type="int" />
  </d-property>
- <d-property id="as" maps-type="setMethod" maps-to="setAS">
  <d-param type="int" />
  </d-property>
- <d-property id="username" maps-type="setMethod" maps-to="setUsername">
  <d-param type="java.lang.String" />
  </d-property>
- <d-property id="password" maps-type="setMethod" maps-to="setPassword">
  <d-param type="java.lang.String" />
  </d-property>
- <d-property id="border" maps-type="setMethod" maps-to="setBorder" post-
child="false" export="optional">
  <d-param type="javax.swing.border.Border" />
  </d-property>
- <d-property return-type="" id="selectionBackground" maps-type="setMethod"
maps-to="setSelectionBackground">
  <d-param id="" type="java.awt.Color" />
  </d-property>
- <d-property return-type="" id="selectionForeground" maps-type="setMethod"
maps-to="setSelectionForeground">
  <d-param id="" type="java.awt.Color" />
  </d-property>
- <d-property id="rowHeight" maps-type="setMethod" maps-to="setRowHeight">
  <d-param type="int" />
  </d-property>
- <d-property id="intercellSpacing" maps-type="setMethod" maps-
to="setIntercellSpacing">
  <d-param type="java.awt.Dimension" />
  </d-property>
- <d-property id="font" maps-type="setMethod" maps-to="setFont">
  <d-param type="java.awt.Font" />
  </d-property>
- <d-property return-type="" id="background" maps-type="setMethod" maps-
to="setBackground">
  <d-param id="" type="java.awt.Color" />
  </d-property>
- <d-property return-type="" id="foreground" maps-type="setMethod" maps-
to="setForeground">
  <d-param id="" type="java.awt.Color" />
  </d-property>
- <d-property return-type="" id="gridColor" maps-type="setMethod" maps-
to="setGridColor">
  <d-param id="" type="java.awt.Color" />
  </d-property>
- <d-property id="autoCreateColumnsFromModel" maps-type="setMethod" maps-
to="setAutoCreateColumnsFromModel">
  <d-param type="boolean" />
  </d-property>
- <d-property id="autoResizeMode" maps-type="setMethod" maps-
to="setAutoResizeMode">
  <d-param type="int" />
  </d-property>
- <d-property id="selectionMode" maps-type="setMethod" maps-
to="setSelectionMode">
```

```
  <d-param type="int" />
  </d-property>
  </d-class>
- <d-class id="DataPanel" used-in-tag="part" maps-type="class" maps-
to="ui.DBPanel" how="replace" export="optional">
  <d-property id="addToContainer" maps-type="method" maps-to="add" />
- <d-property id="visible" maps-type="setMethod" maps-to="setVisible">
  <d-param type="boolean" />
  </d-property>
- <d-property id="opaque" maps-type="setMethod" maps-to="setOpaque">
  <d-param type="boolean" />
  </d-property>
- <d-property id="LoV" maps-type="setMethod" maps-to="setL0Vs">
  <d-param type="int" />
  </d-property>
- <d-property id="caller" maps-type="setMethod" maps-to="setCaller">
  <d-param type="int" />
  </d-property>
- <d-property id="setSearch" maps-type="setMethod" maps-to="setSearchData">
  <d-param type="int" />
  </d-property>
- <d-property id="setSearchChoice" maps-type="setMethod" maps-
to="setSearchChoiceData">
  <d-param type="int" />
  </d-property>
- <d-property id="reset" maps-type="setMethod" maps-to="resetData">
  <d-param type="int" />
  </d-property>
- <d-property id="setAction" maps-type="setMethod" maps-to="setAction">
  <d-param type="java.lang.String" />
  </d-property>
- <d-property id="url" maps-type="setMethod" maps-to="setUrl">
  <d-param type="java.lang.String" />
  </d-property>
- <d-property id="comp" maps-type="setMethod" maps-to="setComp">
  <d-param type="int" />
  </d-property>
- <d-property id="pr" maps-type="setMethod" maps-to="setPR">
  <d-param type="int" />
  </d-property>
- <d-property id="as" maps-type="setMethod" maps-to="setAS">
  <d-param type="int" />
  </d-property>
- <d-property id="username" maps-type="setMethod" maps-to="setUsername">
  <d-param type="java.lang.String" />
  </d-property>
- <d-property id="password" maps-type="setMethod" maps-to="setPassword">
  <d-param type="java.lang.String" />
  </d-property>
- <d-property return-type="" id="background" maps-type="setMethod" maps-
to="setBackground">
  <d-param id="" type="java.awt.Color" />
  </d-property>
- <d-property return-type="" id="foreground" maps-type="setMethod" maps-
to="setForeground">
  <d-param id="" type="java.awt.Color" />
  </d-property>
- <d-property return-type="" id="font" maps-type="setMethod" maps-
to="setFont">
  <d-param id="" type="java.awt.Font" />
```

```
  </d-property>
- <d-property return-type="" id="border" maps-type="setMethod" maps-
to="setBorder">
  <d-param id="" type="javax.swing.border.Border" />
  </d-property>
- <d-property id="preferredSize" maps-type="setMethod" maps-
to="setPreferredSize">
  <d-param type="java.awt.Dimension" />
  </d-property>
- <d-property id="bounds" maps-type="setMethod" maps-to="setBounds">
  <d-param type="java.awt.Rectangle" />
  </d-property>
- <d-property id="layout" maps-type="setMethod" maps-to="setLayout">
  <d-param type="java.awt.LayoutManager" />
  </d-property>
  </d-class>
  </presentation>
  </peers>
```

Slika 5.3.1. *IIS*Case UIML rečnik*

5.3.2. Generisanje SQL upita za prikaz i ažuriranje podataka

Implementacija funkcionalnosti generisanja prototipova aplikacija u okviru alata IIS*Case jedan je od glavnih rezultata sprovedenog istraživanja. Za ove potrebe implementirani su postupci za automatsko generisanje odgovarajućih SQL upita koji podržavaju operacije za prikaz i ažuriranje podataka generisanih transakcionih programa. Programske komponente odnosno objekti klasa `ui.DBTable` i `ui.DBPanel` koji su nastali transformacijom UIML elementima `DataTable` i `DataPanel`, izvršavaju ove upite prilikom akcija čitanja i ažuriranja podataka. Komunikacija generisane aplikacije sa bazom podataka ostvaruje se preko ODBC konekcije. Pristupni podaci za datu ODBC konekciju zadaju se kroz formu datu na slici 3.1.3.1.

Prvi korak u implementaciji procedure automatskog generisanja odgovarajućih SQL upita je generisanje podšema baze podataka koja odgovara tipovima forme datog aplikativnog sistema. Pošto se upiti generišu na nivou tipa komponente, ovime se sužava izbor šema relacija koje predstavljaju ulazni parametar za sam proces generisanja SQL upita. U toku izvršavanja, objekti klasa `ui.DBTable` i `ui.DBPanel` pozivaju objekte klase ui.Query, čije metode izvršavaju odgovarajuće SQL upite na poziv akcija unosa, čitanja, izmjene i brisanja podataka. U ovoj pottački opisani su algoritmi za generisanje skupova šema relacija na osnovu kojih će se formirati dati SQL upiti.

Na slici 5.3.2.1 prikazan je algoritam za generisanje skupova šema relacija za formiranje SQL upita za tip komponente $O \in O$ tipa forme $\mathcal{F}$. U zavisnosti od primjenljivosti tipa forme $\mathcal{F}$, dati skupovi koriste se za formiranje SQL upita za unos, čitanje, modifikovanje i brisanje podataka na tipovima komponenti datog tipa forme. Kao ulazni parametri datog algoritma navedeni su skupovi šema relacija podšema tipa forme $\mathcal{F}$: $P_u(\mathcal{F})$ (u slučaju da dati tip forme pripada klasi za upite) i $P_a(\mathcal{F})$ (u slučaju da dati tip forme pripada klasi za ažuriranje). Skupovi $P_u(\mathcal{F})$ i $P_a(\mathcal{F})$ zadati su redom definicijama 5.1.1.5 i 5.1.2.2. Dati

skupovi prethodno se generišu na osnovu algoritama specificiranih u [22], a opisanih u tački 5.1 ovog poglavlja. U odnosu na [22], algoritmi za generisanje podšema tipova formi u ovom radu uklopljeni su u jedan širi kontekst problema koji podrazumijeva generisanje prototipova aplikacija i implementacije njihove funkcionalnosti za čitanje i ažuriranje podataka.

Algoritmi dati na slikama 5.3.2.2 - 5.3.2.5 predstavljaju pomoćne algoritme koji su sastavni dio algoritma prikazanog na slici 5.3.2.1. Ovaj algoritam generiše sljedeće skupove:

- L_i - skup koji se sastoji od jedne šeme relacije za formiranje upita za unos podataka,
- L_r - skup šema relacija za formiranje upita za čitanje podataka,
- L_{md} - skup šema relacija za formiranje upita za modifikaciju i brisanje podataka.

Na ovaj način, umjesto skupova P_u i P_a koriste se restriktivniji skupovi L_i , L_r i L_{md}, što pojednostavljuje generisanje upita za čitanje i ažuriranje podataka.

Skup L_i sastoji od jedne šeme relacije tzv. "noseće" šeme relacije koja se određuje algoritmom prikazanim na slici 5.3.2.2. U pitanju je šema relacije iz skupa P_u čiji je primarni ključ jednak jednoj od nosećih grupa iz skupa $\chi(O_i)$ datog tipa komponente O_i. U slučaju da takva šema relacije ne postoji, "noseća" šema relacije traži se u skupu P_a. Prvo se izabiraju sve šeme relacija iz skupa P_a čiji su ključevi dijelom sadržani u nekoj od nosećih grupa datog tipa komponente, a zatim se iz datog skupa izabira minimalni čvor koji predstavlja "noseću" šemu relacije.

Skup šema relacija L_r koristi se za formiranje SQL upita za čitanje podataka. Ovaj skup šema relacija izabira se iz skupa P_u na osnovu algoritma datog na slici 5.3.2.3. Prvo se biraju šeme relacije koje su minimalni čvorovi u odnosu na skup P_u i graf zatvaranja $\mathcal{G}$, a čiji su primarni ključevi sadržani u skupu atributa "noseće" šema relacije. Zatim se ovaj skup proširuje tzv. skupom "povezujućih" šema relacija u odnosu na svaku izabranu šemu relacije, koji se generiše na osnovu algoritma prikazanog na slici 5.3.2.4. Ovaj algoritam rekurzivno provjerava za svaku šemu relacije koja je nadređena datoj šemi relacije, da li je njen primarni ključ sadržan u skupu atributa "noseće" šeme relacije, i ako jeste njome proširuje skup "povezujućih" šema relacija.

Skup L_{md} generiše se primjenom algoritma za generisanje skupa šema relacija poddrva datog na slici 4.3.2.6. Na početku, generiše se skup svih šema relacija nadređenih "nosećoj" šemi relacije u odnosu na skup P_a i graf zatvaranja $\mathcal{G}$ na osnovu algoritma datog na slici 4.3.2.5. Zatim se rekurzivnim pozivom iste metode generiše ostatak poddrva. Za svaku od

šema relacija skupa L_{md} potrebno je generisati odgovarajuće upite za modifikaciju i brisanje podataka.

PROCES skupovi_šema_relacija (**U**($P_u(\mathcal{F})$, $P_a(\mathcal{F})$, $\mathcal{G}$, O), **I** (L_i, L_r, L_{md}), **UI** ( ))

(*

U: $P_u(\mathcal{F})$ - skup šema relacija podšeme tipa forme iz klase za upite.

$P_a(\mathcal{F})$ - skup šema relacija podšeme tipa forme iz klase za ažuriranje.

$O_i \in O$ – tip komponente iz skupa tipova komponenti tipa forme $\mathcal{F}$.

$\mathcal{G}$ – graf zatvaranja šeme baze podataka.

$\mathcal{X}(O_i)$ – skup unija po jednog ključa od korijenskog do datog tipa komponente.

I: L_i - šema relacije za koju treba formirati upit za unos podataka.

L_r - skup šema relacija za formiranje upita za čitanje podataka.

L_{md} - skup šema relacija za formiranje upita za modifikaciju i brisanje podataka.

*)

POČETAK PROCESA skupovi_šema_relacija

POZOVI generiši_noseću_šemu_relacije (P_u, P_a, $\mathcal{G}$, O, (R_n, K_n))

POSTAVI $L_i \leftarrow \{(R_n, K_n)\}$

POZOVI skup_šema_relacija_za_čitanje_podataka (P_u, $\mathcal{G}$, (R_n, K_n), L_r)

POZOVI generisanje_poddrva (P_a, $\mathcal{G}$, (R_n, K_n), L_{md})

KRAJ PROCESA skupovi_šema_relacija

Slika 5.3.2.1. *Algoritam za generisanje skupova šema relacija za formiranje upita za čitanje i ažuriranje podataka*

PROCES generiši_noseću_šemu_relacije (**U**($P_u(\mathcal{F})$, $P_a(\mathcal{F})$, $\mathcal{G}$, O) ,

I ((R_n, K_n)), **UI** ( ))

(*

U: $P_u(\mathcal{F})$ - skup šema relacija podšeme tipa forme iz klase za upite.

$P_a(\mathcal{F})$ - skup šema relacija podšeme tipa forme iz klase za ažuriranje.

$O_i \in O$ – tip komponente iz skupa tipova komponenti tipa forme $\mathcal{F}$.

$\mathcal{G}$ – graf zatvaranja šeme baze podataka.

$\mathcal{X}(O_i)$ – skup unija po jednog ključa od korijenskog do datog tipa komponente

I: (R_n, K_n) - "noseća" šeme relacije

*)

POČETAK PROCESA generiši_noseću_šemu_relacije

ind ← *false*

RADI noseća_šema_relacije za $(\forall (R_i, K_i) \in P_u(\mathcal{F}))$ **DOK JE** ¬ *ind*

AKO JE $(K_i^p \in K_i)\ (\exists X \in \mathcal{X}(O_i))\ K_i^p = X$ **TADA**

POSTAVI $(R_n, K_n) \leftarrow (R_i, K_i)$

POSTAVI *ind* ← *true*

INAČE

KRAJ AKO

KRAJ RADI noseća_šema_relacije

AKO JE ¬ *ind* **TADA**

POSTAVI *ind1* ← *false*

POSTAVI $S \leftarrow \emptyset$

RADI kreiranje_*S* za $(\forall (R_i, K_i) \in P_a(\mathcal{F}))$ **DOK JE** ¬ *ind1*

POSTAVI $(R_n, K_n) \leftarrow (R_i, K_i)$

RADI provjera_ključa za $(\forall K_i \in K_i)$ **DOK JE** ¬ *ind1*

AKO JE $(\exists X \in \mathcal{X}(O_i))\ K_i \cap X \neq \emptyset$ **TADA**

POSTAVI $S \leftarrow S \cup (R_i, K_i)$

POSTAVI *ind1* ← *true*

INAČE

KRAJ AKO

KRAJ RADI provjera_ključa

KRAJ RADI kreiranje_*S*

RADI noseća_šema_relacije_ P_u za $(\forall (R_i, K_i) \in S)$ **DOK JE** ¬ *ind1*

AKO JE $(\neg \exists (R_j, K_j) \in S)\ \{(R_j, K_j), (R_i, K_i)\} \in \mathcal{G}$ **TADA**

POSTAVI $(R_n, K_n) \leftarrow (R_i, K_i)$

POSTAVI *ind1* ← *true*

INAČE

```
        KRAJ AKO
      KRAJ RADI noseća_šema_relacije_ P_u
    INAČE
    KRAJ AKO
KRAJ PROCESA generiši_noseću_šemu_relacije
```

Slika 5.3.2.2. *Algoritam za generisanje* "noseće" šeme relacije

```
PROCES skup_šema_relacija_za_čitanje_podataka ( U( P, 𝒢, (R_n, K_n )), I (L), UI ( ) )
( *
U:    P - skup šema relacija.
      𝒢 – graf zatvaranja šeme baze podataka.
      (R_n, K_n ) – noseća šema relacije.
I:    L - skup šema relacija za upite
* )
POČETAK PROCESA skup_šema_relacija_za_čitanje_podataka
  POSTAVI S_pom ← Ø
  POSTAVI L ← Ø
  RADI šeme_relacije_ P za (∀ (R_i, K_i) ∈ P) DOK JE R_n ⊄ S_pom
    AKO JE (¬ ∃(R_j, K_j)∈ S) {(R_j, K_j), (R_i, K_i)}∈ 𝒢 TADA
      AKO JE (K_i^p ∈ K_i) K_i^p ⊆ R_n TADA
        POSTAVI S_pom ← S_pom ∪ R_i
        POSTAVI L ← L ∪ {(R_i, K_i)}
      INAČE
      KRAJ AKO
    INAČE
    KRAJ AKO
    POZOVI vezani_skup_šema_relacija (P, 𝒢, (R_i, K_i), S_pom, L1)
    POSTAVI L ← L ∪ L1
  KRAJ RADI šeme_relacije_ P
KRAJ PROCESA skup_šema_relacija_za_čitanje_podataka
```

Slika 5.3.2.3. *Algoritam za generisanje skupa šema relacija potrebnih za formiranje upita za čitanje podataka*

PROCES skup_ povezujućih_šema_relacija (**U**(P, $\mathcal{G}$, (R, K), (R_n , K_n), S_pom),
I (L), **UI** ( ))
(*
U: P – skup šema relacija.
$\mathcal{G}$ – graf zatvaranja šeme baze podataka.
(R, K) – šema relacije u odnosu na koju se formira vezani skup
(R_n , K_n) – noseća šema relacije
S_pom – pomoćni skup šema relacija.
I: L – skup "povezujućih" šema relacija
*)
POČETAK PROCESA skup_ povezujućih_šema_relacija
POZOVI skup_nadređenih_šema_relacija (P, $\mathcal{G}$, (R, K), *Sup*)
POSTAVI $L \leftarrow \emptyset$
RADI traži ($\forall$ (R_i , K_i) $\in$ *Sup*)
AKO JE ($K_i^p \in K_i$) $K_i^p \subseteq R_n$ **TADA**
POSTAVI $L \leftarrow L \cup \{(R_i, K_i)\}$
POSTAVI S_pom $\leftarrow$ S_pom $\cup$ R_i
INAČE
KRAJ AKO
POZOVI vezani_skup_šema_relacija (P, $\mathcal{G}$, (R_i , K_i), S_pom, L_1)
POSTAVI $L \leftarrow L \cup L_1$
KRAJ RADI traži
KRAJ PROCESA skup_ povezujućih_šema_relacija

Slika 5.3.2.4. *Algoritam za generisanje skupa "povezujućih" šema relacija*

PROCES skup_nadređenih_šema_relacija (**U**(P, $\mathcal{G}$, (R, K)) , **I** (*Sup*), **UI** ( ))
(*
U: P - skup šema relacija.
$\mathcal{G}$ – graf zatvaranja šeme baze podataka.
(R, K) – šema relacije u odnosu na koju se formira skup nadređenih šema relacija iz skupa P u odnosu na graf $\mathcal{G}$.
I: *Sup* – skup nadređenih šema relacija
*)
POČETAK PROCESA skup_nadređenih_šema_relacija
POSTAVI $N \leftarrow \emptyset$

RADI traži $(\forall (R_i, K_i) \in P)$
AKO JE $\{(R, K), (R_i, K_i)\} \in \mathcal{G}$ **TADA**
POSTAVI $Sup \leftarrow Sup \cup (R_i, K_i)$
INAČE
KRAJ AKO
KRAJ RADI traži
KRAJ PROCESA skup_nadređenih_šema_relacija

Slika 5.3.2.5. *Algoritam za generisanje skupa direktno nadređenih šema relacija*

PROCES generisanje_poddrva (**U**(P, $\mathcal{G}$, (R, *K*)) , **I** (*L*), **UI** ( ))
(*
U: P - skup šema relacija.
$\mathcal{G}$ – graf zatvaranja šeme baze podataka.
(R, *K*) – šema relacije u odnosu na koju se formira poddrvo u odnosu na graf $\mathcal{G}$, koga čini skup šema relacija iz skupa P.
I: *L* – skup šema relacija koje pripadaju generisanom poddrvu.
*)
POČETAK PROCESA generisanje_poddrva
$L \leftarrow L \cup (R, K)$
POZOVI skup_nadređenih_šema_relacija (P, $\mathcal{G}$, (R, *K*), *Sup*)
RADI traži $(\forall (R_i, K_i) \in Sup)$
AKO JE $\{(R, K), (R_i, K_i)\} \in \mathcal{G}$ **TADA**
POZOVI generiši_poddrvo (P, $\mathcal{G}$, $(R_i, K_i), L_1$)
POSTAVI $L \leftarrow L \cup L_1$
INAČE
KRAJ AKO
KRAJ RADI traži
KRAJ PROCESA generisanje_poddrva

Slika 5.3.2.6. *Algoritam za generisanje skupa šema relacija poddrva*

Na osnovu skupova šema relacija generisanih primjenom navedenih algoritama, dalje se formiraju SQL upiti koji obezbjeđuju sve potrebne funkcionalnosti prikaza i pretrage, kao i operacija unosa, modifikacije i brisanja podataka, u okviru generisanih prototipova aplikacija. Ovime se zaokružuje proces generisanja aplikacija, dodavanjem finalne funkcionalnosti koja kreirani korisnički interfejs, u realnom vremenu, povezuje sa generisanom šemom baze podataka.

U ovom trenutku, generisanje prototipova aplikacija predstavlja finalnu fazu razvoja informacionih sistema u okviru okruženja IIS*Studio. Budući pravci razvoja, između ostalog, orjentisani su ka generisanju finalnih implementacija aplikativnih sistema, odnosno mogućnosti automatskog generisanja koda u izabranim programskim okruženjima.

6. Zaključak

U radu je opisan jedan pristup generisanju izvršnih softverskih specifikacija i obavljena je verifikacija definisanih postupaka kroz praktičnu implementaciju softverskog alata za generisanje softverskih specifikacija. Implementirani alat je integrisan u šire softversko okruženje za razvoj informacionih sistema IIS*Studio, sa ciljem praktičnog obezbjeđenja razvoja softvera zasnovanog na savremenim modelski orijentisanim pristupima. Pokazano je da alati razvojnog okruženja IIS*Studio zaokružuju proces projektovanja informacionog sistema počevši od specifikacije korisničkih zahtjeva do implementacije projektovane šeme baze podataka i generisanja funkcionalnog prototipa aplikativnog sistema. Softverskom podrškom aktivnosti kreiranja i primjene šablona korisničkog interfejsa, funkcionalno modelovanje aplikativnog sistema prošireno je i oblikovanjem opštih modela korisničkog interfejsa aplikacije.

Implementacijom IIS*Studio alata IIS*UIModeler, kao i proširenjem funkcionalnosti alata IIS*Case generatorom prototipova aplikacija, te njegovom uspješnom primjenom na odabranom praktičnom primjeru, pokazano je da putem formalnog modela zasnovanog na konceptu tipa forme, ne samo što je moguće uspješno generisati šemu baze podataka, već je moguće i precizno specificirati i generisati korisnički interfejs aplikativnog sistema koji zadovoljava potrebe krajnjeg korisnika.

Dalji razvoj alata okruženja IIS*Studio zasnovan je na proširenju postojećih funkcionalnosti. Najvažniji pravci razvoja su:

- implementacija *render*-a koji bi mogao interpretirati specifikacije aplikacija informacionih sistema u drugim programskim okruženjima i
- implementacija različitih generatora programskog koda.

Generator aplikacija implementiran u IIS*Case-u generiše prototip aplikativnog sistema u *Java* programskom okruženju. Implementacija *render*-a koji bi datu specifikaciju aplikacija informacionih sistema mogao interpretirati u okviru nekog drugog programskog okruženja, npr. .NET okruženja primjenom *render*-a specificiranog u [30], značajno bi unapredio funkcionalnost automatskog generisanja.

Konceptima ugrađenim u IIS*Case moguće je opisati praktično sve elemente nekog aplikativnog sistema, što daje mogućnost generisanja ne samo prototipova, već i kompletnog programskog koda aplikativnog sistema. Na ovaj način, projektant bi, osim mogućnosti generisanja desktop aplikacija, imao mogućnost primjene generatora programskog koda za odabrane skript jezike, kao što je npr. široko upotrebljavani PHP, kako bi generisao, ali i

potpuno automatizovano implementirao *web* aplikaciju, spremnu za upotrebu u komercijalnoj produkciji.

S druge strane, dalji razvoj prezentovanih pristupa i alata je usmjeren ka daljem približavanju MDA standardima. To prije svega znači implementaciju podrške za UML 2.0 u alatima okruženja IIS*Studio, ali i dalje unapređenje i proširenje koncepata, na kojima se opisani postupci zasnivaju, u cilju boljeg usklađivanja sa UML 2.0 konceptima.

IIS*Case podržava generisanje specifikacija projektovane šeme baze podataka i specifikacije korisničkog interfejsa aplikativnog sistema, zasnovane na XML-u. Neophodno je, u mogućoj mjeri, obezbjediti funkciju izvoza i uvoza takvih specifikacija kao XMI specifikacija PIM modela. Primjenom metode transformacije modela moguće je date XMI specifikacije transformisati u XMI specifikacije koje je moguće uvoziti u druge MDA alate. Na taj način, bila bi omogućena interoperabilnost sa UML 2.0 alatima koji se danas sve češće upotrebljavaju u procesu modelovanja informacionih sistema.

Kako je navedeno u [26], planira se dalje unapređenje koncepta izraza ograničenja i primjene domenski specifičnih jezika. Ovaj pravac obuhvata razvoj DSL jezika za reprezentaciju koncepta funkcija na nivou PIM modela, kao i razvoj algoritama za transformaciju izraza ograničenja definisanih na nivou tipova formi (PIM model), u izvršnu PSM specifikaciju, odnosno SQL/DDL programski kod.

Prilog A. Algoritmi za generisanje podšema tipova formi

PROCES minimalni_čvorovi_podšeme (**U**($\mathcal{G}$, $\tilde{\mathcal{G}}$, M, T($\mathcal{F}$)) , **I** (P^S_{min} ($\mathcal{F}$)), **UI** ( ))

(*

U: Grafovi zatvaranja $\mathcal{G}$ i $\tilde{\mathcal{G}}$, skup minimalnih čvorova grafa M i skup nosećih grupa obilježja tipa forme T($\mathcal{F}$).

I: Skup šema relacija $P^S_{min} = \{ (R_i, K_i) \in S \mid Min(R_i) \}$

*)

POČETAK PROCESA minimalni_čvorovi_podšeme

POSTAVI $P^S_{min} \leftarrow \emptyset$

RADI minimalni_element za ($\forall X_i \in T$)

POSTAVI $M\_ind \leftarrow false$

RADI izbor_iz_M za ($\forall m_j \in M$) **DOK JE** $\neg M\_ind$

AKO JE ($\exists K_{m_j} \in K_{m_j}$) ($X_i = K_{m_j}$) **TADA** (* K_{m_j} je ključ elementa m_j *)

POSTAVI $M\_ind \leftarrow true$

POSTAVI $P^S_{min} \leftarrow P^S_{min} \cup \{m_j\}$

INAČE

AKO JE $X_i \subseteq \left[\bigcup_{R_j \in \tilde{\Pi}(m_i)} R_j \right]$ **TADA** (* $X_i \subseteq (m_i)^+_\Gamma$ *)

POSTAVI $M\_ind \leftarrow true$

POSTAVI $G\_ind \leftarrow true$

POSTAVI $max \leftarrow m_i$

RADI snižavanje_minimuma **DOK JE** G_ind

POSTAVI $P\_ind \leftarrow false$

RADI izbor_minimuma za ($\forall R_j \in \Pi(max)$) **DOK JE** $\neg P\_ind$

AKO JE $X_i \subseteq \left[\bigcup_{R_i \in \tilde{\Pi}(R_j)} R_i \right]$ **TADA** (* $X_i \subseteq (R_j)^+_\Gamma$ *)

POSTAVI $P\_ind \leftarrow true$

POSTAVI $max \leftarrow R_j$

INAČE

KRAJ AKO
KRAJ RADI izbor minimuma
AKO JE $(\neg P\_ind) \vee (\Pi(max) = \emptyset)$ **TADA**
POSTAVI $G\_ind \leftarrow false$
INAČE
KRAJ AKO
KRAJ RADI snižavanje_minimuma
POSTAVI $P^S_{min} \leftarrow P^S_{min} \cup \{max\}$
INAČE
KRAJ AKO
KRAJ AKO
KRAJ RADI izbor_iz_M
KRAJ RADI minimalni_element
KRAJ PROCESA minimalni_čvorovi_podšeme

Slika A.1. *Algoritam generisanje skupa minimalnih čvorova*

PROCES minimalni_čvorovi (**U** ($\mathcal{G}$), **I** (M), **UI** ( ))
(*
U: Graf zatvaranja $\mathcal{G} = (S, \Pi, H)$
I: $M = \{R_i \in S \mid \neg(\exists R_j \in S)((R_j, R_i) \in \rho)\}$
*)
POČETAK PROCESA minimalni_čvorovi
POSTAVI $M \leftarrow \emptyset$
RADI traženje za $(\forall R_i \in S)$
AKO JE $H(R_i) = \emptyset$ **TADA**
POSTAVI $M \leftarrow M \cup \{R_i\}$
INAČE
KRAJ AKO
KRAJ RADI traženje
KRAJ PROCESA minimalni_čvorovi

Slika A.2. *Algoritam formiranja skupa minimalnih čvorova grafa zatvaranja*

PROCES relevantni_čvorovi (**U**(P^{S}_{min}, $W(\mathcal{F})$, $\overline{\mathrm{T}}(\mathcal{F})$, $\mathcal{G}$) , **I** (R_c, Ω), **UI** ( ))

(*

U: Skup minimalnih čvorova P^{S}_{min}, skup obilježja tipa forme $W(\mathcal{F})$,

$\overline{\mathrm{T}}(\mathcal{F}) = \{ \mathrm{X_i} \subseteq W \mid (\exists\, \mathrm{O_j} \in O)(\, \mathrm{X_i} \in \mathcal{X}(\mathrm{O_j})\,) \}$, i graf $\mathcal{G} = (S, \overline{\Pi}, \overline{\mathrm{H}})$.

I: $R_c(\mathcal{F}) = P^{S}_{min} \cup P^{S}_{rlp}$, $P^{S}_{rlp} = \{ (\mathrm{R_j}, K_j) \in S \mid (\exists\, \mathrm{R_i} \in P^{S}_{min})\,(\mathrm{R_j} \in \Omega(\mathrm{R_i}))\,\}$.

$\Omega : P^{S}_{min} \rightarrow P(S)$ – funkcija relevantnih čvorova.

*)

POČETAK PROCESA relevantni_čvorovi

POSTAVI za $(\forall\, \mathrm{R_i} \in P^{S}_{min})\,(\Omega(\mathrm{R_i}) \leftarrow \emptyset)$

RADI lociranje za $(\forall\, \mathrm{R_j} \in S \setminus P^{S}_{min})$

AKO JE $(\mathrm{R_j} \cap W) \setminus \mathrm{K^p_j} \neq \emptyset \vee (\exists\, \mathrm{X_k} \in \overline{\mathrm{T}}(\mathcal{F}))(\exists\, \mathrm{K_j} \in K_j)(\mathrm{X_k} = \mathrm{K_j})$ **TADA**

(* $\mathrm{K^p_j}$ je prtimarni ključ šeme relacije $(\mathrm{R_j}, K_j)$ *)

RADI traženje za $(\forall\, \mathrm{R_i} \in P^{S}_{min})$

AKO JE $\mathrm{R_j} \in \tilde{\Pi}(\mathrm{R_i})$ **TADA**

POSTAVI $\Omega(\mathrm{R_i}) \leftarrow \Omega(\mathrm{R_i}) \cup \{\mathrm{R_j}\}$

INAČE

KRAJ AKO

KRAJ RADI traženje

INAČE

KRAJ AKO

KRAJ RADI lociranje

POSTAVI $R_c(\mathcal{F}) \leftarrow P^{S}_{min}$

RADI formiranje_skupa_rel_čvorova za $(\forall\, \mathrm{R_i} \in P^{S}_{min})$

POSTAVI $R_c \leftarrow R_c \cup \Omega(\mathrm{R_i})$

KRAJ RADI formiranje_skupa_rel_čvorova

KRAJ PROCESA relevantni_ čvorovi

Slika A.3. *Algoritam generisanja skupa relevantnih čvorova*

PROCES izbor_max_zast_čvora (**U** (*SC*, *ZAST* , *ind* , SIG_t) , **I** (R_{max}), **UI** ( ))

(*

U: $SC = \{ R_1, ..., R_k \}$ – skup čvorova

$ZAST : SC \rightarrow N_o$ – funkcija zastupljenosti čvorova

$ind \in \{true , false\}$ – određuje da li se testira pripadnost maksimalnog elementa u SIG_t

I: $R_{max} \in SC$ – čvor sa maksimalnom zastupljenošću

*)

POČETAK PROCESA izbor_max_zast_čvora

POSTAVI $MAXZAST \leftarrow Ø$

RADI izbor_max za ($\forall R'_k \in SC$)

AKO JE $ZAST(R'_k) > MAXZAST$ **TADA**

POSTAVI $R_{max} \leftarrow R'_k$

POSTAVI $MAXZAST \leftarrow ZAST(R'_k)$

INAČE

AKO JE $ind \wedge (ZAST(R'_k) = MAXZAST) \wedge (R'_k \in SIG_t)$ **TADA**

POSTAVI $R_{max} \leftarrow R'_k$

INAČE

KRAJ AKO

KRAJ AKO

KRAJ RADI izbor_max

KRAJ PROCESA izbor_max_zast_čvora

Slika A.4. *Algoritam izbora čvora sa maksimalnom zastupljenošću*

PROCES relevantni_i_vezni_čvorovi (**U** (*RI* , Π , $\tilde{\Pi}$) , **I** (Θ , *ZAST*, *RSL*, *RLD*), **UI** (*RLN*, Ω_t , SIG_t))

(*

U: *RI* – polazni čvor, Π i $\tilde{\Pi}$ - funkcije podređenih čvorova

I: *ZAST* – funkcija zastupljenosti čvora na putu do nekog relevantnog čvora

$\Theta(RI, R_j)$ – skup veznih čvorova od *RI* do R_j

RSL – relevantni čvorovi sljedećeg nivoa

RLD – skup podređenih relevantnih čvorova do kojih može da se stigne iz *RI*

UI: *RLN* – skup podređenih relevantnih čvorova za koje se ispituje mogućnost generisanja puta počev od čvora *RI*

Ω_t – skup podređenih relevantnih čvorova za koje još nije kompletiran put, počev od minimalnog čvora R_t

SIG_t – skup čvorova koji sigurno ulaze u sastav skupa P^s_{pod}

*)

POČETAK PROCESA relevantni_i_vezni_čvorovi

POSTAVI $RSL \leftarrow \emptyset$

(* Inicijalizacija skupa čvorova sljedećeg nivoa *)

POSTAVI $RLD \leftarrow \emptyset$

POSTAVI $(\forall R_l \in \Pi(RI))(ZAST(R_l) \leftarrow 0)$

RADI ispitivanje_relevantnih_čvorova $(\forall R_j \in RLN)$

AKO JE $R_j \in \Pi(RI)$ **TADA**

POSTAVI $SIG_t \leftarrow SIG_t \cup \{R_j\}$

POSTAVI $\Omega_t \leftarrow \Omega_t \setminus \{R_j\}$

POSTAVI $RLN \leftarrow RLN \setminus \{R_j\}$

POSTAVI $RSL \leftarrow RSL \cup \{R_j\}$

INAČE (* $R_j \subseteq (R_l)^+_\Gamma$ *)

POSTAVI $\Theta(RI, R_j) \leftarrow \{R_l \in \Pi(RI) \mid R_j \subseteq \left[\bigcup_{R_k \in \tilde{\Pi}(R_l)} R_k\right]\}$

(* $\Theta(RI, R_j)$ – skup kandidata za generisanje puta do R_j *)

AKO JE $\Theta(RI, R_j) \neq \emptyset$ **TADA**

POSTAVI $RLD \leftarrow RLD \cup \{R_j\}$

POSTAVI $RLN \leftarrow RLN \setminus \{R_j\}$

POSTAVI $(\forall R_l \in \Theta(RI, R_j))(ZAST(R_l) \leftarrow ZAST(R_l) + 1)$

INAČE

KRAJ AKO

```
      KRAJ AKO
   KRAJ RADI ispitivanje_relevantnih_čvorova
KRAJ PROCESA relevantni_i_vezni_čvorovi
```

Slika A.5. *Algoritam ispitivanja relevantnih i formiranja skupa veznih čvorova*

```
PROCES propagacija_puta ( U (RI , Π , Θ , ZAST, SIG_t , RLD) , I (SLN_i ), UI ( ) )
( *
I:      SLN_i          – skup čvorova sljedećeg nivoa
*)
POČETAK PROCESA propagacija_puta
   POSTAVI SC ← Π ( RI )
   POSTAVI SLN_i ← Ø
   RADI propagacija DOK JE RLD ≠ Ø
      POZOVI izbor_max_zast_čvora ( SC , ZAST , true , SIG_t , R_l)
      ( *   favorizuju se čvorovi koji su već sadržani u SIG_t   * )
      POSTAVI SC ← SC \ { R_l }
      POSTAVI POM ← Ø
      RADI izbor_zastupljenih_čvorova (∀ R_j ∈ RLD)
         AKO JE R_l ∈ Θ (RI , R_j) TADA
            POSTAVI POM ← POM ∪ { R_j }
         INAČE
         KRAJ AKO
      KRAJ RADI izbor_zastupljenih_čvorova
      AKO JE POM ≠ Ø TADA
         RLD ← RLD \ POM
         SLN_i ← SLN_i ∪ { R_l }
      INAČE
      KRAJ AKO
   KRAJ RADI propagacija
KRAJ PROCESA propagacija_puta
```

Slika A.6. *Algoritam propagacije puta*

PROCES prebacivanja_kandidata (**U** (RI , RSL, SLN_i, P^S_{min}) , **I** ( ), **UI** (SIG_t))

POČETAK PROCESA prebacivanja_kandidata

  AKO JE $(RI \notin SIG_t) \wedge (RI \notin P^S_{min})$ **TADA**

    POSTAVI $ind \leftarrow false$

    RADI sljedeći_nivo za ($\forall R_m \in SLN_i \cup RSL$) **DOK JE** $\neg\ ind$

      AKO JE ($K^p_m \not\subset K^p_I$) **TADA** (* K^p_I je primarni ključ čvora RI *)

        POSTAVI $ind \leftarrow true$

      INAČE

      KRAJ AKO

    KRAJ RADI sljedeći_nivo

    AKO JE ind **TADA**

      POSTAVI $SIG_t \leftarrow SIG_t \cup \{RI\}$

    INAČE

    KRAJ AKO

  INAČE

  KRAJ AKO

KRAJ PROCESA prebacivanja_kandidata

Slika A.7. *Algoritam prebacivanja kandidata u skup sigurnih čvorova*

PROCES generisanje_P^S_{pod} (**U** ($\mathcal{G}$,$\tilde{\mathcal{G}}$, P^S_{min}, R_c, Ω) , **I** (P^S_{pod}), **UI** ( ))

(*

U: Grafovi zatvaranja $\mathcal{G}$ i $\tilde{\mathcal{G}}$, skup minimalnih čvorova P^S_{min}, skup relevantnih čvorova R_c i funkcija relevantnih čvorova Ω.

I: Skup šema relacija P^S_{pod}.

*)

POČETAK PROCESA generisanje_P^S_{pod}

  POSTAVI $RC \leftarrow R_c \setminus P^S_{min}$

  POSTAVI $(\forall R_t \in P^S_{min})(ZAST(R_t) \leftarrow |\ \Omega(R_t)\ |)$

  POSTAVI $PSMIN \leftarrow P^S_{min}$

  RADI formiranje_puta **DOK JE** $PSMIN \neq \emptyset$

POZOVI izbor_max_zast_čvora (*PSMIN*, *ZAST* , *false* , Ø, R_t)

POSTAVI $PSMIN \leftarrow PSMIN \setminus \{R_t\}$

POSTAVI $TNV_t \leftarrow \{R_t\}$

(* TNV_t skup čvorova tekućeg nivoa od kojih se generišu putevi *)

POSTAVI $SIG_t \leftarrow Ø$ (* skup sigurnih čvorova za generisani put *)

POSTAVI $\Omega_t \leftarrow \Omega(R_t) \cap RC$ (* u skup Ω_t se stavljaju samo oni čvorovi koji nisu već obrađeni po osnovu nekog drugog minimalnog čvora *)

POSTAVI $\Omega_t \leftarrow RC \setminus \Omega(R_t)$ (* inicijalizacija skupa podređenih relevantnih čvorova za koje još nije generisan put od čvora R_t *)

RADI generisanje_sljedećeg_nivoa **DOK JE** $\Omega_t \neq Ø$

POSTAVI $RLN \leftarrow \Omega_t$ (* skup relevantnih čvorova za koje još nije izabran vezni čvor *RI* iz tekućeg nivoa čvorova *)

POSTAVI $NTN_t \leftarrow | TNV_t |$ (* broj čvorova tekućeg nivoa *)

POSTAVI $i \leftarrow NTN_t$ (* broj neobrađenih čvorova tekućeg nivoa *)

RADI generisanje_puteva **DOK JE** $i \neq Ø$

POZOVI izbor_max_zast_čvora (TNV_t, *ZAST* , *false* , Ø , *RI*)

(* izbor čvora $R_i \in TNV_t$ sa maksimalnim brojem $ZAST(R_i)$ *)

POSTAVI $TNV_t \leftarrow TNV_t \setminus \{RI\}$

POZOVI relevantni_i_vezni_čvorovi (*RI*, Π , $\tilde{\Pi}$, Θ , *ZAST*, *RSL*, *RLD*, *RLN*, Ω_t, SIG_t)

POZOVI propagacija_puta (*RI* , Π , Θ , *ZAST*, SIG_t , *RLD* , SLN_i)

POZOVI prebacivanje_kandidata (*RI* , *RSL*, SLN_i , P^S_{min} , SIG_t)

POSTAVI $i \leftarrow i$ -1

KRAJ RADI generisanje_puteva

POSTAVI $TNV_t \leftarrow \bigcup_{i=1}^{NTN_t} (SLN_i)$ (* sljedeći nivo postaje tekući *)

KRAJ RADI generisanje_sljedećeg_nivoa

KRAJ RADI formiranje_puta

POSTAVI $P^S_{pod} \leftarrow Ø$

RADI inicijalizacija_ P^S_{pod} za ($\forall$ i$\in$ {1, …, | P^S_{min} | })

POSTAVI $P^S_{pod} \leftarrow P^S_{pod} \cup SIG_i$

KRAJ RADI inicijalizacija_ P^S_{pod}

KRAJ PROCESA generisanje_ P^S_{pod}

Slika A.8. *Algoritam formiranja skupa potrebnih podređenih čvorova*

PROCES skupovi_za_ažuriranje (**U** ($P^S_u(\mathcal{F})$, $W(\mathcal{F})$, $O(\mathcal{F})$, $K_p(\mathcal{F})$, $\mathcal{X}$, f, *dom*), **I** (*OIns* ($\mathcal{F}$), *ODel* ($\mathcal{F}$), *WUpd* ($\mathcal{F}$)), **UI** ( ))

(*

U: $P^S_u(\mathcal{F})$ – skup šema relacija šeme baze podataka za formiranje podšeme:

$P^S_u(\mathcal{F}) = P^S_{min}(\mathcal{F}) \cup P^S_{pod}(\mathcal{F})$.

$W(\mathcal{F})$ – skup obilježja tipa forme.

$O(\mathcal{F})$ – skup tipova komponenti tipa forme.

$K_p(\mathcal{F})$ – skup primarnih obilježja tipa forme tipa forme.

$\mathcal{X}(O_i)$ – skup unija po jednog ključa od korijenskog do datog tipa komponente.

f: U $\rightarrow P$(U) – funkcija izvođenja uloge obilježja.

I: $OIns(\mathcal{F}) = \{(O_i, R_j) \mid O_i \in O(\mathcal{F}) \wedge R_j \in P^S_u(\mathcal{F}) \wedge Ins(O_i, R_j)\}$

$ODel(\mathcal{F}) = \{(O_i, R_j) \mid O_i \in O(\mathcal{F}) \wedge R_j \in P^S_u(\mathcal{F}) \wedge Del(O_i, R_j)\}$

$WUpd(\mathcal{F}) = \{A \in W(\mathcal{F}) \mid Upd(A)\}$

*)

POČETAK PROCESA skup_za_ažuriranje

POSTAVI $OIns(\mathcal{F}) \leftarrow \emptyset$

POSTAVI $ODel(\mathcal{F}) \leftarrow \emptyset$

POSTAVI $WUpd(\mathcal{F}) \leftarrow \emptyset$

POSTAVI $P \leftarrow P^S_u(\mathcal{F})$

RADI pretraga_komponenti za ($\forall O_i \in O$)

POSTAVI *ind* $\leftarrow$ *false*

RADI pretraga_šema relacija za ($\forall (R_j, K_j) \in P$) **DOK JE** $\neg$ *ind*

AKO JE $(\exists X_i \in \mathcal{X}(O_i))(\exists K_j \in K_j)(X_i = K_j)$ **TADA**

POSTAVI $ODel(\mathcal{F}) \leftarrow ODel(\mathcal{F}) \cup \{(O_i, R_j)\}$

POSTAVI $P \leftarrow P \setminus \{(R_j, K_j)\}$

POSTAVI $ind \leftarrow true$

AKO JE $(\forall K_j \in K_j)(\exists X_i \in \mathcal{X}(O_i))(X_i = K_j) \wedge$

$(\forall A \in R_j \setminus W(\mathcal{F}))(w \in dom(A))$ **TADA**

POSTAVI $OIns(\mathcal{F}) \leftarrow OIns(\mathcal{F}) \cup \{(O_i, R_j)\}$

INAČE

KRAJ AKO

INAČE

KRAJ AKO

KRAJ RADI pretraga_šema relacija

KRAJ RADI pretraga_komponenti

RADI obilježja za ažuriranje za $(\forall A \in W(\mathcal{F}) \setminus K_p(\mathcal{F}))$

AKO JE $f(A) = \emptyset$ **TADA**

POSTAVI $ind \leftarrow false$

RADI prolaz_kroz_*ODel* $(\forall (O_i, R_j) \in ODel(\mathcal{F}))$ **DOK JE** $\neg\ ind$

AKO JE $A \in N_i \cap R_j$ **TADA** (* N_i je skup obilježja tipa komponente O_i *)

POSTAVI $ind \leftarrow true$

POSTAVI $WUpd(\mathcal{F}) \leftarrow WUpd(\mathcal{F}) \cup \{A\}$

INAČE

KRAJ AKO

KRAJ RADI prolaz_kroz_*ODel*

INAČE

KRAJ AKO

KRAJ RADI obilježja za ažuriranje

KRAJ PROCESA skup_za_ažuriranje

Slika A.9. *Algoritam analize primjenljivosti forme iz klase* $A_{\mathcal{F}}$ *(a) za ažuriranje*

PROCES formiranje_ P^S_{ref} (**U** ($\mathcal{G}$, $OIns(\mathcal{F})$, $ODel(\mathcal{F})$, $WUpd(\mathcal{F})$) ,

I ($P^S_{ref}(\mathcal{F})$) , **UI** ( ))

(*

U: Graf zatvaranja $\mathcal{G}$ i skupovi $OIns(\mathcal{F})$, $ODel(\mathcal{F})$, $WUpd(\mathcal{F})$.

I: Skup šema relacija $P^S_{ref}(\mathcal{F}) = \{(\mathrm{R_i}, K_i) \in S \mid Ref(\mathrm{R_i})\}$.

*)

POČETAK PROCESA formiranje_ P^S_{ref}

POSTAVI $P^S_{ref} \leftarrow \emptyset$

RADI prolaz_kroz_*ODel* za ($\forall (\mathrm{O_i}, \mathrm{R_i}) \in ODel(\mathcal{F})$)

POSTAVI $P^S_{ref} \leftarrow P^S_{ref} \cup \mathrm{H}(\mathrm{R_i})$

RADI prolaz_kroz_podređene za ($\forall\ \mathrm{R_k} \in \Pi(\mathrm{R_i})$)

AKO JE $\left[WUpd(\mathcal{F}) \cap \mathrm{K^p_k} \neq \emptyset\right] \vee \left[(\mathrm{O_i},\ \mathrm{R_i}) \in OIns(\mathcal{F}) \wedge\ \mathrm{K^p_k} \subseteq \mathrm{R_i}\right]$ **TADA**

(* $\mathrm{K^p_k}$ je primarni ključ šeme relacije $\mathrm{R_i}$ *)

POSTAVI $P^S_{ref} \leftarrow P^S_{ref} \cup \{\mathrm{R_k}\}$

INAČE

KRAJ AKO

KRAJ RADI prolaz_kroz_ podređene

KRAJ RADI prolaz_kroz_*ODel*

KRAJ PROCESA formiranje_ P^S_{ref}

Slika A.10. *Algoritam za formiranje skupa referentnih čvorova*

PROCES generisanje_ P_a (**U** ($\mathcal{G}, \tilde{\mathcal{G}}, M,\ O(\mathcal{F}),\ \overline{\mathrm{T}}(\mathcal{F}), \mathrm{T}(\mathcal{F}), W(\mathcal{F}),\ \mathrm{K_p}(\mathcal{F})$,

$\mathcal{X}$, f , dom) , **I** ($P_u(\mathcal{F}), P_a(\mathcal{F})$) , **UI** ( ))

POČETAK PROCESA generisanje_ P_a

POZOVI minimalni_čvorovi_podšeme ($\mathcal{G}, \tilde{\mathcal{G}}, M,\ \mathrm{T}(\mathcal{F}), P^S_{min}$)

POZOVI relevantni_čvorovi (P^S_{min}, $W(\mathcal{F})$, $\overline{\mathrm{T}}(\mathcal{F})$, $\tilde{\mathcal{G}}$, R_c, Ω)

POZOVI generisanje_ P^S_{pod} ($\mathcal{G}$, $\tilde{\mathcal{G}}$, P^S_{min}, R_c, Ω, P^S_{pod})

POSTAVI $P^S_u \leftarrow P^S_{min} \cup P^S_{pod}$

POZOVI skupovi_za_ažuriranje ($P^S_u(\mathcal{F})$, $W(\mathcal{F})$, $O(\mathcal{F})$, $\mathrm{K_p}(\mathcal{F})$, $\mathcal{X}$, f, dom, $OIns(\mathcal{F})$, $ODel(\mathcal{F})$, $WUpd(\mathcal{F})$)

POZOVI formiranje_ P^S_{ref} ($\mathcal{G}$, $OIns(\mathcal{F})$, $ODel(\mathcal{F})$, $WUpd(\mathcal{F})$, $P^S_{ref}(\mathcal{F})$)

POSTAVI $P^S_a \leftarrow P^S_u \cup P^S_{ref}$

POSTAVI $\mathrm{K_p} \leftarrow \left[\bigcup_{\mathrm{R_i} \in P^S_a} \mathrm{K^p_i} \right]$

POSTAVI $P_u \leftarrow \emptyset$

RADI eliminacija za ($\forall\, \mathrm{R_i} \in P^S_u$)

POSTAVI $P_u \leftarrow P_u \cup \{(\mathrm{R_i} \cap (W \cup \mathrm{K_p})), (\mathrm{K^p_i} \cup \{\mathrm{K_i} \in K_i \mid \mathrm{K_i} \subseteq W\})\}$

KRAJ RADI eliminacija

POSTAVI $P_a \leftarrow P_u$

RADI eliminacija_II za ($\forall\, \mathrm{R_i} \in P^S_a \setminus P^S_u$)

POSTAVI $P_a \leftarrow P_a \cup \{(\mathrm{R_i} \cap (W \cup \mathrm{K^p_i})), \{\mathrm{K^p_i}\}\}$

KRAJ RADI eliminacija_II

KRAJ PROCESA generisanje_ P_a

Slika A.11. *Algoritam* eliminacije *suvišnih obilježja iz podšeme i generisanje skupova* $P_u(\mathcal{F})$ *i* $P_a(\mathcal{F})$.

PROCES redukcija_ P^S_{min} (**U** (P^S_{min}, $\tilde{\mathcal{G}}$) , **I** ($P^S_{min_R}$) , **UI** ( ))

(*

U: Skup P^S_{min} =$\{ R_1, ... , R_k \}$ i tranzitivni graf zatvaranja $\tilde{\mathcal{G}}$.

I: Skup $P^S_{min_R} = \{(R_i, K_i) \in P^S_{min} | \neg(\exists R_j \in P^S_{min} \setminus \{R_i\})(R_i \subseteq (R_j)^+_\Gamma)\}$.

*)

POČETAK PROCESA redukcija_ P^S_{min}

POSTAVI $P^S_{min_R} \leftarrow P^S_{min}$

RADI izbor_R za ($\forall R_i \in P^S_{min_R}$)

RADI upoređivanje za ($\forall R_j \in P^S_{min} \setminus \{R_i\}$)

AKO JE $R_j \subseteq \left[\bigcup_{R_l \in \tilde{\Pi}(R_i)} R_l \right]$ **TADA** (* $R_i \subseteq (R_j)^+_\Gamma$ *)

POSTAVI $P^S_{min_R} \leftarrow P^S_{min_R} \setminus \{R_j\}$

INAČE

KRAJ AKO

KRAJ RADI upoređivanje

KRAJ RADI izbor_R

KRAJ PROCESA redukcija_ P^S_{min}

Slika A.12. *Algoritam redukcije skupa minimalnih čvorova podšeme*

PROCES proširenje_skupa_obilježja (**U** (U, $W(\mathcal{F})$, f) , **I** ($W_{ex}(\mathcal{F})$) , **UI** ( ))

(*

U: U – skup obilježja šema baze podataka, W – skup obilježja tipa forme.
f: U$\leftarrow P$(U) – funkcija izvođenja uloge obilježja.

I: $W_{ex}(\mathcal{F})$ – prošireni skup obilježja tipa forme.

*)

POČETAK PROCESA proširenje_skupa_obilježja

POSTAVI $W_{ex} \leftarrow W$

RADI prolaz_kroz_W ($\forall A \in W$) ($f(A) \neq \emptyset$)

POSTAVI $W_{ex} \leftarrow W_{ex} \cup (f(A) \cap U)$

POSTAVI $WPOM \leftarrow \{A' \in f(A) \mid f(A') \neq \emptyset\}$
RADI prolaz_kroz_f za ($\forall$ A'$\in WPOM$) **DOK JE** $WPOM \neq \emptyset$
POSTAVI $WPOM \leftarrow WPOM \setminus \{A'\}$
POSTAVI $WPOM \leftarrow WPOM \cup \{ A'' \in f(A') \mid f(A'') \neq \emptyset\}$
POSTAVI $W_{ex} \leftarrow W_{ex} \cup (f(A') \cap U)$
KRAJ RADI prolaz_kroz_f
KRAJ RADI prolaz_kroz_W
KRAJ PROCESA proširenje_skupa_obilježja

Slika A.13. *Algoritam formiranja proširenog skupa obilježja*

PROCES generisanje_P421 (**U** (U, $\mathcal{G}$, $\tilde{\mathcal{G}}$, M, $\tilde{T}(\mathcal{F})$, $T_y(J_y)$, $W(\mathcal{F})$), **I** (P_u), **UI** ( ))

(*

U: $T_y(J_i) = \{ Y_1, \ldots, Y_m \}$ skup definisan na osnovu zavisnosti spajanja $J_y = \bowtie \{ Y_1, \ldots, Y_m \}$ iz uslova (4.20) [22]

I: Skup šema relacija podšeme P_u datog tipa forme $\mathcal{F}$ iz klase $A_{\mathcal{F}}(u)$, prema izrazu (4.21) [22].

*)

POČETAK PROCESA generisanje_P421

POZOVI minimalni_čvorovi_podšeme (**U**($\mathcal{G}$, $\tilde{\mathcal{G}}$, M, $T_y(J_y)$, P^S_{min}')

POZOVI redukcija_P^S_{min} (P^S_{min}', $\tilde{\mathcal{G}}$, $P^S_{min_R}$)

POZOVI relevantni_čvorovi ($P^S_{min_R}$, $W(\mathcal{F})$, $\overline{T}(\mathcal{F})$, $\tilde{\mathcal{G}}$, R_c, Ω)

POZOVI generisanje_P^S_{pod} ($\mathcal{G}$, $\tilde{\mathcal{G}}$, $P^S_{min_R}$, R_c, Ω, P^S_{pod}')

POSTAVI $P^S_u \leftarrow P^S_{min_R} \cup P^S_{pod}$

POSTAVI $K_p \leftarrow \left[\bigcup_{R_i \in P^S_u} K^p_i \right]$

POSTAVI proširenje_skupa_obilježja (U, $W(\mathcal{F})$, f, W_{ex})

POSTAVI $P^S_u \leftarrow \emptyset$

RADI eliminacija ($\forall$ $R_i \in P_u^S$)

POSTAVI $P_u \leftarrow P_u \cup \{(R_i \cap (W \cup K_p), \{K_i^p\})\}$

KRAJ RADI eliminacija

KRAJ PROCESA generisanje_P421

Slika A.14. *Algoritam generisanja skupa šema relacija podšeme* $P_u(\mathcal{F})$ *prema izrazu* (4.21) [22]

PROCES generisanje_P_{rek}^S (**U** (U, $K_p(\mathcal{F})$, S, f), **I** (P_{rek}^S) , **UI** ( ))

POČETAK PROCESA generisanje_P_{rek}^S

POSTAVI $P_{rek}^S \leftarrow \emptyset$

RADI izvedena_obilježja ($\forall A \in K_p(\mathcal{F}) \setminus U$)

POSTAVI $WPOM \leftarrow \{A' \in f(A) \mid f(A') \neq \emptyset\}$

RADI prolaz_kroz_f za ($\forall A' \in WPOM$) **DOK JE** $WPOM \neq \emptyset$

POSTAVI $WPOM \leftarrow WPOM \setminus \{A'\}$

POSTAVI $WPOM \leftarrow WPOM \cup \{A'' \in f(A') \mid f(A'') \neq \emptyset\}$

AKO JE ($\exists R_i \in S$)($\exists K_i \in K_i$)($f(A') \cap U = K_i$) **TADA**

POSTAVI $P_{rek}^S \leftarrow P_{rek}^S \cup \{(R_i, K_i)\}$

INAČE

KRAJ AKO

KRAJ RADI prolaz_kroz_f

KRAJ RADI izvedena_obilježja

KRAJ PROCESA generisanje_P_{rek}^S

Slika A.15. *Algoritam za generisanje skupa šema relacija podšeme* $P_{rek}^S(\mathcal{F})$

PROCES generisanje_P425 (**U** (U, $\mathcal{G}$, $\tilde{\mathcal{G}}$, M, $\overline{\mathrm{T}}^{r}(\mathcal{F})$, $\mathrm{T}^{r}(\mathcal{F})$, $W(\mathcal{F})$, $\mathrm{K_p}(\mathcal{F})$) ,

I (P_u) , **UI** ( ))

(*

U: $\mathrm{T}^{r}(\mathcal{F})$ i $\overline{\mathrm{T}}^{r}(\mathcal{F})$ – skupovi definisani prema izrazu (4.23) [22]

I: Skup šema relacija podšeme P_u prema izrazu (4.25) [22]

*)

POČETAK PROCESA generisanje_P425

POZOVI minimalni_čvorovi_podšeme (**U**($\mathcal{G}$, $\tilde{\mathcal{G}}$, M, $\mathrm{T}^{r}(\mathcal{F})$, P^{sr}_{min})

POZOVI redukcija_ P^{s}_{min} (P^{sr}_{min}, $\tilde{\mathcal{G}}$, $P^{s}_{min_R}$)

POZOVI relevantni_čvorovi ($P^{s}_{min_R}$, $W(\mathcal{F})$, $\overline{\mathrm{T}}^{r}(\mathcal{F})$, $\tilde{\mathcal{G}}$, R_c, Ω)

POZOVI generisanje_ P^{s}_{pod} ($\mathcal{G}$, $\tilde{\mathcal{G}}$, $P^{s}_{min_R}$, R_c, Ω , P^{sr}_{pod})

POZOVI generisanje_ P^{s}_{rek} (U, $\mathrm{K_p}(\mathcal{F})$, S, f, P^{s}_{rek})

POSTAVI $P^{s}_{u} \leftarrow P^{s}_{min_R} \cup P^{sr}_{pod} \cup P^{s}_{rek}$

POSTAVI $\mathrm{K_p} \leftarrow \left[\bigcup_{\mathrm{R_i} \in P^{s}_{u}} \mathrm{K}^{\mathrm{p}}_{\mathrm{i}} \right]$

POZOVI proširenje_skupa_obilježja (U, $W(\mathcal{F})$, f, W_{ex})

POSTAVI $P_u \leftarrow \emptyset$

RADI eliminacija ($\forall\ \mathrm{R_i} \in P^{s}_{u}$)

POSTAVI $P_u \leftarrow P_u \cup \{(\mathrm{R_i} \cap (W_{ex} \cup \mathrm{K_p}), \{\mathrm{K}^{\mathrm{p}}_{\mathrm{i}}\})\}$

KRAJ RADI eliminacija

KRAJ PROCESA generisanje_P425

Slika A.16. *Algoritam generisanja skupa šema relacija podšeme* $P_u(\mathcal{F})$ *prema izrazu* (4.25) [22]

PROCES generisanje_P427 (**U** (U, $\mathcal{G}$, $\tilde{\mathcal{G}}$, M, $\overline{\mathrm{T}}(\mathcal{F})$, $\mathrm{T}_y(J_y)$, $W(\mathcal{F})$, $\mathrm{K}_p(\mathcal{F})$),

I (P_u) , **UI** ( ))

(*

U: $\mathrm{T}_y(J_y) = \{\mathrm{Y}_1, \ldots, \mathrm{Y}_m\}$ skup definisan na osnovu zavisnosti spajanja $J_y = \bowtie\{\mathrm{Y}_1, \ldots, \mathrm{Y}_m\}$ iz uslova (4.26) [22]

I: Skup šema relacija podšeme P_u prema izrazu (4.27) [22]

*)

POČETAK PROCESA generisanje_P427

POZOVI minimalni_čvorovi_podšeme ($\mathcal{G}$, $\tilde{\mathcal{G}}$, M, $\mathrm{T}_y(J_y)$, P^{sr}_{min}')

POZOVI redukcija_P^{s}_{min} (P^{sr}_{min}', $\tilde{\mathcal{G}}$, $P^{s}_{min_R}$)

POZOVI relevantni_čvorovi ($P^{s}_{min_R}$, $W(\mathcal{F})$, $\overline{\mathrm{T}}(\mathcal{F})$, $\tilde{\mathcal{G}}$, R_c, Ω)

POZOVI generisanje_P^{s}_{pod} ($\mathcal{G}$, $\tilde{\mathcal{G}}$, $P^{s}_{min_R}$, R_c, Ω, P^{sr}_{pod}')

POZOVI generisanje_P^{s}_{rek} (U, $\mathrm{K}_p(\mathcal{F})$, S, f, P^{s}_{rek})

POSTAVI $P^{s}_{u} \leftarrow P^{s}_{min_R} \cup P^{sr}_{pod}{}' \cup P^{s}_{rek}$

POSTAVI $\mathrm{K}_p \leftarrow \left[\bigcup_{\mathrm{R}_i \in P^{s}_{u}} \mathrm{K}^{p}_{i} \right]$

POSTAVI proširenje_skupa_obilježja (U, $W(\mathcal{F})$, f, W_{ex})

POSTAVI $P_u \leftarrow \emptyset$

RADI eliminacija ($\forall\, \mathrm{R}_i \in P^{s}_{u}$)

POSTAVI $P_u \leftarrow P_u \cup \{(\mathrm{R}_i \cap (W_{ex} \cup \mathrm{K}_p), \{\mathrm{K}^{p}_{i}\})\}$

KRAJ RADI eliminacija

KRAJ PROCESA generisanje_P427

Slika A.17. *Algoritam generisanja skupa šema relacija podšeme $P_u(\mathcal{F})$ prema izrazu* (4.27) [22]

PROCES test_zatvaranja_nosecih_grupa (**U** (T($\mathcal{F}$), $\tilde{\mathcal{G}}$), **I** (*ind*), **UI** ())
POČETAK PROCESA test_zatvaranja_nosecih_grupa
POSTAVI $ind \leftarrow true$
RADI izbor_X za ($\forall X_i \in T(\mathcal{F})$) **DOK JE** *ind*
POSTAVI $ind1 \leftarrow false$
RADI test za ($\forall R_j \in S$) **DOK JE** $\neg ind1$

$$\textbf{AKO JE } X_i \subseteq \left[\bigcup_{R_l \in \tilde{\Pi}(R_j)} R_l \right] \textbf{ TADA} \quad (* \quad X_i \subseteq (R_j)^{+}_{\Gamma} \quad *)$$

POSTAVI $ind1 \leftarrow true$
INAČE
KRAJ AKO
KRAJ RADI test
AKO JE $\neg ind1$ **TADA**
POSTAVI $ind \leftarrow false$ (* test je neuspješan *)
INAČE
KRAJ AKO
KRAJ RADI izbor_X
KRAJ PROCESA test_zatvaranja_nosecih_grupa

Slika A.18. *Algoritam testa zatvaranja nosećih grupa obilježja* (*uslov* 4.18, *odnosno* 4.24)[22]

PROCES test_spojivosti (**U** (T ($\mathcal{F}$), *SJ*), **I** ($T_y(J_y)$), **UI** ( ))
(*
I: $ind \in \{true, false\}$, $T_y(J_y) = \{Y_1, \ldots, Y_m\}$ skup definisan na osnovu zavisnosti spajanja, $J_y = \bowtie\{Y_1, \ldots, Y_m\}$, u slučaju da je test pozitivan.
*)
POČETAK PROCESA test_spojivosti
POSTAVI $ind \leftarrow false$ (* inicijalno se test smatra neuspješnim *)
POSTAVI $T_y(J_y) \leftarrow \emptyset$
RADI testiranje za ($\forall J_y = \bowtie\{Y_1, \ldots, Y_m\} \in SJ$) **DOK JE** $\neg ind$

$$\textbf{AKO JE } \left(\bigcup_{i=1}^{k} X_i = \bigcup_{i=1}^{m} Y_i \right) \textbf{ TADA}$$

POSTAVI *ind*1 ← *true*
RADI ispit_sadržavanja za ($\forall$ $Y_i \in \{ Y_1, \ldots, Y_m \}$) **DOK JE** *ind*1
AKO JE ($\exists$ $X_j \in T(\mathcal{F})$)($Y_i \subseteq X_j$) **TADA**
INAČE
POSTAVI *ind*1 ← *false*
KRAJ AKO
KRAJ RADI ispit_sadržavanja
AKO JE *ind*1 **TADA**
POSTAVI *ind* ← *true* (* test je uspješan *)
POSTAVI $T_y(J_y) \leftarrow \{ Y_1, \ldots, Y_m \}$
INAČE
KRAJ AKO
INAČE
KRAJ AKO
KRAJ RADI testiranje
KRAJ PROCESA test_spojivosti

Slika A.19. *Algoritam testa spojivosti* (*uslov* 4.20, *odnosno* 4.26) [22]

PROCES transformacija_T (**U** (U, *dom*, *f*, $K_p(\mathcal{F})$, $T(\mathcal{F})$, $\overline{T}(\mathcal{F})$),

I (*ind*, $T^r(\mathcal{F})$, $\overline{T}^r(\mathcal{F})$) , **UI** ( ))

(*
U: *dom* – domenska funkcija $f: U \leftarrow P(U)$ – funkcija izvođenja uloge obilježja.
$K_p(\mathcal{F})$ – skup primarnih obilježja tipa forme.
$T(\mathcal{F})$ – skup nosećih grupa obilježja.
$\overline{T}(\mathcal{F}) = \{ X_i \subseteq W \mid (\exists O_i \in O)(X_i \in \mathcal{X}(O_i)) \}$, $\mathcal{X}(O_i)$ – skup unija po jednog ključa od korijenskog do datog tipa komponente.

I: $T^r(\mathcal{F})$ – transformisani skup $T(\mathcal{F})$, prema izrazu (4.23) [22]
$\overline{T}^r(\mathcal{F})$ – transformisani skup $\overline{T}(\mathcal{F})$, prema izrazu (4.23) [22]
*)
POČETAK PROCESA transformacija_T
POSTAVI *ind* ← *true* (* inicijalno se test smatra uspješnim *)
RADI ispitivanje_obilježja za ($\forall A \in K_p(\mathcal{F}) \backslash U$) **DOK JE** *ind*
POSTAVI *w_ind* ← *false*
POSTAVI *WPOM* ← {A}

RADI ispitivanje_*Wrek* za ($\forall A^r \in WPOM$) **DOK JE** $\neg w\_ind$

AKO JE ($\forall A_B \in f(A^r) \cap U)(dom(A_B) = dom(A)$) **TADA**

POSTAVI $w\_ind \leftarrow true$

INAČE

POSTAVI $WPOM \leftarrow (WPOM \setminus \{A^r\}) \cup \{A' \in f(A^r) \mid f(A') \neq \emptyset\}$

KRAJ AKO

KRAJ RADI ispitivanje_*Wrek*

AKO JE $\neg w\_ind$ **TADA**

POSTAVI $ind \leftarrow false$

INAČE

KRAJ AKO

KRAJ RADI ispitivanje_obilježja

AKO JE *ind* **TADA**

POSTAVI $T^r \leftarrow \emptyset$,

POSTAVI $\overline{T}^r \leftarrow \emptyset$

RADI transformisanje_X za ($\forall X_i \in \overline{T}$)

POSTAVI $X_i^r \leftarrow (X_i \cap U)$

RADI prolaz_X za ($\forall A \in X_i \setminus U$)

POSTAVI $X_i^r \leftarrow X_i^r \cup (f(A) \cap U)$

KRAJ RADI prolaz_X

POSTAVI $\overline{T}^r \leftarrow \overline{T}^r \cup \{X_i^r\}$

AKO JE ($\forall X_i \in T$) **TADA**

POSTAVI $T^r \leftarrow T^r \cup \{X_i^r\}$

INAČE

KRAJ AKO

KRAJ RADI transformisanje_X

INAČE

KRAJ AKO

KRAJ PROCESA transformacija_T

Slika A.20. *Algoritam testa uslova* (4.22) [22] *i transformacija skupa nosećih grupa obilježja*

PROCES test_uslova_4.30 (**U** (NF ($\mathcal{F}$), Γ) , **I** (*ind*, *IZV*) , **UI** ( ))
(*
I: $IZV = \{(X, X') \mid X \rightarrow \emptyset \in NF(\mathcal{F}) \wedge X' \subset X \wedge X \rightarrow X' \in \Gamma^{+} \}$
*)
POČETAK PROCESA test_uslova_4.30
  POSTAVI $ind \leftarrow true$ (* inicijalno se smatra da nema konflikta *)
  POSTAVI $IZV \leftarrow \emptyset$
  RADI nefunkcionalne_zavisnosti za ($\forall X \rightarrow \emptyset \in NF(\mathcal{F})$)
    POSTAVI $LSX \leftarrow \emptyset$
    POSTAVI $X' \leftarrow X$
    RADI konflikti za ($\forall A \in X$)
      AKO JE $A \in (X' \setminus \{A\})^{+}_{\Gamma}$ **TADA**
        POSTAVI $LSX \leftarrow LSX \cup \{A\}$
        POSTAVI $X' \leftarrow X' \setminus \{A\}$
      INAČE
      KRAJ AKO
    KRAJ RADI konflikti
    AKO JE $LSX \neq \emptyset$ **TADA**
      POSTAVI $ind \leftarrow false$ (* test konflikta je neuspješan *)
      POSTAVI $IZV \leftarrow IZV \cup \{(X , X')\}$
    INAČE
    KRAJ AKO
  KRAJ RADI nefunkcionalne_zavisnosti
KRAJ PROCESA test_uslova_4.30

Slika A.21. *Algoritam testa konflikta funkcionalni / nefunkcionalni odnos* (4.30) [22]

PROCES test_uslova_4.19 (**U** (T($\mathcal{F}$), *SJ*) , **I** (*ind*) , **UI** ( ))
POČETAK PROCESA test_uslova_4.19
  POSTAVI $ind \leftarrow true$ (* inicijalno se test smatra neuspješnim *)
  AKO JE $| T(\mathcal{F}) | \neq 1$ **TADA**
    RADI testiranje_*J* za ($\forall J_i = \bowtie \{ Y_1, \ldots, Y_m \} \in SJ$) **DOK JE** $\neg ind$
      POSTAVI $ind1 \leftarrow true$
      RADI testiranje_X za ($\forall X_j \in T(\mathcal{F})$) **DOK JE** $ind1$
        AKO JE ($\exists Y_l \in \{ Y_1, \ldots, Y_m \}$) ($X_j = Y_l$) **TADA**
          POSTAVI $ind1 \leftarrow false$
        INAČE
        KRAJ AKO

```
        KRAJ RADI testiranje_X
        AKO JE ind1 TADA
          POSTAVI  ind ← true     ( *    uspješan test    * )
        INAČE
        KRAJ AKO
      KRAJ RADI testiranje_J
    INAČE
      POSTAVI  ind ← true         ( *    uspješan test    * )
    KRAJ AKO
KRAJ PROCESA test_uslova_4.19
```

Slika A.22. *Algoritam testa definisanosti zavisnosti spoja* (*uslov* 4.19) [22]

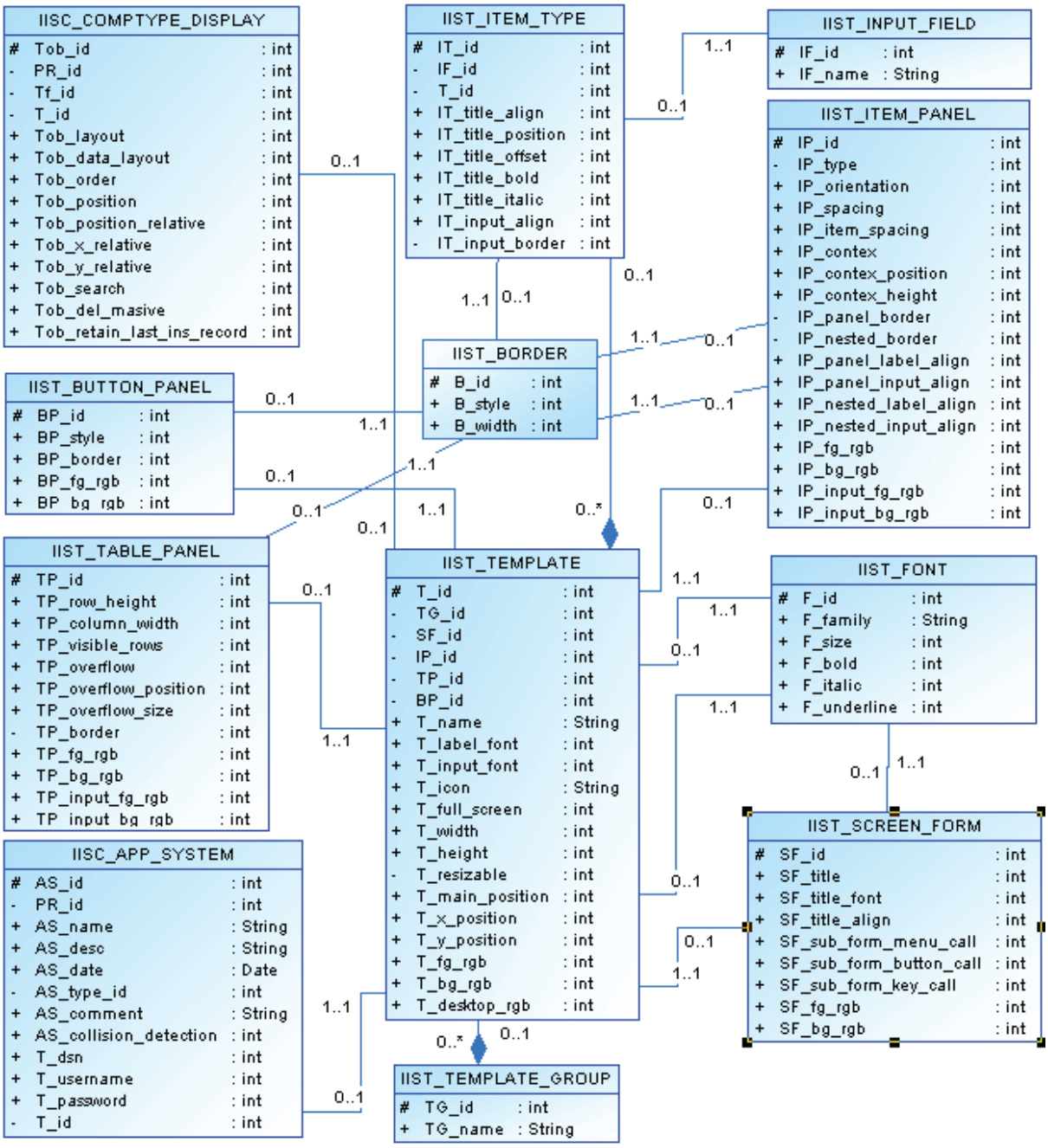
```
PROCES analiza_primjenljivosti ( U ((U, O), G , G~ , f, dom, W (F ), F(F ), NF(F ),
                                   T(F ), T̄ (F ), K_p (F )), I ( ind, IZV, J_y ) , UI ( ) )
( *
U:    (U, O), U – skup obilježja šeme baze podataka, O = { Γ , NΓ , SJ }
I:    ind ∈ {Z00, Z01, Z02, Z03, Z04, Z05} ( po tabeli odlučivanja datoj na slici 5.1.1.1.)
      IZV – izvještaj o konfliktima – po uslovu (4.30) [22]
      J_y – zavisnost spoja za koju važi uslov (4.20), tj. (4.26) [22]
* )
POČETAK PROCESA analiza_primjenljivosti
  AKO JE (∀ A∈ W \ U)( f (A) ≠ Ø ) ∧ ( Γ |= F(F )|_U ) TADA
   ( *  uslovi (4.16) i (4.17) [22]  * )
    AKO JE K_p (F )\U = Ø TADA                ( *   uslov (4.28) [22]  * )

      POZOVI test_zatvaranja_nosecih_grupa ( T (F ) , G~ , ind1 )
      AKO JE ind1 TADA                        ( *   uslov (4.18) [22]  * )
        POZOVI test_uslova_4.19 ( T(F ), SJ, ind1)
        AKO JE ¬ind1 ∧ | T(F ) | ≠ 1 TADA
          POSTAVI  SJ ← SJ ∪ {J(F )}          ( *   ¬ uslov (4.19) [22]  * )
        INAČE                                 ( *   uslov (4.19) [22]  * )
        KRAJ AKO
        POZOVI test_uslova_4.30 (NF(F ), Γ , ind1, IZV )
        AKO JE ind1 TADA                      ( *   uslov (4.30) [22]  * )
          AKO JE (∀ A∈ W ) ( f (A) ≠ Ø ⇒ f (A) ⊆ W ) TADA
            POSTAVI  ind ← Z03                ( *   uslov (4.29) [22]  * )
          INAČE
```

```
            POSTAVI ind ← Z04                    ( *  ¬ uslov (4.29) [22]  * )
          KRAJ AKO
        INAČE                                    ( *  ¬ uslov (4.30) [22]  * )
          POSTAVI ind ← Z04
        KRAJ AKO
      INAČE                                      ( *  ¬ uslov (4.18) [22]  * )
        POZOVI test_uslova_4.30 ( T(F), SJ, ind1, T_y(J_y))
        AKO JE ind1 TADA                         ( *  uslov (4.20) [22]  * )
          POSTAVI ind ← Z05
        INAČE                                    ( *  ¬ uslov (4.20) [22]  * )
          POSTAVI ind ← Z00
        KRAJ AKO
      KRAJ AKO
    INAČE                                        ( *  uslov (4.28) [22]  * )
      POZOVI transformacija_T (U, dom, f, K_P(F), T(F), T̄(F), ind1,
                                                          T^r(F), T̄^r(F))
      AKO JE ind1 TADA                           ( *  uslov (4.22) [22]  * )
        POZOVI test_zatvaranja_nosecih_grupa ( T^r(F), G̃, ind1 )
        AKO JE ind1 TADA                         ( *  uslov (4.24) [22]  * )
          POSTAVI ind ← Z02
        INAČE                                    ( *  ¬ uslov (4.24) [22]  * )
          POZOVI test_spojivosti (T^r(F), SJ, ind1, T_y(J_y))
          AKO JE ind1 TADA                       ( *  uslov (4.26) [22]  * )
            POSTAVI ind ← Z01
          INAČE                                  ( *  ¬ uslov (4.26) [22]  * )
            POSTAVI ind ← Z00
          KRAJ AKO
        KRAJ AKO
      INAČE                                      ( *  ¬ uslov (4.20) [22]  * )
        POSTAVI ind ← Z00
      KRAJ AKO
    KRAJ AKO
  INAČE                                  ( *  ¬ uslov (4.16) ∨ ¬ uslov (4.17) [22]  * )
    POSTAVI ind ← Z00
  KRAJ AKO
KRAJ PROCESA analiza_primjenljivosti
```

Slika A.23. *Algoritam analize primjenljivosti i načina upotrebe tipa forme*

Prilog B. Specifikacija repozitorijuma IIS*Studio

Na slici B.1. prikazan je izvod iz specifikacije repozitorijuma koji se odnosi na strukturu šablona korisničkog interfejsa.



Slika B.1. *Specifikacija repozitorijuma koja se odnosi na šablone korisničkog interfejsa*

IIST_BORDER	
B_id	ID specifikacije formata ivice.
B_style	Izabrani tip ivice.
B_width	Širina ivica.
Ključevi	
1. [B_id]	PK

IIST_BUTTON_PANEL	
BP_id	ID specifikacije šablona dugmadi.
BP_style	Indikator tipa prikaza dugmadi. Moguće vrijednosti su: • 1 – prikaz je grafički, • 0 – prikaz je tekstualni.
BP_border	ID specifikacije formata ivica dugmadi.
BP_fg_rgb	RGB vrijednost boje fonta dugmadi.
BP_bg_rgb	RGB vrijednost boje pozadine dugmadi.
Ključevi	
1. [BP_id]	PK
Međurelaciona ograničenja	
IIST_BUTTON_PANEL [BP_border] ⊆ IIST_BORDER [BP_id]	

IIST_FONT	
F_id	ID specifikacije fonta.
F_family	Izabrani tip fonta.
F_size	Veličina fonta.
F_bold	Indikator *bold* formata fonta. Moguće vrijednosti su: • 0 – font ne treba biti prikazan u *bold* formatu, • 1 – font treba biti prikazan u *bold* formatu.
F_italic	Indikator *italic* formata fonta. Moguće vrijednosti su: • 0 – font ne treba biti prikazan u *italic* formatu, • 1 – font treba biti prikazan u *italic* formatu.

F_underline	Indikator koji pokazuje da li je font prikazan u podvučenom formatu. Moguće vrijednosti su: • 0 – font ne treba biti prikazan u podvučenom formatu, • 1 – font treba biti prikazan u podvučenom formatu.
Ključevi	
1. [F_id]	PK

IIST_INPUT_FIELD	
IF_id	ID vrste polja za prikaz i ažuriranje podataka.
IF_name	Naziv vrste polja za prikaz i ažuriranje podataka.
Ključevi	
1. [IF_id]	PK

IIST_ITEM_PANEL	
IP_id	ID specifikacije šablona panela.
IP_type	Tip panela. Moguće vrijednosti su: • 0 – normalni prikaz panela, • 1 – prikaz panela sa jezičcima.
IP_orientation	Raspored panela u slučaju da IP_type ima vrijednost 0. Moguće vrijednosti su: • 0 – paneli su prikazani jedan ispod drugog, • 1 – paneli su prikazani po dva u redu, • 2 – paneli su prikazani po tri u redu.
IP_spacing	Rastojanje između panela.
IP_item_spacing	Rastojanje između polja unutar panela.
IP_contex	Indikator da li se kontekstna grupa polja prikazuje u okviru posebne kontekstne tabele, u slučaju da je vrijednost IP_type 1. Moguće vrijednosti su: • 0 – kontekstna tabela će biti prikazana, • 1 – kontekstna tabela neće biti prikazana.
IP_contex_position	Pozicija kontekstne tabele. Moguće vrijednosti su: • 0 – lijevo od panela,

	• 1 – iznad panela.
IP_contex_height	Visina kontekstne tabele, u slučaju da je vrijednost IP_contex_position 1.
IP_panel_border	ID specifikacije formata ivica panela.
IP_nested_border	ID specifikacije formata ivica ugnježdenih panela.
IP_panel_label_align	Poravnanje naziva polja panela. Moguće vrijednosti su: • 0 – svi nazivi su poravnati po lijevoj ivici, • 1 – svi nazivi su poravnati po desnoj ivici, • 2 – poravnanje nije specificirano.
IP_panel_input_align	Poravnanje polja panela. Moguće vrijednosti su: • 0 – sva polja su poravnata po lijevoj ivici, • 1 – sva polja su poravnata po desnoj ivici, • 2 – poravnanje nije specificirano.
IP_nested_label_align	Poravnanje naziva polja ugnježdenih panela. Moguće vrijednosti su: • 0 – svi nazivi su poravnati po lijevoj ivici, • 1 – svi nazivi su poravnati po desnoj ivici, • 2 – poravnanje nije specificirano.
IP_nested_input_align	Poravnanje polja ugnježdenih panela. Moguće vrijednosti su: • 0 – sva polja su poravnata po lijevoj ivici, • 1 – sva polja su poravnata po desnoj ivici, • 2 – poravnanje nije specificirano.
IP_fg_rgb	RGB vrijednost boje fonta panela.
IP_bg_rgb	RGB vrijednost boje pozadine panela.
IP_input_fg_rgb	RGB vrijednost boje fonta polja panela.
IP_input_bg_rgb	RGB vrijednost boje pozadine polja panela.
Ključevi	
1.	[IP_id] PK
Međurelaciona ograničenja	
IIST_ITEM_PANEL [IP_panel_border] ⊆ IIST_BORDER [B_id]	
IIST_ITEM_PANEL [IP_nested_border] ⊆ IIST_BORDER [B_id]	

IIST_ITEM_TYPE	
IT_id	ID specifikacije šablona polja za prikaz i ažuriranje podataka.
IF_id	ID vrste polja za prikaz i ažuriranje podataka.
T_id	ID specifikacije šablona korisničkog interfejsa.
T_title_align	Poravnanje naziva polja za prikaz i ažuriranje podataka. Moguće vrijednosti su: • 0 – lijevo poravnanje, • 1 – desno poravnanje, • 2 – centralno poravnanje.
T_title_position	Položaj naslova u odnosu na polja za prikaz i ažuriranje podataka. Moguće vrijednosti su: • 0 – lijevo, • 1 – iznad, • 2 – desno, • 3 – ispod.
T_title_offset	Rastojanje naslova od polja za prikaz i ažuriranje podataka.
T_title_bold	Indikator *bold* formata fonta naslova polja za prikaz i ažuriranje podataka. Moguće vrijednosti su: • 0 – font ne treba biti prikazan u *bold* formatu, • 1 – font treba biti prikazan u *bold* formatu.
T_title_italic	Indikator *italic* formata fonta polja za prikaz i ažuriranje podataka. Moguće vrijednosti su: • 0 – font ne treba biti prikazan u *italic* formatu, • 1 – font treba biti prikazan u *italic* formatu.
T_input_align	Poravnanje teksta u polju za prikaz i ažuriranje podataka. Moguće vrijednosti su: • 0 – lijevo poravnanje, • 1 – desno poravnanje, • 2 – centralno poravnanje.
IT_input_border	ID specifikacije formata ivica polja za prikaz i ažuriranje podataka.
Ključevi	
1. [IT_id]	PK
Međurelaciona ograničenja	
IIST_ITEM_TYPE [IF_id] ⊆ IIST_INPUT_FIELD [IF_id]	

IIST_ITEM_TYPE [T_id] ⊆ IIST_TEMPLATE [T_id]
IIST_ITEM_TYPE [IT_input_border] ⊆ IIST_BORDER [B_id]

IIST_SCREEN_FORM		
SF_id		ID specifikacije ekranskih formi.
SF_title		Indikator da li se prikazuje naslov na ekranskim formama ili ne. Moguće vrijednosti su: • 0 – naslov se ne prikazuje, • 1 – naslov se prikazuje.
SF_title_font		ID specifikacije fonta naslova ekranske forme.
SF_title_align		Poravnanje naslova ekranske forme. Moguće vrijednosti su: • 0 – lijevo poravnanje, • 1 – desno poravnanje, • 2 – centralno poravnanje.
SF_sub_form_menu_call		Indikator da li je pristup vezanim ekranskim formama omogućen kroz meni. Moguće vrijednosti su: • 0 – nije omogućen pristup kroz meni, • 1 – omogućen je pristup kroz meni.
SF_sub_form_button_call		Indikator da li je pristup vezanim ekranskim formama omogućen preko *toolbar*-a. Moguće vrijednosti su: • 0 – nije omogućen pristup preko *toolbar*-a, • 1 – omogućen je pristup preko *toolbar*-a.
SF_sub_form_key_call		Indikator da li je pristup vezanim ekranskim formama omogućen pritiskom na odgovarajuću dugmad sa tastature. Moguće vrijednosti su: • 0 – nije omogućen pristup preko tastature, • 1 – omogućen je pristup preko tastature.
SF_fg_rgb		RGB vrijednost boje fonta ekranske forme.
SF_bg_rgb		RGB vrijednost boje pozadine ekranske forme.
Ključevi		
1.	[SF_id]	PK

IIST_TABLE_PANEL	
TP_id	ID specifikacije šablona tabela.
TP_row_height	Visina reda u tabeli.
TP_column_ width	Širina kolone u tabeli.
TP_visible_rows	Visina tabele specificirana brojem vidljivih redova u tabeli.
TP_overflow	Indikator koji ukazuje na to da li se *overflow* grupa polja prikazuje u posebnoj *overflow* oblasti ili ne. Moguće vrijednosti su: • 0 – *overflow* grupa polja se ne prikazuje u posebnoj overflow oblasti, • 1 – *overflow* grupa polja se prikazuje u posebnoj overflow oblasti.
TP_overflow_position	Položaj *overflow* oblasti. Moguće vrijednosti su: • 0 – položaj desno od tabele, • 1 – položaj ispod tabele.
TP_overflow_size	Veličina *overflow* oblasti. U slučaju da TP_overflow_position ima vrijednost 0, TP_overflow_size predstavlja širinu u slučaju da TP_overflow_position ima vrijednost 1, TP_overflow_size predstavlja visinu *overflow* oblasti.
TP_border	ID specifikacije formata ivica tebele.
TP_fg_rgb	RGB vrijednost boje fonta tabele.
TP_bg_rgb	RGB vrijednost boje pozadine tabele.
TP_input_fg_rgb	RGB vrijednost boje fonta polja tabele.
TP_input_bg_rgb	RGB vrijednost boje pozadine polja tabele.
Ključevi	
1. [TP_id]	PK
Međurelaciona ograničenja	
IIST_TABLE_PANEL [TP_border] ⊆ IIST_BORDER [B_id]	

IIST_TEMPLATE	
T_id	ID specifikacije šablona korisničkog interfejsa.
TG_id	ID grupe šablona.
SF_id	ID specifikacije šablona ekranskih formi.
IP_id	ID specifikacije šablona panela.

TP_id	ID specifikacije šablona tabela.
BP_id	ID specifikacije šablona dugmadi.
T_name	Naziv specifikacije šablona korisničkog interfejsa.
T_label_font	ID specifikacije fonta naziva polja, dugmadi i menija.
T_input_font	ID specifikacije fonta polja za prikaz i ažuriranje podataka.
T_icon	Ikona glavnog prozora korisničkog interfejsa.
T_full_screen	Indikator koji ukazuje da li glavni prozor aplikacije ima *full screen* prikaz. Moguće vrijednosti su: • 0 – dimenzije glavnog prozora se posebno definiše, • 1 – glavni prozor se prikazuje u *full screen* formatu.
T_width	Širina glavog prozora aplikacije u slučaju da opcija T_full_screen ima vrijednost 0.
T_height	Visina glavog prozora aplikacije u slučaju da opcija T_full_screen ima vrijednost 0.
T_resizable	Indikator koji ukazuje da li glavni prozor aplikacije ima fiksirane dimenzije. Moguće vrijednosti su: • 0 – glavni prozor ima fiksirane dimenzije, • 1 – dimenzije glavnog prozora nisu fiksirane, tj. korisnik može da mijenja veličinu prozora.
T_main_position	Pozicija glavnog prozora aplikacije. Moguće vrijednosti su: • 0 – prozor je pozicioniran u gornjem lijevom uglu, • 1 – prozor je centralno pozicioniran u odnosu na desktop, • 2 – pozicija prozora se proizvoljno definiše.
T_x _position	Rastojanje gornjeg lijevog ugla glavnog prozora aplikacije od lijeve ivice ekrana u slučaju da je opcija T_position ima vrijednost 2.
T_y _position	Rastojanje gornjeg lijevog ugla glavnog prozora aplikacije od gornje ivice ekrana u slučaju da je opcija T_position ima vrijednost 2.
T_fg_rgb	RGB vrijednost boje fonta glavnog prozora aplikacije.

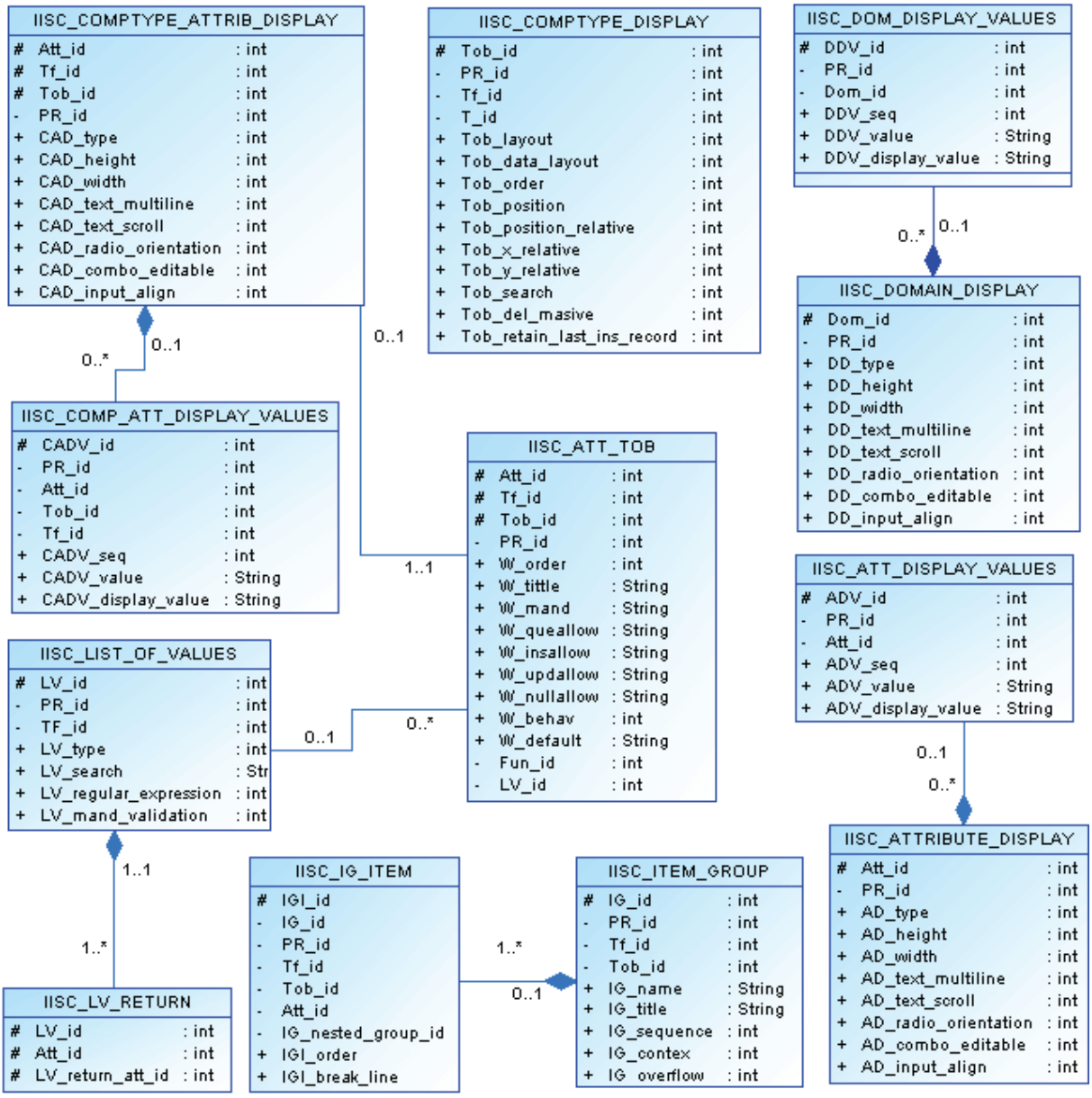
T_bg_rgb	RGB vrijednost boje pozadine glavnog prozora aplikacije.
T_desktop_rgb	RGB vrijednost boje desktopa glavnog prozora aplikacije.
Ključevi	
1. [T_id]	PK
Međurelaciona ograničenja	
IIST_TEMPLATE [TG_id] ⊆ IIST_TEMPLATE_GROUP [IG_id]	
IIST_TEMPLATE [SF_id] ⊆ IIST_SCREEN_FORM [SF_id]	
IIST_TEMPLATE [TP_id] ⊆ IIST_TABLE_PANEL [TP_id]	
IIST_TEMPLATE [IT_id] ⊆ IIST_ITEM_PANEL [IT_id]	
IIST_TEMPLATE [BP_id] ⊆ IIST_BUTTON_PANEL [BP_id]	
IIST_TEMPLATE [T_label_font] ⊆ IIST_FONT [F_id]	
IIST_TEMPLATE [T_input_font] ⊆ IIST_FONT [F_id]	

IIST_TEMPLATE_GROUP	
TG_id	ID grupe specifikacija šablona korisničkog interfejsa.
TG_name	Naziv grupe specifikacija šablona korisničkog interfejsa.
Ključevi	
1. [TG_id]	PK

IISC_APP_SYSTEM	
AS_id	ID aplikativnog sistema.
PR_id	ID projekta.
AS_name	Naziv aplikativnog sistema.
AS_desc	Opis aplikativnog sistema.
AS_date	Datum i vrijeme kreiranja aplikativnog sistema.
AS_type_id	ID tipa aplikativnog sistema.
AS_comment	Komentar.
AS_collision_detection	Označava do kog koraka je urađen algoritam usaglašavanja skupova ograničenja podšema sa šemom datog aplikativnog sistema i da li su detektovane kolizije ili upozorenja. Vrijednosti idu od -5 do 5:

	• 0 - nije urađen nijedan korak, • 1 - urađen algoritam za usaglašavanje skupova atributa, • 2 - urađen i algoritam za usaglašavanje skupova ograničenja NULL vrijednosti atributa, • 3 - urađen i algoritam za usaglašavanje skupova ograničenja jedinstvene vrijednosti atributa, • 4 - urađen i algoritam za usaglašavanje skupova ograničenja ključa, • 5 - urađen i algoritam za usaglašavanje skupova ograničenja referencijalnih integriteta, • negativne vrijednosti označavaju da su na tom koraku detektovane kolizije ili upozorenja.
AS_dsn	DSN ODBC konekcije na bazu podataka.
AS_username	Korisničko ime za pristup ODBC konekciji.
AS_password	Password za pristup ODBC konekciji.
T_id	ID specifikacije šablona korisničkog interfejsa.
Ključevi	
1. [AS_id]	PK
Međurelaciona ograničenja	
IISC_APP_SYSTEM [T_id] ⊆ IIST_TEMPLATE[T_id]	

Na slici B.2. prikazan je izvod iz specifikacije repozitorijuma koji se odnosi na vizuelne atribute definisane na tipovima komponenti i njihovim strukturnim elemantima, na grupe polja i liste vrijednosti.



Slika B.2. *Specifikacija repozitorijuma koja se odnosi na vizuelne atribute, grupe polja i liste vrijednosti*

IISC_DOMAIN_DISPLAY	
Dom_id	ID domena.
PR_id	ID projekta.
DD_type	ID tipa polja, odnosno programske komponente koja odgovara datom domenu.
DD_height	Visina polja.

DD_width	Širina polja.
DD_text_multiline	Indikator prikaza podataka u više redova u slučaju da DD_type ima vrijednost 0. Moguće vrijednosti su: • 0 – tekst se prikazuje u jednom redu, • 1 – tekst se prikazuje u više redova.
DD_text_scroll	Indikator prikaza klizača na programskoj komponenti u slučaju da DD_type ima vrijednost 0. Moguće vrijednosti su: • 0 – klizači se ne prikazuju, • 1 – klizači se uvijek prikazuju, • 2 – klizači se prikazuju po potrebi.
DD_radio_orientation	Indikator načina rasporeda opcija radio grupe u slučaju da DD_type ima vrijednost 1. Moguće vrijednosti su: • 0 – opcije su poređane horizontalno, • 1 – opcije su poređane vertikalno.
DD_combo_editable	Indikator editabilnosti padajuće liste u slučaju da DD_type ima vrijednost 3. Moguće vrijednosti su: • 0 – padajuće lista nije editabilna, • 1 – padajuće lista je editabilna.
Ključevi	
1. [Dom_id]	PK
Međurelaciona ograničenja	
IISC_DOMAIN_DISPLAY [PR_id] ⊆ IISC_PROJECT [PR_id]	
IISC_DOMAIN_DISPLAY [Dom_id] ⊆ IISC_DOMAIN [Dom_id]	

IISC_DOM_DISPLAY_VALUES	
DDV_id	ID specifikacije ponuđenih vrijednosti.
PR_id	ID projekta.
Dom_id	ID domena.
DDV_seq	Redni broj ponuđene vrijednosti.
DDV_value	Vrijednost koja će biti memorisana u bazi podataka.
DDV_display_value	Vrijednost koja je prikazana korisniku pri izboru jedne od ponuđenih vrijednosti.
Ključevi	
1. [DDV_id]	PK
Međurelaciona ograničenja	

IISC_DOM_DISPLAY_VALUES [PR_id] ⊆ IISC_PROJECT [PR_id]
IISC_DOM_DISPLAY_VALUES [Dom_id] ⊆ IISC_DOMAIN [Dom_id]

IISC_ATTRIBUTE_DISPLAY	
Att_id	ID atributa.
PR_id	ID projekta.
DD_type	ID tipa polja, odnosno programske komponente koja odgovara datom atributu.
DD_height	Visina polja.
DD_width	Širina polja.
DD_text_multiline	Indikator prikaza podataka u više redova u slučaju da DD_type ima vrijednost 0. Moguće vrijednosti su: • 0 – tekst se prikazuje u jednom redu, • 1 – tekst se prikazuje u više redova.
DD_text_scroll	Indikator prikaza klizača na programskoj komponenti u slučaju da DD_type ima vrijednost 0. Moguće vrijednosti su: • 0 – klizači se ne prikazuju, • 1 – klizači se uvijek prikazuju, • 2 – klizači se prikazuju po potrebi.
DD_radio_orientation	Indikator načina rasporeda opcija radio grupe u slučaju da DD_type ima vrijednost 1. Moguće vrijednosti su: • 0 – opcije su poređane horizontalno, • 1 – opcije su poređane vertikalno.
DD_combo_editable	Indikator editabilnosti padajuće liste u slučaju da DD_type ima vrijednost 3. Moguće vrijednosti su: • 0 – padajuće lista nije editabilna, • 1 – padajuće lista je editabilna.
Ključevi	
1. [Att_id]	PK
Međurelaciona ograničenja	
IISC_ATTRIBUTE_DISPLAY [PR_id] ⊆ IISC_PROJECT [PR_id]	
IISC_ATTRIBUTE_DISPLAY [Att_id] ⊆ IISC_ATTRIBUTE [Att_id]	

IISC_ATT_DISPLAY_VALUES	
ADV_id	ID specifikacije ponuđenih vrijednosti.
PR_id	ID projekta.

Att_id	ID atributa.
ADV_seq	Redni broj ponuđene vrijednosti.
ADV_value	Vrijednost koja će biti memorisana u bazi podataka.
ADV_display_value	Vrijednost koja je prikazana korisniku pri izboru jedne od ponuđenih vrijednosti.
Ključevi	
1. [ADV_id]	PK
Međurelaciona ograničenja	
IISC_ATT_DISPLAY_VALUES [PR_id] ⊆ IISC_PROJECT [PR_id]	
IISC_ATT_DISPLAY_VALUES [Att_id] ⊆ IISC_ATTRIBUTE [Att_id]	

IISC_COMPTYPE_ATTRIB_DISPLAY	
Att_id	ID atributa.
Tf_id	ID tipa forme.
Tob_id	ID tipa komponente.
PR_id	ID projekta.
CAD_type	ID tipa polja, odnosno programske komponente koja odgovara datom atributu na tipu komponente.
CAD_height	Visina polja.
CAD_width	Širina polja.
CAD_text_multiline	Indikator prikaza podataka u više redova u slučaju da DD_type ima vrijednost 0. Moguće vrijednosti su: • 0 – tekst se prikazuje u jednom redu, • 1 – tekst se prikazuje u više redova.
CAD_text_scroll	Indikator prikaza klizača na programskoj komponenti u slučaju da DD_type ima vrijednost 0. Moguće vrijednosti su: • 0 – klizači se ne prikazuju, • 1 – klizači se uvijek prikazuju, • 2 – klizači se prikazuju po potrebi.
CAD_radio_orientation	Indikator načina rasporeda opcija radio grupe u slučaju da DD_type ima vrijednost 1. Moguće vrijednosti su: • 0 – opcije su poređane horizontalno, • 1 – opcije su poređane vertikalno.
CAD_combo_editable	Indikator editabilnosti padajuće liste u slučaju da DD_type ima vrijednost 3. Moguće vrijednosti su:

	• 0 – padajuće lista nije editabilna, • 1 – padajuće lista je editabilna.
Ključevi	
1. [Att_id+Tf_id+Tob_id]	PK
Međurelaciona ograničenja	
IISC_COMPTYPE_ATTRIB_DISPLAY [PR_id] ⊆ IISC_PROJECT [PR_id]	
IISC_COMPTYPE_ATTRIB_DISPLAY [Att_id, Tf_id, Tob_id, PR_id] ⊆ IISC_ATT_TOB [Att_id, Tf_id, Tob_id, PR_id]	

IISC_COMP_ATT_DISPLAY_VALUES	
CADV_id	ID specifikacije ponuđenih vrijednosti.
PR_id	ID projekta.
Att_id	ID atributa.
Tf_id	ID tipa forme.
Tob_id	ID tipa komponente.
CADV_seq	Redni broj ponuđene vrijednosti.
CADV_value	Vrijednost koja će biti memorisana u bazi podataka.
CADV_display_value	Vrijednost koja je prikazana korisniku pri izboru jedne od ponuđenih vrijednosti.
Ključevi	
1. [CADV_id]	PK
Međurelaciona ograničenja	
IISC_COMP_ATT_DISPLAY_VALUES [PR_id] ⊆ IISC_PROJECT [PR_id]	
IISC_COMP_ATT_DISPLAY_VALUES [Att_id, Tf_id, Tob_id, PR_id] ⊆ IISC_ATT_TOB [Att_id, Tf_id, Tob_id, PR_id]	

IISC_LIST_OF_VALUES	
LV_id	ID liste vrijednosti.
PR_id	ID projekta.
Tf_id	ID tipa forme.
LV_type	Tip liste vrijednosti. Moguće vrijednosti su: • 0 – vrijednost polja se setuje samo kroz listu vrijednosti. • 1 – polje je editabilno i vrijednost je moguće ručno ukucati ili setovati kroz listu vrijednosti.
LV_search	Indikator da li se ukucana vrijednost koristi kao riječ za pretragu prilikom izlistavanja

	rekorda u listi vrijednosti, u slučaju da LV_type ima vrijednost 1. Moguće vrijednosti su: • 0 – ukucana vrijednost se ne koristi kao riječ za pretragu. • 1 – ukucana vrijednost se ne koristi kao riječ za pretragu.
LV_regular	U listi vrijednosti će biti prikazane samo one vrijednosti koje zadovoljavaju dati regularni izraz.
LV_mand_validation	Način validacije unešene vrijednosti u slučaju da LV_type ima vrijednost 1. Moguće vrijednosti su: • 0 – unešena vrijednost se ne validira. • 1 – unešena vrijednost se validira pri napuštanju polja, ispisuje se poruka o greški i nemoguće je napustiti polje dok se ne unese validna vrijednost.
Ključevi	
1.	[LV_id] PK
Međurelaciona ograničenja	
IISC_LIST_OF_VALUES [PR_id] ⊆ IISC_PROJECT [PR_id]	
IISC_LIST_OF_VALUES [Tf_id, PR_id] ⊆ IISC_FORM_TYPE [Tf_id, PR_id]	

IISC_LV_RETURN	
LV_id	ID pridruživanja liste vrijednosti atributu.
Att_id	ID atributa čija se vrijednost vraća pri izboru iz liste vrijednosti.
LV_return_att_id	ID atributa sa tipa komponente na koji se vraća vrijednost atributa liste vrijednosti sa ID-ijem Att_id.
Ključevi	
1.	[LV_id+Att_id] PK
Međurelaciona ograničenja	
IISC_LV_RETURN [Att_id] ⊆ IISC_ATTRIBUTE [Att_id]	
IISC_LV_RETURN [LV_return_att_id] ⊆ IISC_ATTRIBUTE [Att_id]	

IISC_ITEM_GROUP	
IG_id	ID grupe polja.

PR_id	ID projekta.
Tf_id	ID tipa forme.
Tob_id	ID tipa komponente.
IG_name	Naziv grupe polja.
IG_title	Naslov grupe polja.
IG_sequence	Redni broj grupe polja.
IG_contex	Indikator da li je grupa polja kontektsnog tipa. Moguće vrijednosti su: • 0 – grupa polja nije kontekstnog tipa, • 1 – grupa polja jeste kontekstnog tipa.
IG_overflow	Indikator da li je grupa polja *overflow* tipa. Moguće vrijednosti su: • 0 – grupa polja nije *overflow* tipa, • 1 – grupa polja jeste *overflow* tipa.
Ključevi	
1. [IG_id]	PK
Međurelaciona ograničenja	
IISC_ITEM_GROUP [PR_id] ⊆ IISC_PROJECT [PR_id]	
IISC_ITEM_GROUP [Tf_id, Tob_id, PR_id] ⊆ IISC_COMPONENT_TYPE_OBJECT_TYPE [Tf_id, Tob_id, PR_id]	

IISC_IG_ITEM	
IGI_id	ID polja.
IG_id	ID grupe polja.
PR_id	ID projekta.
Tf_id	ID tipa forme.
Tob_id	ID tipa komponente.
Att_id	ID atributa.
IG_nested_group_id	ID podgrupe polja.
IGI_order	Redni broj atributa ili podgrupe u datoj grupi polja.
IGI_breakline	Indikator relativnog položaja polja u odnosu na prethodno polje u grupi. Moguće vrijednosti su: • 0 – polje se prikazuje desno od prethodnog polja, • 1 – polje se prikazuje ispod prethodnog polja.
Ključevi	

1.	[IGI_id]	PK
Međurelaciona ograničenja		
IISC_IG_ITEM [PR_id] ⊆ IISC_PROJECT [PR_id]		
IISC_IG_ITEM [Att_id, Tf_id, Tob_id, PR_id] ⊆ IISC_ATT_TOB [Att_id, Tf_id, Tob_id, PR_id]		
IISC_IG_ITEM [IG_id] ⊆ IISC_ITEM_GROUP [IG_id]		
IISC_IG_ITEM [IG_nested_group_id] ⊆ IISC_IG_ITEM [IG_id]		

IISC_COMPTYPE_DISPLAY	
Tob_id	ID tipa komponente.
PR_id	ID projekta.
Tf_id	ID tipa forme.
T_id	ID šablona korisničkog interfejsa.
Tob_layout	Tip prikaza tipa komponente. Moguće vrijednosti su: • 0 – tip komponente će biti prikazan u okviru novog prozora, • 1 – tip komponente će biti prikazan u okviru prozora nadređenog tipa komponenete (*master-detail* forma).
Tob_data_layout	Tip prikaza podataka na tipu komponente. Moguće vrijednosti su: • 0 – prikaz u okviru panela, • 1 – tabelarni prikaz.
Tob_order	Redni broj tipa komponente na tipu forme.
Tob_position	Položaj tipa komponente u odnosu na prethodni tip komponente u slučaju da Tob_layout ima vrijednost 1. Moguće vrijednosti su: • 0 – prikazuje se desno od prethodnog tipa komponente, • 1 – prikazuje se ispod od prethodnog tipa komponente.
Tob_position_relative	Položaj ekranske forme tipa komponente u odnosu na prozor nadređenog tipa komponente u slučaju da Tob_layout ima vrijednost 0. Moguće vrijednosti su: • 0 – prikazuje se centrirano, • 1 – prikazuje se u lijevom gornjem uglu, • 2 – proizvoljno se definišu koordinate.
Tob_x_relative	X koordinata položaja tipa komponente u odnosu na nadređeni tip komponente u slučaju da Tob_position_relative ima vrijednost 2.

Tob_y_relative	Y koordinata položaja tipa komponente u odnosu na nadređeni tip komponente u slučaju da Tob_position_relative ima vrijednost 2.
Tob_search	Indikator opcije za pretragu na tipu komponente. Moguće vrijednosti su: • 0 – nije omogućena je pretraga na tipu komponente, • 1 – omogućena je pretraga na tipu komponente,
Tob_delete_masive	Indikator opcije brisanje više polja odjednom. Moguće vrijednosti su: • 0 – nije dozvoljeno brisanje više polja odjednom, • 1 – dozvoljeno je brisanje više polja odjednom.

Ključevi		
1.	[Tob_id]	PK

Međurelaciona ograničenja
IISC_COMPTYPE_DISPLAY [PR_id] ⊆ IISC_PROJECT [PR_id]
IISC_COMPTYPE_DISPLAY [Tf_id, Tob_id, PR_id] ⊆ IISC_COMPONENT_TYPE_OBJECT_TYPE [Tf_id, Tob_id, PR_id]
IISC_COMPONENT_TYPE_OBJECT_TYPE [T_id] ⊆ IIST_TEMPLATE [T_id]

Prilog C. UIML specifikacija prototipa korisničkog interfejsa

```
<?xml version="1.0" encoding="ISO-8859-2" ?>
- <uiml>
- <interface id="IISUIModeler">
- <structure>
- <part id="MainFrame" class="JFrame">
- <style>
  <property name="bounds">240,100,800,600</property>
  <property name="title">Application System</property>
  <property name="iconImage">IIST.gif</property>
  <property name="resizable">true</property>
  </style>
- <part id="mainMenubar" class="JMenuBar">
- <part id="mainFile" class="JMenu">
+ <style>
+ <part id="mainClose" class="JMenuItem">
  <part id="mainSeparate" class="JSeparator" />
+ <part id="mainQuit" class="JMenuItem">
   </part>
+ <part id="ApplicationformTypeMenuBar" class="JMenu">
+ <part id="mainHelp" class="JMenu">
  </part>
- <part id="dtp" class="JDesktopPane">
+ <style>
- <part id="1compType" class="JInternalFrame">
- <style>
  <property name="defaultCloseOperation">1</property>
  <property name="title">Form Type Table</property>
  <property name="visible">false</property>
  <property name="resizable">false</property>
  <property name="closable">true</property>
  <property name="frameIcon">IIST.gif</property>
  <property name="iconifiable">true</property>
  <property name="bounds">20,20,620,427</property>
  </style>
- <part id="1Title" class="JLabel">
- <style>
  <property name="horizontalAlignment">LEFT</property>
  <property name="verticalAlignment">BOTTOM</property>
  <property name="bounds">20,0,560,19</property>
  <property name="font">SansSerif.bold-bold-14</property>
  <property name="text">Component Type</property>
  </style>
  </part>
- <part id="1TableScroll" class="JScrollPane">
- <style>
  <property name="bounds">25,29,500,157</property>
  </style>
- <part id="1Table" class="JTable">
- <style>
  <property name="selectionMode">0</property>
  <property name="autoResizeMode">0</property>
- <property name="columnNames">
- <constant model="list">
  <constant value="Item 1" />
  <constant value="Item 2" />
  </constant>
  </property>
```

```
- <property name="content">
- <constant model="table.rowMajor">
- <constant>
  <constant value="Item 1 Value 11" />
  <constant value="Item 2 Value 21" />
  </constant>
…
- <constant>
  <constant value="Item 1 Value 120" />
  <constant value="Item 2 Value 220" />
  </constant>
  </constant>
  </property>
  </style>
  </part>
  </part>
- <part id="1OverflowScroll" class="JScrollPane">
- <style>
  <property name="bounds">25,186,500,102</property>
  </style>
- <part id="21itemGroup" class="JPanel">
- <style>
  <property name="bounds">0,0,192,60</property>
  <property name="preferredSize">192,60</property>
  <property name="name">Item Group 2</property>
  <property name="background">ccffcc</property>
  <property name="layout">null</property>
  </style>
- <part id="41InputLabel" class="JLabel">
- <style>
  <property name="bounds">12,12,42,11</property>
  <property name="horizontalAlignment">LEFT</property>
  <property name="text">Item 4</property>
  <property name="font">SansSerif.bold-bold-11</property>
  </style>
  </part>
- <part id="41InputFieldScrollPane" class="JScrollPane">
- <style>
  <property name="border">EtchedBorder</property>
  </style>
- <part id="41InputField" class="JTextArea">
- <style>
  <property name="text">Item 4 Value</property>
  <property name="editable">false</property>
  <property name="text">Item 4 Value</property>
  </style>
  </part>
- <style>
  <property name="bounds">54,12,150,60</property>
  </style>
  </part>
  </part>
  </part>
- <part id="1NavigationPanel" class="JPanel">
- <style>
  <property name="bounds">-15,312,640,35</property>
  <property name="opaque">false</property>
  </style>
- <part id="1FirstButton" class="JButton">
```

```
- <style>
  <property name="text">First</property>
  </style>
  </part>
- <part id="1PrevButton" class="JButton">
- <style>
  <property name="text">Prev</property>
  </style>
  </part>
- <part id="1Records" class="JLabel">
- <style>
  <property name="horizontalAlignment">CENTER</property>
  <property name="preferredSize">75,30</property>
  <property name="text">Record: 1/100</property>
  </style>
  </part>
- <part id="1NextButton" class="JButton">
- <style>
  <property name="text">Next</property>
  </style>
  </part>
- <part id="1LastButton" class="JButton">
- <style>
  <property name="text">Last</property>
  </style>
  </part>
  </part>
- <part id="1ButtonPanel" class="JPanel">
- <style>
  <property name="bounds">-15,347,640,35</property>
  <property name="opaque">false</property>
  </style>
- <part id="1NewButton" class="JButton">
- <style>
  <property name="text">New</property>
  </style>
- <behavior>
- <rule>
- <condition>
  <event class="actionPerformed" />
  </condition>
- <action>
  <property part-name="1compTypeInput" name="visible">true</property>
  </action>
  </rule>
  </behavior>
  </part>
- <part id="1DuplicateButton" class="JButton">
- <style>
  <property name="text">Duplicate</property>
  <property name="visible">true</property>
  </style>
- <behavior>
- <rule>
- <condition>
  <event class="actionPerformed" />
  </condition>
- <action>
  <property part-name="1compTypeInput" name="visible">true</property>
  </action>
  </rule>
```

```
  </behavior>
  </part>
- <part id="1DeleteButton" class="JButton">
- <style>
  <property name="text">Delete</property>
  </style>
  </part>
- <part id="1MasiveDeleteButton" class="JButton">
- <style>
  <property name="text">Delete All</property>
  </style>
  </part>
- <part id="1ApplyButton" class="JButton">
- <style>
  <property name="text">Apply</property>
  </style>
  </part>
- <part id="1OKButton" class="JButton">
- <style>
  <property name="text">OK</property>
  </style>
  </part>
- <part id="1Cancel" class="JButton">
- <style>
  <property name="text">Cancel</property>
  </style>
- <behavior>
- <rule>
- <condition>
  <event class="actionPerformed" />
  </condition>
- <action>
  <property part-name="1compType" name="visible">false</property>
  </action>
  </rule>
  </behavior>
  </part>
  </part>
  </part>
+ <part id="2compType" class="JInternalFrame">
+ <part id="3compType" class="JInternalFrame">
+ <part id="4compType" class="JInternalFrame">
+ <part id="5compType" class="JInternalFrame">
+ <part id="6compType" class="JInternalFrame">
+ <part id="MenuForm" class="JInternalFrame">
  </part>
+ <part id="51compType" class="JDialog">
+ <part id="1compTypeInput" class="JDialog">
+ <part id="51compTypeInput" class="JDialog">
+ <part id="7compTypeInput" class="JDialog">
+ <part id="4compTypeSearch" class="JDialog">
+ <part id="About" class="JDialog">
  </part>
  </structure>
- <style>
  <property part-class="JFrame" name="defaultCloseOperation">2</property>
  <property part-class="JFrame" name="font">SansSerif.bold-bold-
11</property>
  <property part-class="JDialog" name="layout">null</property>
```

```
  <property part-class="JDialog" name="font">SansSerif.bold-bold-
11</property>
  <property part-class="JDialog" name="background">ccffcc</property>
  <property part-class="JDialog" name="foreground">006600</property>
  <property part-class="JInternalFrame" name="layout">null</property>
  <property part-class="JInternalFrame" name="font">SansSerif.bold-bold-
11</property>
  <property part-class="JInternalFrame"
name="background">ccffcc</property>
  <property part-class="JInternalFrame"
name="foreground">006600</property>
  <property part-class="JMenu" name="background">ccccff</property>
  <property part-class="JMenu" name="foreground">6600cc</property>
  <property part-class="JMenu" name="font">SansSerif.bold-bold-
11</property>
  <property part-class="JMenuBar" name="background">ccccff</property>
  <property part-class="JMenuBar" name="foreground">6600cc</property>
  <property part-class="JMenuItem" name="background">ccccff</property>
  <property part-class="JMenuItem" name="foreground">6600cc</property>
  <property part-class="JMenuItem" name="font">SansSerif.bold-bold-
11</property>
  <property part-class="JTabbedPane" name="background">ccffcc</property>
  <property part-class="JTabbedPane" name="foreground">003399</property>
  <property part-class="JTabbedPane" name="font">SansSerif.bold-bold-
11</property>
  <property part-class="JPanel" name="background">ccffcc</property>
  <property part-class="JPanel" name="foreground">003399</property>
  <property part-class="JPanel" name="font">SansSerif.bold-bold-
11</property>
  <property part-class="JButton" name="background">cc99ff</property>
  <property part-class="JButton" name="foreground">6600cc</property>
  <property part-class="JButton" name="font">SansSerif.bold-bold-
11</property>
  <property part-class="JButton"
name="contentAreaFilled">false</property>
  <property part-class="JButton" name="opaque">true</property>
  <property part-class="JButton"
name="border">BevelBorder,raised</property>
  <property part-class="JButton" name="preferredSize">75,30</property>
  <property part-class="JLabel" name="opaque">false</property>
  <property part-class="JLabel" name="font">SansSerif.bold-bold-
11</property>
  <property part-class="JTextField" name="border">EtchedBorder</property>
  <property part-class="JTextField" name="font">Serif.italic-
11</property>
  <property part-class="JTextField" name="background">ccccff</property>
  <property part-class="JTextField" name="foreground">990000</property>
  <property part-class="JTextArea" name="border">EtchedBorder</property>
  <property part-class="JTextArea" name="font">Serif.italic-11</property>
  <property part-class="JTextArea" name="background">ccccff</property>
  <property part-class="JTextArea" name="foreground">990000</property>
  <property part-class="JRadioButton"
name="border">EmptyBorder,0,0,0,0</property>
  <property part-class="JRadioButton" name="font">Serif.italic-
11</property>
  <property part-class="JCheckBox"
name="border">EmptyBorder,0,0,0,0</property>
  <property part-class="JCheckBox" name="font">Serif.italic-11</property>
  <property part-class="JComboBox" name="border">EtchedBorder</property>
  <property part-class="JComboBox" name="font">Serif.italic-11</property>
  <property part-class="JComboBox" name="background">ccccff</property>
```

```
  <property part-class="JComboBox" name="foreground">990000</property>
  <property part-class="JList" name="border">EtchedBorder</property>
  <property part-class="JList" name="font">Serif.italic-11</property>
  <property part-class="JList" name="background">ccccff</property>
  <property part-class="JList" name="foreground">990000</property>
  <property part-class="JTable" name="background">ffffff</property>
  <property part-class="JTable" name="foreground">009900</property>
  <property part-class="JTable" name="font">Serif.italic-11</property>
  <property part-class="JTable" name="intercellSpacing">0,0</property>
  <property part-class="JTable" name="rowHeight">16</property>
  <property part-class="JTable"
name="border">BevelBorder,raised</property>
  </style>
  </interface>
- <peers>
- <logic>
+ <d-component id="MainFrame" name="MainFrame" maps-to="ui.UIFrame">
  </logic>
  <presentation base="Java_1.5_Harmonia_1.0" how="union"
export="optional" />
  </peers>
  </uiml>
```

Literatura

[1] Abrams M, Helms J, *User Interface Markup Language (UIML) Specification. Working Draft 3.1*, document wd-UIML-UIMLspecification-3.1, 11 March 2004. http://www.oasis-open.org/committees/download.php/5937/uiml-core-3.1-draft-01-20040311.pdf

[2] Aleksić S, Luković I, Pavićević J, *Jedan generator SQL opisa šeme baze podataka*, XIII Konferencija Industrijski sistemi IS 2005, 7-9. 9. 2005, Herceg Novi, Srbija i Crna Gora, Zbornik radova, pp. 341-348.

[3] Ali M. F, Pérez-Quiñones M. A, Abrams M, Shell E, *Building Multi-Platform User Interfaces with UIML*, Proceedings of 4th International Conference on Computer-Assisted Design of User Interfaces (CADUI 2002), France, May 2002, pp. 255- 266.

[4] ATLAS group, LINA & INRIA, *KM3: Kernel MetaMetaModel*, version 0.3, 2005. http://www.eclipse.org/gmt/am3/km3/doc/KernelMetaMetaModel%5Bv00.06%5D.pdf

[5] Bézivin J, *In Search of a Basic Principle for Model-Driven Engineering*, Novatica Journal, Special Issue, March-April 2004.

[6] Booch G, Brown A, Iyengar S, Rumbaugh J, Selic B, *An MDA Manifesto*, MDA Journal, May 2004.

[7] Cabot J, Clarisó R, Riera D, *Verifying UML/OCL Operation Contracts*, 7th International Conference on Integrated Formal Methods (IFM 2009), LNCS 5423, pp. 40-55.

[8] Cariou E, Marvie R, Seinturier L, Duchien L, *OCL for the Specification of Model Transformation Contracts*, OCL and Model Driven Engineering, UML 2004 Conference Workshop, October 12, 2004, Lisbon, Portugal, pp. 69-83, 2004.

[9] Choobinch J, Mannio V. M, Nunamaker F. J, Konsynski R. B, *An Expert Database Design System Based on Analysis of Forms*, IEEE Transactions on Software Engineering, Vol.14, No 2, 242-253, 1988.

[10] Date C. J, *An Introduction to Database Systems*, (sixth edition), Addison-Wesley Publishing Company, Reading, Massachusetts, 1995.

[11] Date C. J, *Composite Foreign Keys and Nulls*, In C. J. Date and H. Darwen Relational Database Writings 1989-1991, Addison-Wesley Publishing Company, Reading, Massachusetts, 1992.

[12] Date C. J, *Referential Integrity and Foreign Keys*, In Relational Database Writings 1985-1989, Addison- Wesley Publishing Company, Reading, Massachusetts, 1990.

[13] Date C. J, *Referential Integrity*, Proceedings of 7th International Conference on Very Large Data Bases, Canes, France, 1981, pp. 2-12.

[14] Diet J, Lochovsky F, *Interactive Specification and Integration of User Views Using Forms*, Proceedings of the Eight International Conference on Entity-Relationship Approach Toronto, Canada 18-20. October, 1989, pp.171-185.

[15] Eclipse Foundation, *ATL (ATLAS Transformation Language) Project*, *ATL/User Guide*, 2010. http://wiki.eclipse.org/ATL/User_Guide

[16] Favre J.M, *Foundations of Model (Driven) (Reverse) Engineering: Models*, Dagstahl Seminar Proceedings 4101, 2005.

[17] Govedarica M, *Automatizovani razvoj prototipova aplikacija informacionog sistema, doktorska disertacija*, Univerzitet u Novom Sadu, Fakultet Tehničkih nauka, Novi Sad, 2002.

[18] Govedarica M, Luković I, Mogin P, *Generating XML Based Specifications of Information Systems*, Computer Science and Information Systems (ComSIS), Consortium of Faculties of Serbia and Montenegro, Belgrade, Serbia, ISSN: 1820-0214, Vol. 1, No. 1, 2004, pp. 117-140.

[19] João Pereira M, Mernik M, Cruz D, Rangel Henriques P, *Program Comprehension for Domain-Specific Languages*, Computer Science and Information Systems (ComSIS), ISSN: 1820-0214, Vol. 5, No. 2, December 2008, pp 1-17.

[20] Kelly S, Tolvanen J, *Domain-Specific Modeling: Enabling Full Code Generation*, Wiley-IEEE Computer Society Press, 2008, ISBN: 0470036664.

[21] Kleppe A, Warmer J, Bast W, *MDA Explained: The Model Driven Architecture: Practice and Promise*, Addison-Wesley, Boston, 2003.

[22] Luković I, *Automatizovano generisanje podšeme relacione baze podataka putem formi*, magistarski rad, Univerzitet u Beogradu, Elektro-tehnički fakultet, Beograd, 1993.

[23] Luković I, *From the Synthesis Algorithm to the Model Driven Transformations in Database Design*, 10th International Scientific Conference on Informatics (Informatics 2009), November 23-25, 2009, Herlany, Slovakia, Proceedings, Slovak Society for Applied Cybernetics and Informatics and Technical University of Košice - Faculty of Electrical Engineering and Informatics, ISBN 978-80-8086-126-1, pp. 9-18, Invited Talk.

[24] Luković I, Mogin P, *Lossless Joins of Relational Database Views*, Review of Research, Faculty of Science, Novi Sad, Serbia (Yugoslavia), Mathematics Series, ISSN: 1450-5444, Vol. 26, No. 2, 1996, pp. 49-73.

[25] Luković I, Mogin P, Pavićević J, Ristić S, *An Approach to Developing Complex Database Schemas Using Form Types*, Software: Practice and Experience, John Wiley & Sons Inc, Hoboken, USA, ISSN: 0038-0644, DOI: 10.1002/spe.820, Vol. 37, No. 15, 2007, pp. 1621-1656.

[26] Luković I, Popović A, Mostić J, Ristić S, *A Tool for Modeling Form Type Check Constraints*, International Multiconference on Computer Science and Information

Technology (IMCSIT), 2nd Workshop on Advances in Programming Languages (WAPL'09), October 12-14, 2009, Mragowo, Poland, Proceedings, Polish Information Processing Society, ISSN 1896-7094, Vol. 4, pp. 681-688.

[27] Luković I, Ristić S, Aleksić S, Popović A, *An Application of the MDSE Principles in IIS*Case*, III Workshop on Model Driven Software Engineering (MDSE 2008), Berlin, Germany, December 11-12, 2008, Proceedings, TFH, University of Applied Sciences Berlin, pp.53-62.

[28] Luković I, Ristić S, Mogin P, *A Metodology of A Database Schema Design Using The Subschemas*, IEEE International Conference on Computational Cybernetics (ICCC), Siofok, Hungary, August 29-31, 2003, Proceedings, Budapest Polytechnic, Budapest, Hungary, in CD ROM.

[29] Luković I, Ristić S, Mogin P, Pavićević J, *Database Schema Integration Process – A Methodology and Aspects of Its Applying*, Novi Sad Journal of Mathematics (Formerly Review of Research, Faculty of Science, Mathematic Series), Serbia, ISSN: 1450-5444, Vol. 36, No. 1, 2006, pp 115-150.

[30] Luyten K, Coninx K, *Uiml.net: An Open Uiml Renderer for the .Net Framework*, Proceedings of the CADUI 2004, pp. 257-268, Funchal, Madeira, Portugal.

[31] Milosavljević G, Perišić B, *A Method and a Tool for Rapid Prototyping of Large-Scale Business Information Systems*, Computer Science and Information Systems, Vol. 1, Number 2, November 2004.

[32] Milosavljević G, *Prilog metodama brzog razvoja adaptivnih poslovnih informacionih sistema*, doktorska disertacija, Univerzitet u Novom Sadu, Fakultet Tehničkih nauka, Novi Sad, 2010.

[33] Mogin P, Luković I, *A Prototyping CASE Tool*, XXVIII International Symposium on Automotive Technology and Automation (ISATA), Stuttgart, Germany, September 18-22, 1995, Proceedings for the Dedicated Conference on Rapid Prototyping in the Automotive In-dustries, Fraunhofer Institute for Industrial Engineering (IAO), Stuttgart, Germany, pp. 261-268.

[34] Mogin P, Luković I, Govedarica M, *Principi projektovanja baza podataka*, drugo izdanje, Univerzitet u Novom Sadu – Fakultet Tehničkih nauka, Novi Sad, Srbija, 2004, ISBN: 86-80249-81-5.

[35] Mogin P, Luković I, Karadžić Ž, *Relational Database Schema Design And Application Generating Using IIS*CASE Tool*, Proceedings of International Conference on Technical Informatics, Timisoara, Romania, 16-19. 11. 1994, Vol. 5, pp.49-58.

[36] Mogin P, Luković I, *Principi baza podataka*, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Stylos, Novi Sad, 1996.

[37] Mostić J, *Specifikacija regularnih izraza i funkcija u alatu IIS*Case*, magistarski rad, Univerzitet u Novom Sadu, Fakultet Tehničkih nauka, Novi Sad, 2009.

[38] Nagaraj N.S, Thonse S, *MDA – Enabling seamless application development*, Comfactory, SETLabs, Infosys Technologies Ltd, 2001. http://www.infosys.com/research/publications/Documents/mda-analysis.pdf

[39] Object Management Group, *MDA Guide*, version 1.0.1, volume 1, document omg/03-06-01, 2003. http://www.omg.org/cgi-bin/doc?omg/03-06-01

[40] Object Management Group, *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification*, version 1.0, document formal/2008-04-03, 2008. http://www.omg.org/spec/QVT/1.0/PDF/

[41] Object Management Group, *Meta Object Facility (MOF) 2.0 XMI Mapping Specification*, version 2.1.1, document formal/07-12-01, 2007. http://www.omg.org/cgi-bin/doc?formal/2007-12-01

[42] Object Management Group, *Meta Object Facility (MOF) Core Specification*, version 2.0, document formal/06-01-01, 2006. http://www.omg.org/spec/MOF/2.0/PDF/

[43] Object Management Group, *OMG Unified Modeling LanguageTM (OMG UML), Infrastructure*, version 2.2, document formal/2009-02-04, 2009. http://www.omg.org/spec/UML/2.2/Infrastructure/PDF/

[44] Object Management Group: *Object Constraint Language*, version 2.0, document formal/06-05-01, 2006. http://www.omg.org/cgi-bin/doc?formal/06-05-01.pdf

[45] Pavićević J, Luković I, Mogin P, Ristić S, *IIS*Case - alat za automatizovano projektovanje i integraciju šema baza podataka*, XIII Konferencija Industrijski sistemi IS 2005, 7-9. 9. 2005, Herceg Novi, Srbija i Crna Gora, Zbornik radova pp. 321-330.

[46] Pavićević J, *Razvoj CASE alata za automatizovano projektovanje i integraciju šema baza podataka*, magistarski rad, Univerzitet Crne Gore, Prirodno-matematički fakultet, Podgorica, 2005.

[47] Popović A, *Specifikacija vizuelnih atributa i struktura poslovnih aplikacija u alatu IIS*Case*, magistarski rad, Univerzitet u Novom Sadu, Fakultet Tehničkih nauka, Novi Sad, 2006.

[48] Ristić S, *Istraživanje problema konsolidacije podšema baza podataka*, doktorska disertacija, Univerzitet u Novom Sadu, Ekonomski fakultet Subotica, 2003.

[49] Ristić S, Luković I, Aleksić S, Popović A, *An Approach to Building Platform Independent Models*, XIV International Scientific Conference on Industrial Systems IS 2008, October 2-3, 2008, Novi Sad, Serbia, Proceedings, University of Novi Sad, Faculty of Technical Sciences, ISBN 978-86-7892-135-3, pp. 201-206.

[50] Ristić S, Luković I, Pavićević J, Mogin P, *Resolving Database Constraint Collisions Supported by IIS*Case Tool*, XVII International Conference on Information and Intelligent Systems (IIS), Varaždin, Croatia, September 20-22, 2006, Proceedings, University of Zagreb, Faculty of Organization and Informatics, Varazdin, Croatia, ISBN: 953-6071-27-4, pp. 43-52.

[51] Rumbaugh J, Jacobson I, *The Unified Modeling Language Reference Manual*, Addison-Wesley, USA, 1999.

[52] Schmidt D. C, *Model-driven engineering*, IEEE Computer, Vol.39, No.2, 25-31, 2006.

[53] Singh Y, Sood M, *Models and Transformations in MDA*, First International Conference on Computational Intelligence, Communication Systems and Networks, 2009, pp. 253-258.

[54] Stahl T, Völter M, *Model Driven Software Development: Technology, Engineering, Management*, John Wiley & Sons, Ltd. 2006.

[55] Ullman D. J, *Principles of Database Systems*, Computer Science Press, Inc. Rockville, Maryland, 1982.

[56] World Wide Web Consortium (W3C), *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, W3C Recommendation, 2008. http://www.w3.org/TR/REC-xml/