UML profil za reprezentaciju šeme relacione baze podataka

Igor Tomić
Tehnička škola
Gradiška, BiH
tomicigor@hotmail.com

Dražen Brđanin, Slavko Marić
Elektrotehnički fakultet
Banja Luka, BiH
bdrazen@etfbl.net, ms@etfbl.net

Sadržaj **— U ovom radu predložena je jedna specifikacija UML profila za reprezentaciju šeme relacione baze podataka i prikazana implementacija *plug-ina* u razvojnoj platformi otvorenog koda, koji omogućava ručno modelovanje i programsko generisanje šeme u skladu sa definisanim profilom. Primjena profila ilustrovana je na primjeru vizuelizacije jedne konkretne šeme relacione baze podataka.**

Ključne riječi - UML*; *profil; šema relacione baze podataka*; *dijagram klasa*; *Eclipse*; *Topcased*; *plug-in; CWM

I. UVOD

Relacioni model [1] prešao je dug put razvoja od 70-ih godina prošlog vijeka do danas i postao je dominantan model organizacije baza podataka. Opis sveukupne strukture podataka i ograničenja koja se odnose na podatke u relacionoj bazi specifikuje se šemom relacione baze podataka. U postojećoj literaturi ne postoji jedinstven pristup za reprezentaciju šeme relacione baze podataka.

UML (*Unified Modeling Language*) [2], [3] je standardizovani grafički jezik za projektovanje, specifikaciju, vizuelizaciju i dokumentovanje softverskih sistema u različitim faza životnog ciklusa. Prvi razlog za njegovu široku upotrebu jeste bogata notacija za modelovanje koja je nezavisna od procesa modelovanja. Drugi razlog proističe iz otvorenosti koncepta, jer UML može da se proširuje, odnosno specijalizuje za određenu oblast. Skup proširenja standardne notacije za primjenu u određenom domenu naziva se UML profil.

Standardni UML nije u potpunosti prilagođen za projektovanje relacione baze podataka. Za razvoj konceptualnog modela, koji prikazuje tipove entiteta (objekata) i njihove međusobne veze, moguće je koristiti standardni UML dijagram klasa. Međutim, šema relacione baze podataka je specifičan model koji ne može da se reprezentuje standardnom notacijom, već je neophodna specijalizovana notacija koja omogućava modelovanje karakterističnih koncepata (relacione šeme/tabele, ključevi, ograničenja, itd.). Za razliku od nekih drugih domena za koje postoje standardizovani UML profili, trenutno ne postoji standardni profil za modelovanje šeme relacione baze podataka. S obzirom na semantičko bogatstvo UML-a i različite mogućnosti specijalizacije osnovne notacije, moguće su i različite implementacije profila. Pošto ne postoji ni jedinstven teorijski pristup, u praksi (npr. [4]) se koriste različiti profili za modelovanje šeme relacione baze podataka.

Predmet ovog rada jeste specifikacija UML profila za reprezentaciju šeme relacione baze podataka, kao i implementacija *plug-ina* koji omogućava ručno modelovanje i programsko generisanje šeme relacione baze na razvojnoj platformi otvorenog koda u skladu sa definisanim profilom.

Rad je organizovan na sljedeći način. Nakon uvodnog dijela, u drugom dijelu data je specifikacija UML profila za reprezentaciju šeme relacione baze podataka. U trećem dijelu opisana je implementacija Eclipse-Topcased *plug-ina* koji omogućava ručno modelovanje i programsko generisanje šeme relacione baze u skladu sa definisanim profilom. Primjena profila ilustrovana je u četvrtom dijelu kroz programsko generisanje i automatsku vizuelizaciju UML šeme na osnovu jedne CWM (*Common Warehouse Metamodel*) [5] šeme relacione baze podataka. Na kraju su dati zaključci i pravci za nastavak istraživanja.

II. SPECIFIKACIJA UML PROFILA ZA REPREZENTACIJU ŠEME RELACIONE BAZE PODATAKA

Standardna UML notacija može da se prilagodi za primjenu u nekom specifičnom domenu, tako što se na osnovu postojećih koncepata, primjenom standardnih mehanizama za proširenje notacije, definišu novi koncepti koji omogućavaju modelovanje specifičnosti konkretnog domena. Standardni mehanizmi za proširenje, odnosno specijalizaciju notacije su: stereotip, označena vrijednost i ograničenje.

A. Proširenje UML metamodela

S obzirom na to da šema relacione baze podataka reprezentuje sveukupnu strukturu i ograničenja koja se odnose na podatke u relacionoj bazi podataka, polazni osnov za specifikaciju odgovarajućeg profila predstavlja dijagram klasa, kao osnovni i najčešće korišćeni standardni strukturni UML dijagram. Na sl. 1 prikazan je dijagram profila koji predstavlja specijalizaciju standardnog UML dijagrama klasa, koji omogućava reprezentaciju šeme relacione baze podataka u skladu sa [6].

Stereotip «`DatabaseModel`», kao specijalizacija metaklase `Package`, služi za reprezentaciju paketa koji sadrži šemu relacione baze podataka. Ovaj stereotip ima atribut `DBMSname`, koji specifikuje ciljni sistem za

upravljanje bazama podataka (eng. *DataBase Management System* – DBMS) za koji se modeluje odgovarajuća šema relacione baze podataka. Atribut `DBMSname` je instanca tipa podataka «`enumeration`» `DBMSkinds`, koji u ovoj fazi implementacije profila, sadrži sljedeće vrijednosti: *MySQL*, *Oracle*, *Microsoft SQL*, *IBM DB2* i *PostgreSQL*. Ove vrijednosti reprezentuju istoimene, često korišćene sisteme za upravljanje bazama podataka. Od vrijednosti atributa `DBMSname` direktno zavise mogući tipovi podataka za specifikaciju kolona u tabelama (relacionim šemama), koje su sadržane u odnosnoj šemi relacione baze podataka.

Stereotip «`Table`», kao specijalizacija metaklase `Class`, služi za reprezentaciju tabela.

Stereotip «`Column`», kao specijalizacija metaklase `Property`, služi za reprezentaciju atributa relacione šeme, odnosno kolona u tabeli. Ovaj stereotip ima atribute `type` i `isRequired`. Atribut `isRequired` je logički podatak koji specifikuje da li odnosna kolona može sadržavati vrijednost *null*. Podrazumijevana vrijednost ovog atributa je *false* (kolona može da sadrži vrijednost *null*). Atribut `type` specifikuje tip podataka u odnosnoj koloni. U zavisnosti od vrijednosti `DBMSname` atributa (u stereotipu «`DatabaseModel`»), vrijednost atributa `type` može da bude instanca jednog od sljedećih «`enumeration`» tipova: `MySQLDataTypes`, `MicrosoftSQLDataTypes`, `PostgreSQLDataTypes`, `OracleDataTypes`, `IBMDB2DataTypes`. S obzirom na to da je MySQL podrazumijevani ciljni DBMS, podrazumijevana vrijednost atributa `type` je `MySQLDataTypes`.

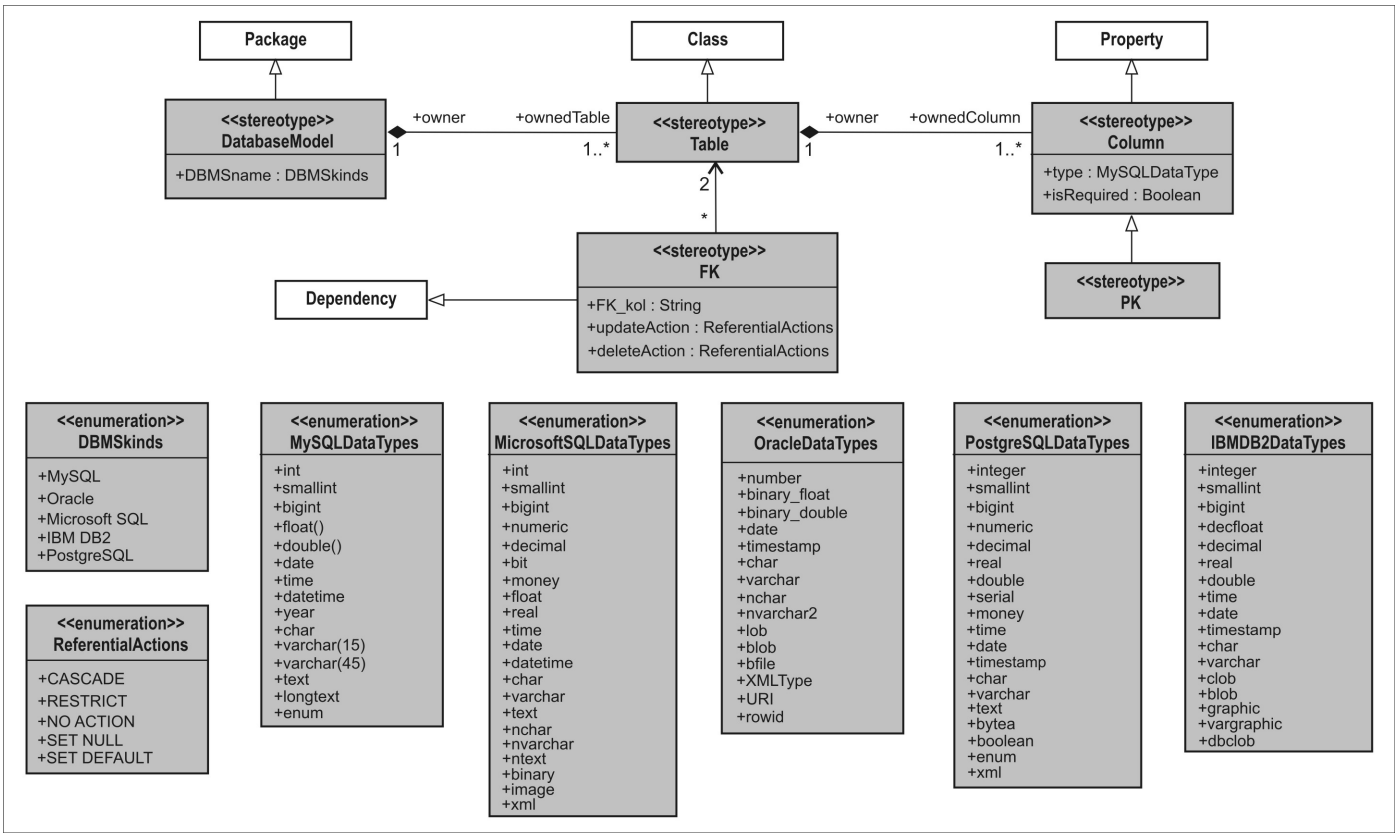
Stereotip «`PK`», kao specijalizacija stereotipa «`Column`», služi za reprezentaciju primarnog ključa, tj. za specifikaciju kolona koje sačinjavaju primarni ključ u tabeli.

Ograničenja po referencijalnom integritetu između dvije tabele reprezentuju se stereotipom «`FK`», koji predstavlja specijalizaciju metaklase `Dependency`. Ova specijalizovana zavisnost usmjerena je od referencirajuće prema referenciranoj tabeli (u skladu sa zavisnošću stranog ključa u referencirajućoj od primarnog ključa u referenciranoj tabeli). Pored atributa `FK_kol`, koji sadrži nazive kolona koje čine strani ključ u referencirajućoj tabeli, ovaj stereotip ima još atribute `updateAction` i `deleteAction` za specifikaciju akcija referencijalnog integriteta. Oba atributa su instance tipa «`enumeration`» `ReferentialActions`, koji sadrži pet tipičnih vrijednosti: *CASCADE*, *RESTRICT*, *NO ACTION*, *SET NULL* i *SET DEFAULT*. Podrazumijevana vrijednost ova dva atributa jeste *NO ACTION*.

B. Specifikacija ograničenja

Definisani UML profil sadrži i ograničenja (pravila) koja se ne mogu specifikovati samo specijalizacijom UML metaklasa. Ta ograničenja specifikovana su OCL (*Object Constraint Language*) izrazima.

OCL [7] je deklarativni jezik za opis pravila koja UML modeli treba da zadovoljavaju. OCL izrazi ne mogu direktno da se izvršavaju i njima ne mogu da se izraze implementacioni detalji, ali mogu da se verifikuju promjene modela. OCL se tipično koristi za specifikaciju ograničenja u modelima, ali i kao upitni jezik.



Slika 1. Dijagram UML profila za reprezentaciju šeme relacione baze podataka (osnovne metaklase prikazane su bijelo, a proširenje metamodela osjenčeno)

Definisani profil omogućava reprezentaciju šeme relacione baze podataka za nekoliko različitih sistema za upravljanje bazama podataka, od kojih svaki ima sopstveni skup tipova podataka za reprezentaciju atributa (kolona). U zavisnosti od izabranog ciljnog sistema, na raspolaganju su pripadajući tipovi podataka. Takođe, svaka baza podataka mora da sadrži bar jednu tabelu. Ova ograničenja se specifikuju sljedećim OCL šablonom:

```
Contex DatabaseModel
inv:
 self.ownedElement->forAll(c|c.oclIsKindOf(Table)) and
 self.ownedElement->size()>0 and
 if (self.DBMSname==DBMSkinds.DBMS_i)
  (let tables:Collection(Table)=self.ownedElement
       -> collect(f|f.oclIsKindOf(Table))
   tables.ownedColumn->
     forAll(t|t.type.oclIsTypeOf(DBMS_i_Types)))
```

gdje *DBMS_i* uzima redom sve vrijednosti iz enumeracije DBMSkinds (*MySQL*, *Oracle*, *Microsoft SQL*, *IBM DB2* i *PostgreSQL*), a *DBMS_i_Types* redom korespondentne nazive tipova podataka (MySQLDataTypes, itd).

Svaka tabela mora da sadrži bar jednu kolonu i mora da ima primarni ključ koji jedinstveno identifikuje svaki red u tabeli (ne mogu da postoje dva zapisa sa istom vrijednošću na atributima primarnog ključa). Pored toga, za svake dvije povezane tabele mora da važi i ograničenje referencijalnog integriteta, tj. vrijednost stranog ključa u referencirajućoj tabeli mora da referencira odnosnu *n-torku* (koja mora da postoji) u referenciranoj tabeli. Ova ograničenja specifikuju se sljedećim OCL šablonom:

```
Contex Table
inv:
 self.ownedColumn->forAll(c|c.oclIsKindOf(Column)) and
 self.ownedColumn->size()>0 and
 let pKey:Collection(Column) =
     self.ownedColumn->collect(f|f.oclIsKindOf(PK))
 pKey->size()>0 and
 pKey.allInstances->forAll(i,j|i<>j) and
 self.Client Dependency->forAll(f|f.oclIsKindOf(FK)) and
 let dep:Collection(Dependency) =
     self.Client Dependency->collect(d|d.oclIsKindOf(FK))
 dep.allInstances->forAll(d,g|d.Supplier<>g.Supplier) and
 dep.allInstances->forAll(d|
   let pKey:Collection(Column)=
       d.Supplier.ownedColumn->
       collect(p|p.oclIsKindOf(PK))
   let fKey:Collection(Column)=
       d.Client.ownedColumn->collect(f|f.Name==d.FK_kol)
   pKey.type==fKey.type and
   pKey.allInstances->forAll(i|fKey.allInstances->
     collect(j|j==i)->size()>0))

Contex FK
inv:
 self.allInstances->forAll(j|
  let fKey:Collection(Column)=
      self.Client.ownedColumn->
        collect(k|k.Name==j.FK_kol)->size()>0)
```

Kolone koje pripadaju primarnom ključu ne mogu da imaju *null* vrijednost. Ovo ograničenje specifikovano je sljedećim OCL izrazom:

```
Contex PK
inv:
  self.isRequired=true
```

III. IMPLEMENTACIJA UML PROFILA

Definisani UML profil za reprezentaciju šeme relacione baze podataka je platformski nezavisan i može praktično da se implementira u različitim razvojnim okruženjima i alatima za modelovanje. U ovom radu profil je implementiran kao *plug-in* za Eclipse Topcased [8] razvojnu platformu. Osnovni kôd (Java) metode koja implementira profil u skladu sa predloženom specifikacijom (djelimično) je prikazan na sl. 2.

```
public static void generateProfile()
{
  // kreiranje projekta za UML profil
  IProject project = ResourcesPlugin.getWorkspace().
                     getRoot().getProject("UML");
  // ...
  uml = createProfile("UML_database_profile");
  // kreiranje stereotipova
  model=createStereotype(uml,"DatabaseModel",false);
  table=createStereotype(uml,"Table",false);
  column=createStereotype(uml,"Column",false);
  pk=createStereotype(uml,"PK",false);
  fk=createStereotype(uml,"FK",false);
  // metaklase koje referenciraju stereotipovi
  org.eclipse.uml2.uml.Class propertyMetaclass =
    referenceMetaclass(uml,
      UMLPackage.Literals.PROPERTY.getName());
  org.eclipse.uml2.uml.Class classMetaclass =
    referenceMetaclass(uml,
      UMLPackage.Literals.CLASS.getName());
  org.eclipse.uml2.uml.Class modelMetaclass =
    referenceMetaclass(uml,
      UMLPackage.Literals.PACKAGE.getName());
  org.eclipse.uml2.uml.Class fkMetaclass =
    referenceMetaclass(uml,
      UMLPackage.Literals.DEPENDENCY.getName());
  // specijalizacija stereotipova
  createExtension(propertyMetaclass,column,false);
  createGeneralization(pk,column);
  createExtension(classMetaclass,table,false);
  createExtension(modelMetaclass,model,false);
  createExtension(fkMetaclass,fk,false);
  // kreiranje DBMStypes enumeracije
  Enumeration DBMStypes = createEnumeration(uml,
    "DBMSkinds");
  createEnumerationLiteral(DBMStypes, "MySQL");
  createEnumerationLiteral(DBMStypes, "Oracle");
  // ...
  // kreiranje atributa u DatabaseModel stereotipu
  Property DBMSname=createAttribute
    (model,"DBMSname", DBMStypes, 0, 1);
  // ucitavanje prostih tipova podataka
  PrimitiveType booleanPT =
    importPrimitiveType(uml,"Boolean");
  PrimitiveType stringPT =
    importPrimitiveType(uml,"String");
  // kreiranje atributa u Column stereotipu
  Property isRequired = createAttribute(column,
    "isRequired", booleanPT, 0, 1);
  // kreiranje MySQLDataTypes enumeracije
  Enumeration MySQLDataTypes =
    createEnumeration(uml, "MySQLDataTypes");
  createEnumerationLiteral(MySQLDataTypes,"int");
  // ...
  // kreiranje ostalih enumeracija tipova podataka
  // ...
```

Slika 2. Osnovni kôd metode koja implementira UML profil

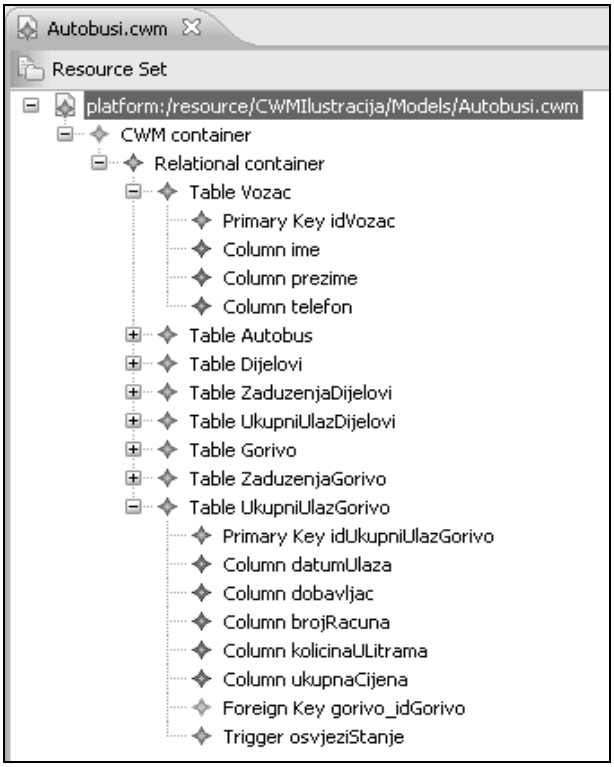
```
  Property type = createAttribute(column, "type",
    MySQLDataTypes, 0, 1);

  // kreiranje atributa u FK stereotipu
  Property FK_kol = createAttribute(fk, "FK_kol",
    stringPT, 0, 1);

  // kreiranje ReferencialAction enumeracije
  Enumeration ReferencialAction =
    createEnumeration(uml, "ReferencialAction");
  createEnumerationLiteral(ReferencialAction,
    "NO ACTION");
  createEnumerationLiteral(ReferencialAction,
    "CASCADE");
  // ...

  Property updateAction = createAttribute(fk,
    "updateAction", ReferencialAction, 0, 1);
  Property deleteAction = createAttribute(fk,
    "deleteAction", ReferencialAction, 0, 1);

  // kreiranje kompozicije modela i tabele
  createAssociation(model, true,
    AggregationKind.COMPOSITE_LITERAL, "ownedTable",
    1, LiteralUnlimitedNatural.UNLIMITED, table,
    true,AggregationKind.NONE_LITERAL, "owner",1,1);

  // kreiranje kompozicije tabele i kolone
  createAssociation(table, true,
    AggregationKind.COMPOSITE_LITERAL,"ownedColumn",
    1, LiteralUnlimitedNatural.UNLIMITED, column,
    true,AggregationKind.NONE_LITERAL, "owner",1,1);

  // kreiranje asocijacije tabele i stranog kljuca
  createAssociation(table, true,
    AggregationKind.NONE_LITERAL, "", 0,
    LiteralUnlimitedNatural.UNLIMITED, fk,
    true, AggregationKind.NONE_LITERAL, "", 2, 2);

  // definicija profila
  defineProfile(uml);

  // memorisanje profila
  save(uml,URI.createURI("/UML").
       appendSegment("UML_profil").
       appendFileExtension(UMLResource.
       PROFILE_FILE_EXTENSION));
}
```

Slika 2. Osnovni kôd metode koja implementira UML profil (nastavak)

IV. ILUSTRATIVNI PRIMJER PRIMJENE UML PROFILA

Implementirani profil omogućava ručno modelovanje šeme relacione baze podataka u Topcased razvojnom okruženju, ali i programsko generisanje šeme relacione baze u skladu sa definisanim profilom. Ovdje je primjena implementiranog profila ilustrovana kroz programsko generisanje i automatsku vizuelizaciju UML šeme na osnovu CWM šeme relacione baze podataka.

CWM [5] je specifikacija za modelovanje i razmjenu metapodataka o relacionim, nerelacionim, višedimenzionalnim i drugim objektima u skladištima podataka. CWM predstavlja jedinstven okvir za reprezentaciju metapodataka te procesa i operacija za manipulaciju metapodacima. CWM specifikacija omogućava laku razmjenu metapodataka između različitih platformi i alata za skladištenje podataka te samih skladišta podataka u distribuiranim heterogenim okruženjima. CWM šema relacione baze podataka, koja se u ovom primjeru koristi kao polazni osnov za programsko generisanje i automatsku vizuelizaciju odgovarajuće UML šeme, prikazana je na sl. 3.



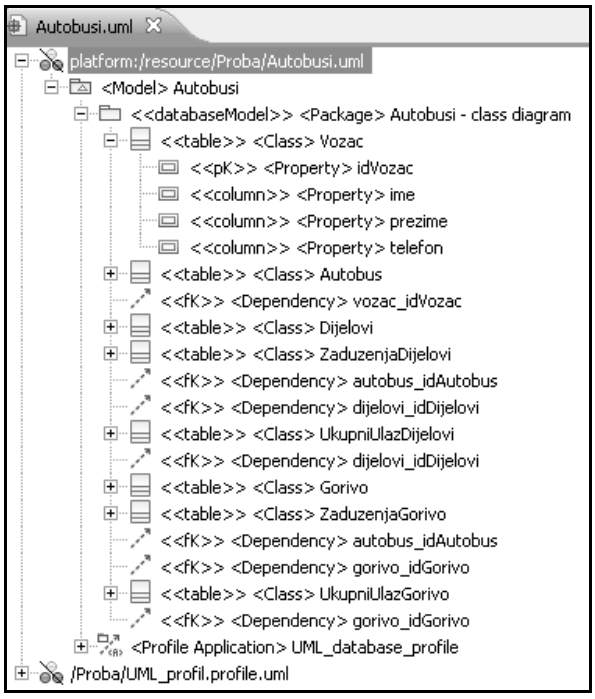
Slika 3. Primjer CWM šeme relacione baze podataka

Realizovani softverski alat za automatsku vizuelizaciju CWM šeme relacione baze podataka predstavlja unaprijeđenu verziju alata [9], u kojem se za vizuelizaciju koristi standardna UML notacija. Alat je implementiran kao Eclipse-Topcased *plug-in* pod nazivom **VisualCWM**.

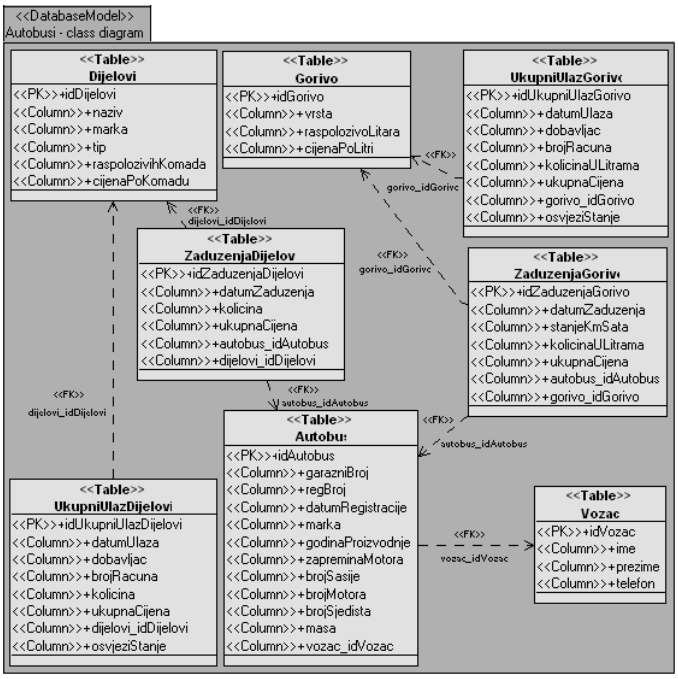
Funkcionalnost Topcased platforme i implementiranog generatora zasnovana je na EMF-zasnovanoj [10] implementaciji UML 2.1 specifikacije, koja omogućava vizuelno UML modelovanje, programsko generisanje dijagrama i automatsku vizuelizaciju programski generisanih UML dijagrama. Svaki UML dijagram u Topcased modelu reprezentuje se sa dva istoimena fajla različitih ekstenzija. Fajl sa ekstenzijom *.uml* sadrži XMI (*XML Metadata Interchange*) reprezentaciju modela, a fajl sa *.umldi* ekstenzijom opisuje njegovu vizuelizaciju.

Implementacija se temelji na kombinaciji DOM XML parsera za analizu ulazne CWM šeme i primjeni UML2 *factory* [11] za generisanje ciljnog dijagrama. Generator na osnovu ulaznog *.cwm* fajla generiše odgovarajuće XML stablo i ekstrahuje karakteristične koncepte ulazne CWM šeme, a zatim generiše korespondentne koncepte ciljnog dijagrama klasa u skladu sa definisanim profilom i XMI specifikacijom i kreira ciljni *.uml* fajl. Automatska vizuelizacija programski generisanog dijagrama (kreiranje *.umldi* fajla) potom se vrši pomoću ugrađene Topcased funkcionalnosti.

Rezultat primjene generatora na ulaznu CWM šemu (sl. 3) jeste automatski generisana XMI reprezentacija (*.uml* fajl) ciljnog dijagrama klasa (sl. 4), dok je njena vizuelizacija prikazana na sl. 5. Dobijeni dijagram pokazuje da je implementirani generator automatski generisao šemu u skladu sa definisanim profilom.



Slika 4. Programski generisana XMI reprezentacija polazne CWM šeme sa sl. 3



Slika 5. Rezultat vizuelizacije programski generisane šeme koja reprezentuje polaznu CWM šemu sa sl. 3

V. ZAKLJUČAK

U radu je predložena specifikacija UML profila za reprezentaciju šeme relacione baze podataka, koja se zasniva na specijalizaciji UML metamodela i odgovarajućim OCL ograničenjima. Na osnovu predložene specifikacije implementiran je *plug-in* u Topcased razvojnoj platformi otvorenog koda, koji omogućava ručno modelovanje i programsko generisanje šeme za nekoliko često korišćenih sistema za upravljanje bazama podataka.

Primjena profila ilustrovana je na primjeru automatske vizuelizacije jedne konkretne CWM šeme relacione baze podataka pomoću automatskog softverskog generatora, koji vrši ekstrakciju karakterističnih elemenata ulazne CWM šeme i automatski generiše korespondentne koncepte u UML dijagramu klasa kojim se vizuelizuje polazna šema u skladu sa definisanim profilom. Generator je implementiran kao Topcased *plug-in* pod nazivom **VisualCWM**.

Dobijeni rezultati pokazuju da predloženi UML profil omogućava zadovoljavajuću vizuelizaciju šeme relacione baze podataka. U nastavku istraživanja planirana je ugradnja implementiranog automatskog generatora šeme relacione baze podataka u ADBdesign [12] alat za automatizovano projektovanje baze podataka na osnovu poslovnog modela.

LITERATURA

[1] E. Codd, "A relational model of data for large shared data banks", Communications of the ACM, 13(5), pp. 377-387, 1970.

[2] Object Management Group (OMG), Unified Modeling Language (UML): Infrastructure, v2.4.1. OMG, 2011.

[3] Object Management Group (OMG), Unified Modeling Language (UML): Superstructure, v2.4.1. OMG, 2011.

[4] E. Naiburg, and R. Maksimchuk, UML for database design. Reading: Addison-Wesley, 2001.

[5] Object Management Group (OMG), Common Warehouse Metamodel Specification (CWM) v1.1. OMG, 2003.

[6] S. Marić, i D. Brđanin, Relacione baze podataka. Banja Luka: Elektrotehnički fakultet, 2012.

[7] Object Management Group (OMG), Object Constraint Language (OCL), ISO/IEC 19507, 2012.

[8] Topcased, http://www.topcased.org.

[9] I. Tomić, D. Brđanin, i S. Marić, "Jedan pristup za vizuelizaciju CWM-bazirane šeme relacione baze podataka", Zbornik radova INFOTEH-JAHORINA, vol. 11, pp. 639-642, 2012.

[10] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. Grose, Eclipse Modeling Framework. Boston: Pearson Education, 2003.

[11] K. Hussey, Getting Started with UML2. New York: IBM Corp, 2006.

[12] D. Brdjanin, and S. Maric, "An approach to automated conceptual database design based on the UML activity diagram", Computer Science and Information Systems, 9 (1), pp. 249-283, 2012.

ABSTRACT

In this paper we propose a specification of UML profile for representation of the relational database schema and present the corresponding implementation of an open-source plug-in that enables manual modeling, as well as software schema generation in accordance with the defined profile. The proposed approach is illustrated by an example of visualization of a particular relational database schema.

UML PROFILE FOR RELATIONAL DATABASE SCHEMA REPRESENTATION

Igor Tomic, Drazen Brdjanin, Slavko Maric