ДОКУМЕНТАЦИЈА ТЕХНИЧКОГ РЕШЕЊА

IIS*Case V7.0 − алат за пројектовање и генерисање база података и апликација – Модул за генерисање прототипова апликација

Аутори техничког решења

Иван Луковић, Јелена Бановић, Александар Поповић, Павле Могин, Соња Ристић, Миро Говедарица, Иван Бекер

Година када је техничко решење урађено

2010.

Апстракт

Техничко решење је **Модул за генерисање прототипова апликација,** који је део IIS*Case-а – софтверског алата за аутоматско пројектовање и генерисање шема база података. Овај програмски модул генерише извршне софтверске спецификације, чиме омогућава аутоматско генерисање прототипова апликација у раној фази развоја информационог система. У модулу су имплементирани алгоритми за генерисање подшема типова форми неопходних за имплементацију функционалности читања и ажурирања података путем корисничког интерфејса генерисаних прототипова апликација, као и генератор који на основу задатих спецификација апликација информационих система и изабраног шаблона корисничког интерфејса генерише функционалне прототипове апликација.

Модул за генерисање прототипова апликација имплементиран је у клијент-сервер архитектури, где сервер представља произвољан систем за управљање базама података са којим се комуницира путем ODBC протокола. Апликација је реализована на програмском језику Јава, у окружењу Oracle JDeveloper верзија 10.1.3.1.0.

Извршне софтверске спецификације информационих система генеришу се путем UIML (*User Interface Markup Languge*) спецификација које се могу интерпретирати коришћењем *Java Render*-а. Овај поступак интерпретира само визуелне карактеристике корисничког интерфејса, а не и сложене функционалности као што су нпр. функционалности које подржавају комуникацију са базом података. Овај недостатак *Java Render*-а превазиђен је имплементацијом Јава класа које подржавају сложене функционалности генерисаних прототипова, што представља значајан допринос овог техничког решења.

Поред тога, посебан допринос овог решења огледа се у имплементацији алгоритама за генерисање подшема типова форми неопходних за имплементацију функционалности читања и ажурирања података путем корисничког интерфејса генерисаних прототипова апликација.

Техничко решење примењује се у едукацији студената академских основних студија и докторских студија на Факултету техничких наука у Новом Саду и Природно-математичком факултету Универзитета Црне Горе у Подгорици, а започиње и експериментална примена, за развој софтверских решења, на Економском факултету и Машинском факултету у Крагујевцу и у Застава аутомобили А.Д. Крагујевац.

1. Увод

Модул за генерисање прототипова апликација, је део **IIS*Case-a** (***I**ntegrated **I**nformation **S**ystems **C**ASE Tool*). IIS*Case је алат који припада класи интегрисаних CASE производа, са задатком да подржи највећи број активности развоја програмског производа. При томе, општи циљ развоја IIS*Case-a је да обезбеди аутоматизацију и интелигентну подршку спровођењу комплексних и високо формализованих пројектантских и програмерских активности.

У процесу пројектовања информационог система, након задавања пројектних спецификација и генерисања имплементационе шеме базе података, модул за генерисање прототипова апликација IIS*Case-a, у финалном кораку, пројектанту пружа могућност генерисања трансакционих програма, односно извршних софтверских спецификација апликативних система. Поступак генерисања реализован је имплементацијом већег броја, по обиму сложених алгоритама. Рад генератора трансакционих програма своди се на "позадинско" извршење тих алгоритама, а пројектанту се пружа увид само у крајњи резултат – прототип апликативног система, евентуално са придруженим "дијагностичким" извјештајем. Овај прототип је функционална апликација која задовољава све полазне корисничке функционалне и визуелне захтеве, као и дефинисана пословна правила дата кроз спецификацију пројекта и корисничког шаблона. Апликација користи ODBC конекцију за комуникацију са претходно генерисаном имплементационом шемом базе података и подржава све функционалности приказа и ажурирања података.

Процес генерисања трансакционог програма састоји се из два корака:

1. *Генерисање подшема типова форми*, имплементирано путем алгоритама дефинисаних у [9]. Овај корак је предуслов за извршавање следећег корака, јер као резултат даје неопходне улазне информације на основу којих ће бити креирани SQL упити, путем којих ће генерисана апликација комуницирати са базом података.
2. *Генерисање апликације*. Овај корак укључује генерисање UIML документа са спецификацијом визуелних и функционалних аспеката апликативног система. Генератор покреће *Java Renderer* који генерисану спецификацију интерпретира као Јава апликацију.

2. Област техничког решења

Техничко решење припада научној области Информатика, ужа научна област Информациони системи. Техничко решење *IIS*Case V7.0 − алат за пројектовање и генерисање база података и апликација – Модул за генерисање прототипова апликација* по Правилнику Министарства за науку и технолошки развој Републике Србије припада категорији М80 (*Техничка и развојна решења*). У оквиру ове категорије, техничко решење спада у *Развој софтвера* (**М85**).

3. Преглед постојећих решења

Са појавом методологија и алата који омогућавају креирање софтверских модела на високо апстрактном нивоу, приступ који подразумева употребу модела као кључне и нераздвојне компоненте развојног процеса, постао је један од водећих приступа у развоју софтверских система.

Моделовање софтверских система представља дизајнирање структуре и функционалности софтверских система које претходи самом креирању програмског кода. Употребом модела, софтверски систем се развија на високо апстрактном нивоу. Скривајући детаље, модел даје различите слике система. С једне стране то може да буде генерална слика система (поглед на систем који даје информације на којој је платформи имплементиран, како је повезан са другим системима, итд.), док с друге стране модел може да буде фокусиран на одређене аспекте структуре система (пословни процеси које апликативни систем аутоматизује, пословна правила и сл.).

Један од приступа који дају моделима водећу улогу у развојном процесу је *Model-Driven Architecture* или скраћено MDA ([3], [8], [13]). Употреба овог приступа представља могући

начин да се умањи комплексност развојног процеса, да се постигну високи нивои поновне употребљивости постојећих програмских модула и значајно смањи напор уложен у пројекте развоја софтвера. Алати који подржавају MDA олакшавају пројектанту да премости јаз између анализе пројектних захтева и конкретне имплементације система.

MDA представља такав приступ моделовању информационих система који раздваја спецификацију функционалности и структура система од спецификације конкретне имплементације на изабраној платформи. Овај приступ разликује платформски независне моделе (*Platform-Independent Model*, PIM) и платформски специфичне моделе *Platform-Specific Model*, PSM). Док PIM модел представља спецификацију дела или целог система која не зависи од изабране платформе на којој ће систем бити имплементиран, PSM модел подразумева спецификацију која је уско повезана са изабраном платформом и у себи укључује детаље специфичне за конкретну технологију.[12]

У MDA приступу животни циклус софтверског система представљен је трансформацијама модела које се остварују кроз фазе животног циклуса и које, на крају, резултирају у конкретној имплементацији система. Трансформације једног PIM модела у други PIM модел, тзв. PIM-у-PIM трансформације модела, присутне су током фаза анализе и пројектовања. Трансформације PIM модела у PSM моделе, тзв. PIM-u-PSM трансформације, извршавају се током трансформације спецификација пројекта у имплементацију система. Такође постоје и трансформације PSM модела у PIM моделе, тзв. PSM-u-PIM трансформације које се примјењују у процесу реверзног инжењеринга. ([12], [17])

Постоји неколико кључних језика и/или окружења у оквиру MDA приступа од којих су најважнији: UML, XMI, QVT, OCL і MOF. *Unified Modeling Language* (UML) постао је водећи језик за пројектовање модела и архитектуре система. *XML Metadata Interchange* (XMI) је стандард који се користи за репрезентацију UML модела и омогућавање интероперабилности између MDA алата различитих произвођача. *Query/View/Transformation* (QVT) је језик са софтверским окружењем које омогућава креирање различитих трансформација модела. *Object Constraint Language* (OCL) је језик који се користи за спецификацију ограничења у моделима. *Meta Object Facility* (MOF) дефинише конструкционе елементе неопходне за креирање репрезентације метамодела. Сви концепти обухваћени MDA приступом, засновани су на MOF-у.

OMG група званично је објавила *MetaObject Facility* (MOF) језик у Августу 2004. године дефинишући да сви језици за моделовање коришћени у MDA приступу развоју софтверских система морају бити описани MOF језиком. На тај начин обезбеђено је да се метаподаци специфицирају на стандардизован начин, чиме се испуњава предуслов за аутоматизовање различитих трансформација модела између различитих развојних окружења. Постоје различити приступи, односно језици, који подржавају поступке трансформације модела, а овде ће бити описана три: OCL, QVT и ATL.

Object Constraint Language (OCL) представља формални, текстуални језик који омогућава прецизно дефинисање ограничења и упита на било ком MOF моделу. Иако је креиран као формалан језик, OCL је лако читати и писати. У питању је језик за креирање спецификација, и зато није могуће, као код програмских језика, њиме креирати (програмирати) пословну логику система, покренути процес или активирати операцију која није типа упита. Резултат евалуирања OCL израза је враћена вриједност. Ово значи да OCL израз не може да утиче на стање система, тј. не може да промени модел уз који је специфициран.

OCL је првобитно креиран као декларативни језик за описивање правила која се примјењују на UML моделе и у суштини је и сам представљао део UML спецификације, односно формалну екстензију језика UML. Данас се OCL примењује на све MOF метамоделе, укључујући и UML.

OCL је кључна компонента једног од најважнијих стандарда MDA приступа - трансформације модела. У [4] је представљен један приступ имплементацији трансформације модела који уводи појам уговора за трансформацију (оригинални назив на енглеском: *model transformation contracts*). Уговорима је омогућено специфицирање трансформације модела независно од

одабране платформе. Спецификација уговора за трансформацију је базирана на стандардним концептима UML-а і OCL-а. Циљ специфицираних уговора је да дефинишу шта ради трансформација модела, под којим условима може бити примењена и шта се може очекивати као резултат трансформације. У [4] су уговори представљени као важна компонента процеса развоја софтвера, заснованом на моделима, јер њихова примена може да унапреди специфицирање и документовање трансформација, валидацију и тестирање трансформација.

OMG група установила је QVT као језик за трансформацију модела. MOF QVT спецификација дата је у [14]. QVT интегрише концепте OCL 2.0. Структура QVT-а организована је у више нивоа. QVT дефинише три DSL језика:

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

Relations и *Core* представљају декларативне језике. *Relations* спецификације је могуће трансформисати у *Core* спецификације. *Relations* представља језик за спецификацију веза између MOF објеката и подржава њихово упаривање по задатом обрасцу. *Relations* дефинише трансформације између модела кандидата.

Core је декларативни језик који подржава упаривање над скупом променљивих, провером задовољености одређених услова у односу на скуп модела. *Core* спецификација је једнако моћна као *Relations*, али је нешто једноставније синтаксе. С друге стране спецификације *Relations* трансформације јесу сложеније, али су и дескриптивније.

Operational Mappings представља процедурални језик који проширује *Relations* і *Core*. Његова синтакса обухвата конструкционе елементе који се типично користе у процедуралним програмским језицима као што су нпр. петље, услови и сл. Језик *Operational Mappings* омогућава дефинисање трансформација применом потпуно процедуралног приступа који укључује специфицирање оперативних трансформација (*operational transformations*). У тзв. хибридном приступу, *Operational Mappings* омогућава надоградњу трансформација процедуралним операцијама које имплементирају релације.

ATLAS Transformation Language (ATL) представља алат и језик за трансформацију модела који подржава QVT спецификацију [5]. ATL је производ *Eclipse* фондације, непрофитабилног друштва чији пројекти су усмјерени ка развоју *open-source* система и алата. У области инжењерства заснованог на моделима (*Model-Driven Engineering* ili skraćeno MDE) ([2], [6], [16]), ATL, као *open-source* производ, има велики број корисника.

ATL трансформациони програм састављен је од правила која дефинишу како се елементи изворног модела препознају и обрађују на начин који резултира креирањем елемената циљних модела. Осим основне функције трансформације модела, ATL дефинише и додатни модел који пружа функционалност спецификације различитих упита на моделима.[5]

Један од приступа који подразумевају кључну улогу модела током развоја информационог система је *Model-Driven Software Development* (MDSD). Мање прецизан, али чешће употребљаван назив је и *Model-Driven Development* (MDD). Данас MDSD приступ има велику примену и значај који ће у будућности бити још већи, што представља природан наставак развоја програмирања каквог знамо данас.

Сваки конкретни домен примене је од великог значаја за примену MDSD приступа. Његов циљ је да дефинише доменски оријентисане (*domain-specific*) апстракције које ће бити расположиве за формално моделовање. Ово представља велики потенцијал у области аутоматске производње софтвера, што може значајно повећати продуктивност. Такође, ово може имате позитивног утицаја и на квалитет и на могућности одржавања софтвера. Примена доменски

оријентисаног приступа при моделовању, односно пројектовању и имплементацији софтвера детаљно је описана у [7].

Да би се успешно применио концепт "доменски оријентисаног модела" неопходно је испунити следеће услове:

- Морају бити дефинисани доменски оријентисани језици (*Domain Specific Language* - DSL) како би омогућили формулацију модела.
- Морају бити дефинисани језици који би специфицирали неопходне трансформације модела у програмски код.
- Неопходно је постојање компајлера, генератора и трансформатора који могу извршавати трансформације које генеришу програмски код на доступним платформама.

У поређењу са MDA приступом који се ограничава на алате засноване на UML-у, MDSD је мање рестриктиван. MDA се фокусира на стандардизацију модела и интероперабилност између алата за моделовање. MDSD се оријентише на креирање модула за развој софтвера, применљивих у пракси, који могу бити коришћени у различитим *Model-Driven* приступима, независно од одабраног алата.[18]

IIS*Case је алат за пројектовање информационих система и генерисање извршивих прототипова апликација. Он подржава пројектовање шеме базе података, генерисање одговарајућих SQL скрипти и трансакционих програма. Као такав, IIS*Case примењује неке од принципа *Model-Driven* приступа ([10], [11]).

Аналогно MDA алатима, пројектовање информационог система у IIS*Case-у засновано је на два типа модела: оном који не зависи од изабране платформе и оном који зависи. Систем се моделује на високо апстрактном нивоу и пројектант креира модел система без специфицирања детаља везаних за конкретну имплементацију. Због тога овај модел може бити специфициран као PIM модел [15]. На основу PIM модела, модул за генерисање прототипова апликација IIS*Case-а генерише и спецификације прототипова апликација информационих система која укључује детаље везане за конкретну платформу на којој ће прототип бити имплементиран (PSM модел).

4. Опис техничког решења

Софтверска апликација имплементирана је у клијент-сервер архитектури, где сервер представља произвољан систем за управљање базама података са којим се комуницира путем ODBC протокола. Апликација је реализована на програмском језику Јава, у окружењу Oracle JDeveloper верзија 10.1.3.0.2.

Процес генерисања трансакционог програма састоји се из два корака:

1. *Генерисање подшема типова форми*, што је предуслов за извршавање следећег корака, јер као резултат даје неопходне улазне информације на основу којих ће бити креирани SQL упити, путем којих ће генерисана апликација комуницирати са базом података.
2. *Генерисање апликације*. Овај корак укључује генерисање UIML документа са спецификацијом визуелних и функционалних аспеката апликативног система. Генератор покреће *Java Renderer* који генерисану спецификацију интерпретира као Јава апликацију. Изабрано програмско окружење је *Harmonia Incorporation® Java Renderer*. У питању је интерпретер који елементе UIML спецификације трансформише у *Java AWT/Swing* компоненте.

Овај поступак интерпретира само визуелне карактеристике корисничког интерфејса, не и сложене функционалности као што су нпр. функционалности које подржавају комуникацију са базом података. Зато је било неопходно надоместити овај недостатак увођењем Јава класа које подржавају сложене функционалности генерисаних прототипова.

UIML спецификације прототипова апликација садрже све што је потребно да би се елементи спецификације корисничког интерфејса и функционалности пројектованог апликативног система на правилан начин интерпретирали, односно трансформисали у одговарајуће програмске компоненте изабраног програмском окружења. Дефинисање пресликавања елемената пројектантске спецификације у програмске компоненте обухвата:

- спецификирање програмских компоненти трансакционог програма и њихових метода које имплементирају кориснички интерфејс и
- спецификирање програмских компоненти трансакционог програма и њихових метода које имплементирају особине и понашања свих компоненти корисничког интерфејса.

Током интерпретације, елементи UIML документа пресликавају се у одговарајуће програмске компоненте:

- *Java AWT/Swing komponente* које служе за интерпретацију свих елемената корисничког интерфејса и основних функционалности које су покривене методама *Java AWT/Swing* класа.
- *Custom Java komponente*, односно објекте, за ове потребе, креираних Јава класа које имплементирају напредне функционалности као што су нпр. оне које подржавају комуникацију са базом података, пренос параметара и њихових вриедности између екранских форми, итд.

У оквиру UIML спецификације веза између одговарајућих UIML компоненти и одговарајућих Јава класа остварена је кроз структуру `<peers>`. Ова структура омогућава дефинисање пресликавање класа и њихових метода, особина и догађаја у екстерне објекте који нису саставни део UIML спецификације. У оквиру њене подструктуре `<presentation>` дефинисан је речник дозвољених имена класа (специфицираних кроз UIML елеменат `<d-class>`) које могу бити коришћене за спецификирање елемената, особина и догађаја дефинисаних у UIML документу. За потребе имплементације генератора апликације креиран је посебан речник који проширује основни речник *Java 1.5 Harmonia 1.0*. као што је приказано на слици 1.

За потребе имплементације сложених функционалности које подржавају комуникацију са базом података развијене су Јава класе `ui.DBTable` и `ui.DBPanel`. Кроз спецификацију дату на слици 1, UIML елементи класа `DataTable` и `DataPanel` трансформишу се у објекте ових класа. Спецификација ове трансформација дефинисана је у оквиру UIML елемената `<d-class>`. Путем ових елемента задаје се листа дозвољених имена UIML класа. Додатно се дефинишу и класе изабраног програмског окружења (у овом случају Јава класе) у чије објекте се, спецификирањем опције `maps-to`, одговарајуће UIML компоненте пресликавају. Спецификацијом поделемента `<d-property>` и `<d-param>` остварује се веза између метода пресликаних Јава класа и одговарајућих UIML класа, тако да је омогућено да се позивом датих Јава метода мењају особине објеката датих UIML класа.

UIML елементи `DataTable` і `DataPanel` су најважнији структурни елементи UIML спецификације на нивоу типа компоненте. Приликом генерисања UIML спецификације, у зависности од спецификације визуелних атрибута типа компоненти (тачније спецификације начина приказа података – табеларни или путем панела), у оквиру UIML фрагмента који се односи на тип компоненте генерише се UIML елемент класе `DataTable` ili `DataPanel`. У току извршавања, ови елементи се интерпретирају као одговарајуће програмске компоненте, тј. објекти класе `yi.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>
```

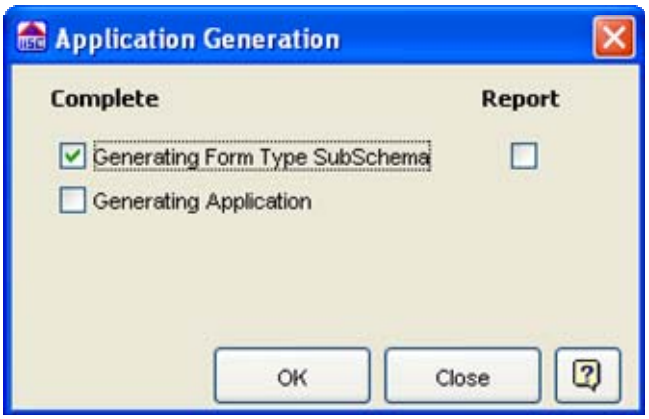
```
- <d-property id="url" maps-type="setMethod" maps-to="setUrl">
  <d-param type="java.lang.String" />
  </d-property>
- <d-property id="comp" maps-type="setMethod" maps-to="setComp">
  <d-param type="int" />
  </d-property>
- <d-property id="pr" maps-type="setMethod" maps-to="setPR">
  <d-param type="int" />
  </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>
```

Слика 1. IIS*Case UIML речник

4.1 Имплементација

У овом поглављу представљена је имплементирана апликација за генерисање прототипова апликација. Процес генерисања трансакционог програма састоји се из два корака (као што је приказано на слици 2):

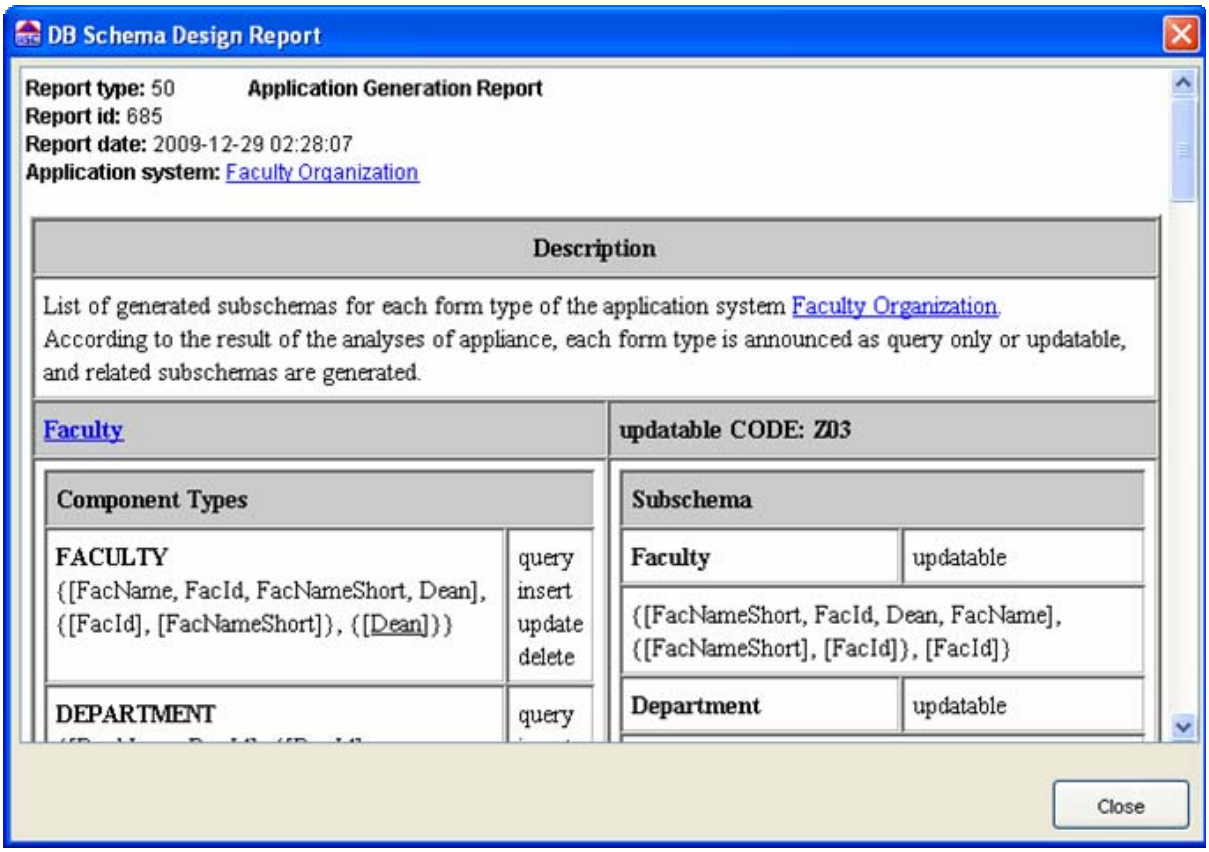
- генерисање подшема типова форми, и
- генерисање апликације.



Слика 2. Генерисање прототипа апликације

У првом кораку аутоматизованог поступка за генерисање трансакционих програма генеришу се подшеме типова форми. У оквиру овог корака имплементирани су алгоритми за генерисање скупа подшеме типова форми из класе за ажурирање и упите. Као резултат, добијају се скупови шема релација које припадају подшемама појединих типова форми, као и резултат анализе применљивости типова форми.

Уколико жели, пројектант може изабрати да се током овог корака генерише одговарајући извештај приказан на слици 3, који даје информацију о крајњим резултатима примене ових алгоритама. Дати извештај садржи списак свих типова форми са информацијама о дозвољеним операцијама на датом типу форме: *query*, *insert*, *update*, і *delete*. Дозвољене операције на типовима компоненти типа форме пројектант задаје путем форме за спецификацију типова компоненти. С друге стране, типовима форми придружени су одговарајући индикатори који су резултат анализе применљивости и начина њихове употребе.



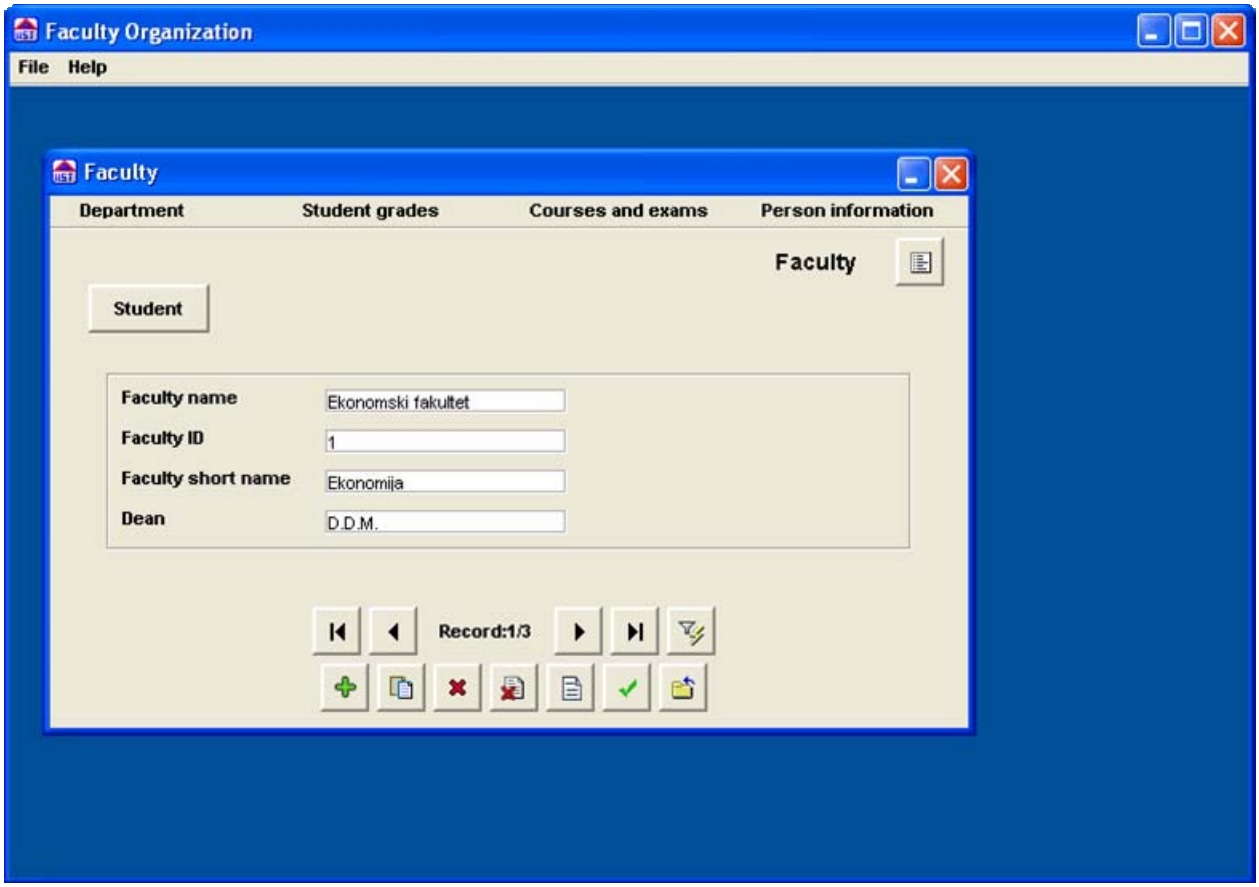
Слика 3. Извештај о генерисаним подшемама типова форми

Уз сваку од шема релација генерисаних подшема приказана је вредност: *updatable* ili *read only*, у зависности од поменутог индикатора. Приликом генерисања прототипа апликација, а у зависности од дозвољених операција на самим типовима компоненти, као и резултатима анализе применљивости и начина употребе типова форми, генеришу се екранске форме које ће подржавати одговарајуће операције за читање и ажурирање података.

Приликом генерисања трансакционог програма генерише се одговарајућа UIML спецификација која укључује све претходно дефинисане визуелне особине апликативног система. Ове особине пројектант задаје кроз:

- избор шаблона корисничког интерфејса,
- специфицирање визуелних атрибута, група поља и листа вредности и
- избор пословне апликације.

Шаблони корисничког интерфејса креирани су применом модула за креирање општег модела корисничког извештаја. Визуелна својства генерисаног трансакционог програма у смислу приказа поља за преглед и ажурирање података одређена су не само изабраним корисничким шаблоном већ и спецификацијама визуелних атрибутима, група поља и листа вредности. Њихова спецификација врши се применом модула за спецификацију визуелних атрибута. Након избора пословне апликације, која претходно мора бити специфицирана применом модула за специфицирање пословних апликација, генератор апликација, приликом генерисања прототипа апликације интегрише елементе свих поменутих спецификација у одговарајућу UIML спецификацију. Интерпретацијом ове спецификације, њени елементи пресликавају се у одговарајуће програмске компоненте: *Java AWT/Swing komponente* или *Custom Java komponente*. На тај начин формира се прототип пословне апликације, као пример на слици 4.



Слика 4. Пример прототипа апликације

5. Примена техничког решења

Техничко решење примењује се у едукацији студената академских основних студија и докторских студија на Факултету техничких наука у Новом Саду и Природно-математичком факултету Универзитета Црне Горе у Подгорици.

Поред тога, као део IIS*Case-a, модул за генерисање прототипа апликације, почео је да се користи експериментално, за развој софтверских решења и то на Економском факултету и Машинском факултету у Крагујевцу и у предузећу Застава аутомобили А.Д. Крагујевац.

6. Закључак

Техничко решење **Модул за генерисање прототипова апликација,** омогућава моделовање визуелног изгледа апликација информационих система независно од креирања њихових пројектних спецификација. Интегрисањем спецификације изабраног шаблона корисничког интерфејса и пројектне спецификације информационог система, добија се финална спецификација коју је могуће трансформисати у апликацију која задовољава сва пројектована правила пословања и визуелне захтеве које је пројектант дефинисао. Прототип апликативног система генерише се у Јава програмском окружењу.

Рад генератора трансакционих програма своди се на "позадинско" извршење тих алгоритама, а пројектанту се пружа увид само у крајњи резултат – прототип апликативног система, евентуално са придруженим "дијагностичким" извјештајем.

Посебан допринос овог техничког решења огледа се у имплементацији Јава класа које подржавају сложене функционалности генерисаних прототипова, ако и у имплементацији комплексних алгоритама за генерисање подшема типова форми неопходних за имплементацију функционалности читања и ажурирања података путем корисничког интерфејса генерисаних прототипова апликација.

Литература

[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] Bézivin J, *In Search of a Basic Principle for Model-Driven Engineering*, Novatica Journal, Special Issue, March-April 2004.

[3] Booch G, Brown A, Iyengar S, Rumbaugh J, Selic B, *An MDA Manifesto*, MDA Journal, May 2004.

[4] 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.

[5] Eclipse Foundation, *ATL (ATLAS Transformation Language) Project*, *ATL/User Guide*, 2010. http://wiki.eclipse.org/ATL/User_Guide

[6] Favre J.M, *Foundations of Model (Driven) (Reverse) Engineering: Models*, Dagstahl Seminar Proceedings 4101, 2005.

[7] Kelly S, Tolvanen J, *Domain-Specific Modeling: Enabling Full Code Generation*, Wiley-IEEE Computer Society Press, 2008, ISBN: 0470036664.

[8] Kleppe A, Warmer J, Bast W, *MDA Explained: The Model Driven Architecture: Practice and Promise*, Addison-Wesley, Boston, 2003.

[9] Luković I, *Automatizovano generisanje podšeme relacione baze podataka putem formi*, magistarski rad, Univerzitet u Beogradu, Elektro-tehnički fakultet, Beograd, 1993.

[10] 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.

[11] 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.

[12] 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

[13] 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

[14] 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/

[15] 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.

[16] Schmidt D. C, *Model-driven engineering*, IEEE Computer, Vol.39, No.2, 25-31, 2006.

[17] Singh Y, Sood M, *Models and Transformations in MDA*, First International Conference on Computational Intelligence, Communication Systems and Networks, 2009, pp. 253-258.

[18] Stahl T, Völter M, *Model Driven Software Development: Technology, Engineering, Management*, John Wiley & Sons, Ltd. 2006.

[19] World Wide Web Consortium (W3C), *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, W3C Recommendation, 2008. http://www.w3.org/TR/REC-xml/

УНИВЕРЗИТЕТ
У НОВОМ САДУ

ФАКУЛТЕТ
ТЕХНИЧКИХ НАУКА

Трг Доситеја Обрадовића 6, 21000 Нови Сад, Република Србија
Деканат: 021 6350-413; 021 450-810; Централа: 021 485 2000
Рачуноводство: 021 458-220; Студентска служба: 021 6350-763
Телефакс: 021 458-133; e-mail: ftndean@uns.ac.rs

Рецензија техничког решења

IIS*Case V7.0 – алат за пројектовање и генерисање база података и апликација – Модул за генерисање прототипова апликација

чији су аутори: Иван Луковић, Јелена Бановић, Александар Поповић, Соња Ристић, Павле Могин, Миро Говедарица и Иван Бекер. Техничко решење је реализовано у оквиру пројекта технолошког развоја ТР-13029 "Развој интелигентног окружења за подршку пројектовања и имплементације информационих система". Решење је развијано за потребе Факултета техничких наука у Новом Саду и предузећа Застава аутомобили А.Д. Крагујевац.

Наведено софтверско решење обезбеђује аутоматизацију процеса развоја софтверских апликација информационог система и представља део ширег окружења из класе *model-driven software development* алата, под називом *Integrated Information System Case* (IIS*Case), који се већ дужи низ година развија на Факултету техничких наука у Новом Саду. Намена *Модула за генерисање прототипова апликација* јесте да на основу претходно креираног општег модела корисничког интерфејса и платформски независних спецификација апликација информационог система, аутоматски генерише извршиве прототипове апликација неког информационог система.

На основу увида у техничку документацију овог софтверског решења, као и тестирањем самог софтвера, закључујем да је реч о оригиналном решењу које, у значајној мери, обезбеђује аутоматизацију процеса развоја апликација информационог система, и тиме омогућава значајно скраћење времена развоја и побољшање квалитета добијених софтверских решења.

У Новом Саду, 12. априла 2010. године.

Експерт - рецензент

Проф. др Драган Иветић,
Факултет техничких наука, Нови Сад

UNIVERZITET U BEOGRADU
FAKULTET ORGANIZACIONIH NAUKA
Jove Ilića 154, 11000 BEOGRAD
Tel: 011/3950-800 Faks: 011/461-221
http://www.fon.bg.ac.yu

PIB 100383934
Matični broj:07004044
Račun 840-1344666-69

Dekanat 493-492; 3950-823 Sekretar 3950-806, Studentska služba 3950-808, Poslediplomske studije 3950-828 Računovodstvo 3950-845

Рецензија

Техничко решење:

IIS*Case V7.0 – алат за пројектовање и генерисање база података и апликација – Модул за генерисање прототипова апликација

Пројекат:

техничко решење развијено у пројекту ТР-13029, са називом "Развој интелигентног окружења за подршку пројектовања и имплементације информационих система".

Корисници:

Факултет техничких наука у Новом Саду, Застава аутомобили А.Д. Крагујевац и друге заинтересоване институције.

Аутори:

Иван Луковић, Јелена Бановић, Александар Поповић, Соња Ристић, Павле Могин, Миро Говедарица и Иван Бекер.

Циљеви и функције:

Главни циљ је обезбеђење аутоматизације процеса развоја софтверских апликација информационог система. Решење је део *model-driven software development* алата, под називом *Integrated Information System Case* (IIS*Case), који се развија на Факултету техничких наука у Новом Саду. *Модул за генерисање прототипова апликација* користи, као улазне спецификације, претходно креиране спецификације апликација и трансакционих програма информационог система, које се дефинишу путем платформски независних и проблемски оријентисаних концепата. Те спецификације комбинују се са, такође, претходно креираним општим моделом корисничког интерфејса. На излазу, модул генерише извршиве прототипове апликација у оквиру развијаног информационог система.

На основу извршеног прегледа овог софтвера, закључујем да је у питању оригинално решење које се може поредити са сличним решењима у свету, али и поседује позитивне особине, карактеристичне само за ово решење. На тај начин, омогућено је унапређење процеса развоја информационих система и укључивање у тај развој пројектаната који не морају поседовати високо формална експертска знања из области пројектовања софтвера.

У Београду, 19. априла 2010. године.

Рецензент

Проф. др Владан Девеџић,
Факултет организационих наука,
Београд