Зборник радова Факултет техничких наука, Нови Сад

UDK: 004.41
DOI: https://doi.org/10.24867/04BE32Andjelic

ЈЕЗИК И ОКРУЖЕЊЕ ЗА МОДЕЛОВАЊЕ И ГЕНЕРИСАЊЕ ВЕБ СТРАНИЦА ИНФОРМАЦИОНИХ СИСТЕМА ОПШТЕ НАМЈЕНЕ

LANGUAGE AND FRAMEWORK FOR MODELING AND GENERATING WEB PAGES OF GENRAL PURPOSE INFORMATION SYSTEMS

Биљана Анђелић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом раду представљени су језик за моделовање веб страница информационих система опште намјене и генератори прототипова веб страница. Посебна пажња усмјерена је на моделовање валидационих правила корисничких форми. За развој meta-modela коришћено је окружење EMF, а окружење Xtend за развој текстуалне синтаксе намјенског језика.*

Abstract – *In this work a domain-specific language for modeling web pages of general-purpose information systems is presented together with a generator of web page prototypes. The focus of this work is on modeling the user form validation. The EMF environment was used for the creation of the meta-model. The Xtend environment was used for developing the textual syntax of the domain-specific language.*

Кључне речи: *Информациони системи, намјенски језици, развој заснован на моделима, формална спецификација веб страница*

1. УВОД

Глобализација која се увукла у све поре људског живота и дјеловања намеће многе захтјеве свјетској привреди и онима који желе бити дио ње. Потреба да производи и услуге у најкраћем року буду доступни подједнако у свим дјеловима свијета намеће висок ниво умрежености коју је немогуће замислити без употребе информационих система.

Информациони систем (енг. *Information System, IS*) као модел реалног система има неколико задатака: обухват, складиштење, пренос, презентовање и обраду података као и аутоматизацију управљачких функција. Обзиром да обухват података представља улазни корак за реализацију свих осталих задатака неопходно је обезбиједити једноставан обухват података, који омогућава кориснику унос искључиво валидних и добро форматираних података. Најчешће, обухват података у оквиру информационог система обавља се путем корисничких форми. У овом раду биће описан језик и одговарајући алати који омогућавају генерисање веб страница информационих система опште намене са посебном пажњом усмјереном ка корисничким формама које уједно пружају први вид валидације података.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији је ментор био др Владимир Димитриески, доцент.

Употребом традиционалних методологија развоја софтвера клијенти су били у могућности да виде функционалан систем тек на крају процеса развоја. То је могло да буде прекасно ако имплементирано рјешење није задовољавало захтјеве и очекивања. Примјеном агилних методологија развоја софтвера са брзим итерацијама, развојем прототипова као и честом комуникацијом са клијентима овај проблем би се могао ако не у потпуности уклонити онда бар значајно умањити.

Увођење моделом вођеног развоја софтвера (енг. *Model-Driven Software Engineering, MDSE*) и намјенских језика представља погодан оквир за успјешну реализацију комуникације између клијената и стручњака за информационе технологије (енг. *Information Technology, IT*). Коришћењем намјенских језика омогућава се клијентима да помоћу познатих концепата из домена учествују у креирању модела информационог система. У овом раду представљено је рјешење које би требало да превазиђе касну детекцију неусклађености захтјева и очекивања са имплементираним рјешењем. Из тог разлога настао је намјенски језик *fvalDSL* који омогућава развој веб страница информационих система опште намјене путем концепата блиских пројектантима *IS* и концепата графичког корисничког интерфејса.

2. ПРЕГЛЕД ПОСТОЈЕЋЕГ СТАЊА У ОБЛАСТИ

Постојање информационих система датира још од времена прије појаве рачунара. Појава рачунара олакшала је развој и одржавање информационих система, али је наметнула и нове захтјеве у смислу брзине обраде, капацитета, повјерљивости и приказа података.

2.1 Традиционални приступи

Велики број традиционалних методологија развоја *IS* засноване су на моделима. У оквиру ових приступа модели се посматрају као важан дио процеса и предлаже се употреба различитих врста модела да би се представиле различите фазе развоја *IS*-а [1]. Једна од битнијих фаза у развоју *IS* је концептуално моделовање. Очекивани резултат ове фазе је креирање модела независних од платформе (енг. *Platform Independent Model, PIM*) на којој се систем извршава. Модели креирани употребом традиционалне методологије развоја софтвера углавном служе као документација. За разлику од традиционалних приступа, рјешење описано у овом раду омогућава да се креирани модел користи као централни артифакт за генерисање функционалног прототипа апликације.

2.2 MDE приступ

Увођење развоја софтвера вођеног моделима (енг. *Model Driven Engineering, MDE*) омогућава превазилажење недостатака који су се испољили у традиционалном приступу развоја софтвера. Креирани модели постају извршиви артифакти што је искоришћено и у овом раду.

2.3 Примјена намјенских језика

Већина алата за спецификацију корисничког интерфејса представљају графичке алате који су намјењени дизајнерима. Они омогућавају корисницима да на брз и једноставан начин специфицирају графички кориснички интерфејс. Постоје и намјенски језици за спецификацију *IS* који омогућавају опис и серверске и клијентске стране као што је намјенски језик за моделовање информационих система опште намјене *WISL (*енг. *Web Information System Language)* [2].

За разлику од намјенског језика *WISL*, намјенски језик *fvalDSL* посједује знатно шири скуп правила која се могу примјенити на атрибуте ентитета. Осим правила која важе над атрибутима, *fvalDSL* посједује и међуатрибутска ограничења. Када се *fvalDSL* упореди са дизајнерским алатима за генерисање графичког корисничког интерфејса може се примјетити да има знатно скромније могућности при тачном позиционирању компоненти као и подешавању њихових димензија. Неке од предности намјенског језика *fvalDSL* у односу на дизајнерске алате јесте и то што је конкретна синтакса реализована као текстуална што га чини једноставног за коришћење *IT* стручњацима. Постојање предефинисаних компоненти графичког корисничког интерфејса знатно скраћује вријеме развоја јер није нужно за сваки атрибут навести компоненту којом се представља.

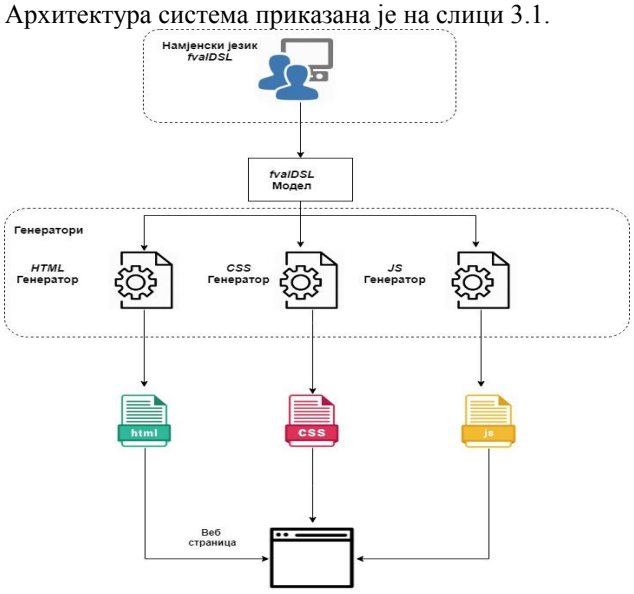
2.4. Интеракција човјек-рачунар

Седамдесетих година двадесетог вијека, појављује се парадигма са називом: „прозори, иконице, мени, показивачи“ (енг. *Window, Icon, Menu, Pointer, WIMP*) која пружа сасвим нови начин интеракције између човјека и рачунара. Овај вид интеракције корисницима је донио многе погодности као што су мулти-таскинг, једноставније коришћење за већину корисника, нема више памћења команди и контекстна помоћ. Поред погодности које је *WIMP* парадигма донијела постоје и недостаци од којих се посебно истиче смањење брзине интеракције напредних корисника.

Да би се превазишао наведени недостатак приликом попуњавања корисничких форми потребно је водити рачуна како су поља за унос груписана и распоређена у зависности од врсте података који се преко њих уносе, времена и мјеста када се приказују поруке о грешци и другим параметрима који могу имати утицаја. Приликом креирања генератора посебна пажња усмјерена је на избор предефинисаних компоненти. Предефинисане компонете за атрибуте се креирају на основу типа атрибута и осталих ограничења тако да буду у складу са тренутно најбољом праксом из области човјек-рачунар. Међутим, пружена је и могућност да *IT* стручњак изабере произвољну компоненту којом ће представити атрибут уколико му предефинисана компонента из неког разлога не одговара.

3. АРХИТЕКТУРА СИСТЕМА

Архитектура система приказана је на слици 3.1.



Слика 3.1 - *Архитектура система*

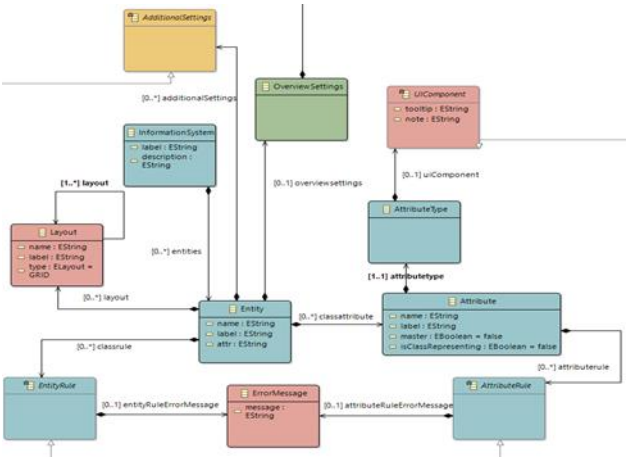
Систем се састоји од два модула: модул са намјенским језиком *fvalDSL* и модул са генераторима кода. Модули су на слици представљени правоугаоницима испрекиданих линија.

Први модул обухвата намјенски језик *fvalDSL* који посједује концепте блиске домену за креирање информационих система и нотацију која омогућава да се концепти инстанцирају. Излаз из првог модула је *fvalDSL* модел који уједно представља и улаз у други модул.

Други модул чине генератори кода. Описани систем посједује три генератора кода. Сва три генератора имају исти улаз, *fvalDSL* модел, а сваки од њих креира по један фајл као излаз. Генератор *HTML*-а (енг. *Hypertext Markup Language*) као излаз даје *HTML* фајл са статичким садржајм. Генератор *JavaScript*-а генерише *JavaScript* фајл са описом динамике, а генератор *CSS*-а (енг. *Cascading Style Sheets*) као резултат даје *CSS* фајл са описом стила.

4. НАМЈЕНСКИ ЈЕЗИК ЗА МОДЕЛОВАЊЕ ВЕБ СРАНИЦА ИНФОРМАЦИНИХ СИСТЕМА ОПШТЕ НАМЈЕНЕ

За развој апстрактне синтаксе коришћен је мета-мета-модел Ecore који омогућава представљање концепата из домена и веза које владају међу њима. На слици 4.1 представљен је најбитнији дио мета-модела намјенског језика fvalDSL. Централни концепт је Entity. Сваки ентитет посједује назив, а може да посједује лабелу и опис. Поред наведених обиљежја ентитет посједује атрибуте, опис изгледа корисничке форме, изглед табеле за приказ додатих записа и скуп међуатрибутских ограничења. У мастер раду из кога је овај рад произашао може се видјети комплетан мета-модел са конкретизацијама концепата EntityRule, AttributeRule, uiComponent и AdditionalSetting као и концептима које референцира концепт OverviewSettings.



Слика 4.1 *Централни дио мета-модела намјенског језика fvalDSL*

Поред ограничења која су јасно видљива на основу мета-модела као што су ограничења домена за вриједности атрибута и садржавања за описани језик постоје ограничења која није могуће специфицирати на нивоу мета-модела, а неопходна су да би се задовољила семантика. Обзиром да Layout може да садржи друге Layout-е, на једном нивоу сви Layout-и морају да буду истог типа што се не може видјети из мета-модела креирано је OCL (енг. Object Constraint Language)[3] правило приказано на листину 4.1.

```
Invariant
allLayoutsOnOneLevelMustBeSameType ('All
layouts on one level must be same type.'):
layouts -> forAll(fL1: Layout, fL2: Layout|
      if fL1.oclIsTypeOf(fL2.oclType())
      then true
      else false endif);
```

Листинг 4.1 *Сви layout-и на једном нивоу морају бити истог типа*

Конкретна синтакса за намјенски језик *fvalDSL* имплементирана је као текстуална. На листингу 4.2. приказан је дио информационог система који омогућава регистрацију корисника. Модел почиње називом система што је у приказаном случају *IS*, потом слиједи навођење лабеле и описа који су опциони. Први ентитет уједно и једини у описаном примјеру је *user*. Ентитет *user* посједује три атрибута. Атрибут *username* је атрибут који представља корисничко име. Корисничко име је знаковни тип података, обавезно је унијети његову вриједност и први карактер мора да буде велико слово, након тога слиједе мала слова. Минимална дужина корисничког имена је два карактера, а максимална 40 карактера. У случају да ограничења нису задовољена исписиваће се подразумјеване поруке које постоје за свако ограничење. Атрибут ће бити представљен компонентом *TextInput*. Атрибути *password* и *confirmPassword* омогућавају унос и потврду лозинке. За оба атрибута важе иста ограничења. Лозинка мора да задовољи прописани формат што у овом примјеру значи да мора да има најмање једно мало слово, једно велико слово и једну цифру. Минималан број карактера које лозинка мора да има је 7, а максималан 40. Порука која се исписује уколико формат није задовољен специфицирана је помоћу атрибута *Error Message* у оквиру ограничења *StringPattern*. Атрибути *password* и *confirmPassword* су представљени компонентом Password. Омогућен је приказ карактера у изворном облику подешавањем параметра *Visible* на вриједност *true*. Да компонента графичког корисничког интерфејса није експлицитно наведена подразумјевало би се да се користи компонента *TextInput*. Након навођења атрибута наведена су и међу-атрибутска ограничења. Прво је наведено ограничење *Equals* које специфицира да вриједности атрибута *password* и *confirmPassword* морају бити једнаке. Друго специфицирано правило *NotEquals* дефинише да *password* и *userName* не смију имати исту вриједност. Обзиром да распоред компоненти на форми није наведен подразумјева се да се компоненте приказују једна испод друге.

```
IS
{
  label "Information system"
  description "Short description"
  user{
   label "User"
   userName{
    Data type String
    Required
    StringPattern '[A-Za-z0-0]+'
    MaxLength 40
    MinLength 2
   }
   password{
    Data type String
    Password {Visible true}
    Required
    StringPattern '(?=.*\\d)(?=.*[a'-z])(?=.*[A-Z])'
    {
     Error message "Password must contain at  least one upper case letter,
     lower case latter and digit.
    }
    MaxLength 40
    MinLength 7
   }
   confirmPassword{
    Data type String
    Password {Visible true}
    Required
    StringPattern '(?=.*\\d)(?=.*[a-z])(?=.*[A- Z])'
    {
      Error message "Password must contain at least one upper case letter,
     lower case latter  and digit. "
    }
    MaxLength 40
    MinLength 7
   }
   Equals{
    attributes(password, confirmPassword)
    Error message {"Passwors must match."}
   }
   NotEquals {attributes(userName,password)}
  }
}
```

Листинг 4.2 - *Модел дијела информационог система за регистрацију*

5. ГЕНЕРАТОР КОДА

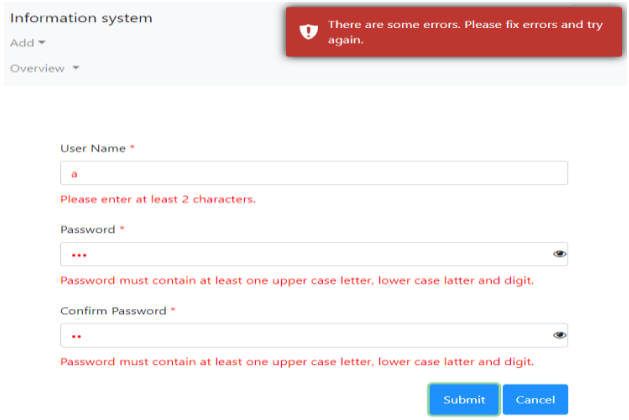
Као што је наведено у поглављу 3, постоје три гренератора која имају исти улаз, али дају три излаза који могу да постоје независно. Уз одређене модификације могу се искористити и у другим рјешењима ако се испоштује начин на који се именују класе и *ID*-ијеви *HTML* елемената

У случају *HTML* генератора, приликом парсирања садржаја учитаног модела, генератор проналази елементе који представљају компоненте графичког корисничког интерфејса и трансфомише их у елементе *HTML*-а. Међутим, постоји могућност да за сваки концепт који треба бити приказан није специфицирана компонента којом ће бити представљена. Тада се приступа додатној провјери атрибута елемента како би се представио компонентом која ће корисницима омогућити најинтуитивнију и најбржу

интеркацију. Прво се врши провјера типа података, а затим се приступа провјери и осталих ограничења која су специфицирана над елементом.

У случају имплементираног *CSS* генератора за намјенски језик *fvalDSL* стил је дефинисан у екстерном фајлу, што чини јаснији и читљивији и *CSS* и *HTML* код. Прво се дефинише подразумјевани стил за елементе као што су форма, панели и табеле за преглед унешених торки, а затим се дефинише специфичан стил за елементе уколико је он специфициран на нивоу модела. Уколико није специфициран или уколико за неки специфичан елемент није могуће специфицирати стил помоћу *fvalDSL*, могуће га је ручно додати или измијенити, навођењем селектора елемента за који се дефинише стил.

У имплементираном *JavaScript* генератору прво се генеришу колекције које садрже мета податке о ентитетима који постоје у информационом систему, затим се врши генерисање помоћних функција које су исте за све ентитет. На крају се врши генерисање валидатора за сваки ентитет посебно на основу описаног модела.



Слика 5.1 - *Дио информационог система за регистрацију корисника*

На слици 5.1 је приказан изгенерисани дио информационог система специфициран на листингу 4.1. Комплекснији примјер апликације је описан у мастер раду из кога је овај рад проистекао.

6. ЗАКЉУЧАК

У овом раду описан је развој веб страница за информационе системе опште намјене путем намјенског језика *fvalDSL*. За разлику од осталих алата који омогућавају развој крокија за графички кориснички интерфејс, који су углавном креирани за дизајнере, посједује текстуалну синтаксу која га чини ближим програмерима који су се навикли да програмирају у текстуалним програмским језицима. Приликом развоја модела путем намјенског језика *fvalDSL* није нужно за сваки атрибут навести компоненту графичког корисничког интерфејса. За сваки атрибут постоји предефинисана компонента којом ће бити представљен у складу са типом података и скупом ограничења која важе над наведеним атрибутом, као и добрим праксама развоја дефинисаним у истраживачкој области интеракције човјек-рачунар.

Поред концепата који представљају компоненте графичког корисничког интерфејса посједује и концепте који су блиски *ER* моделу као што су ентитети, атрибути и везе међу ентитетима. Намјенски језик *fvalDSL* такође посједује и концепте који се односе на валидацију података како на нивоу атрибута, што је врло блиско ограничењу домена, тако и концепте за валидацију међу-атрибутских ограничења која су врло блиска ограничењима појаве типа ентитета из *ER* модела података. Наведени концепти омогућавају развој прототипа који може да врши симулацију и серверске стране што корисиницима даје јаснији увид у понашање информационог система која си се развија.

Да би се помоћу *fvalDSL* могао описати цјелокупан информациони систем прво би се требало усмјерити на проширење мета-модела и увођењем концепата који би пружили подршку за ауторизацију, јер у тренутној имплементацији корисник има могућност да додаје, мијења и брише све ентитете који егзистирају у информационом систему. Потребно је развити и генераторе за серверски дио апликације.

Пожељно би било развити и генераторе за десктоп верзију апликације пошто није ријеткост да постоји потреба да један информациони систем буде развијен у више технологија. Ово би смањило могућност појаве разлике између веб и десктоп верзије информационог система.

7. ЛИТЕРАТУРА

[1] Marco Branbilla, Jordi Cabot, Manuel Wimmer, Model-Driven Software Engineering in Practice Morgan & Claypool Publishers 2012.

[2] Војислав Ђукић, Језик и окружење за моделовање и генерисање информационих система опште намене, Факултет техничких наука, 2015.

[3] Object Constraint Language, https://www.omg.org/spec/OCL/About-OCL/, Приступљено 15.2.2019.

Кратка биографија:

Биљана Анђелић рођена је 1994. године у Бања Луци, Босна и Херцеговина. Факултет техничких наука уписала је 2013 године. Бечелор рад из области Електротехника и рачунарство – Рачунарске науке и информатика одбранила је 2017. год. Мастер рад одбранила је 2019. године.