Novi Sad J. Math.
Vol. 38, No. 1, 2008, 43-57

MODELLING AND IMPLEMENTATION OF THE APPLICATION FOR STUDENTS ENROL FORM UPDATE[1]

Miodrag Mirković[2], Dušan Surla[3]

Abstract. For a semester enrolment, it is necessary to enclose entry form for student enrolment prescribed by the law (ŠV-20 form). The paper describes the modelling and implementation of a web application for semester enrolment based on an XML document. Software architecture is multi-layered. At business logic layer, an update is realized by the object model of the XML schema of ŠV-20 form. Different systems for data or document management can be used for XML documents storage.

AMS Mathematics Subject Classification (2000): 68U35

Key words and phrases: XML Schema, student service information system

1. Introduction

XML technologies are increasingly used for designing information systems. At the University of Novi Sad, XML technologies are applied in the development of the bibliographic information system BISIS [1, 2, 3], as well as the student service information system at the Department of Life Sciences.

In the frame of the development of the student service information system, the documents of the student service such as syllabi, register, matriculation book and enrolment entry form are analyzed and modelled in XML Schema language. Some proposals of XML schemas for these documents are given in [4, 5, 6, 7]. Based on the XML schema given in [7], modelling and implementation of a Web application for updating and processing student enrolment entry form are described in this paper. The application is created in Java environment. The architecture of the application can use different systems for data storage and it has interfaces for communication with the other student service subsystems. This enables the application to be used by different student's service software systems. Besides, the structure of the document that is used for data exchange with Statistical Office of the Republic of Serbia is adopted.

[1] This paper is a part of the research project "Abstract models and applications in computer science", supported by the Ministry of Science of the Republic of Serbia (Project No. 144017)

[2]M&I Systems, Co., Ćirila i Metodija 13A, Novi Sad 21000, Serbia, e-mail: miodrag.mirkovic@mi-system.co.yu

[3]Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad, Trg D. Obradovića 4, Novi Sad 21000, Serbia, e-mail: surla@uns.ns.ac.yu

2. Modelling of the system for student enrolment

In this section, information requests of the system for student enrolment are described. The diagram and descriptions of use cases, corresponding sequence diagrams, as well as the diagram of classes in UML notation 2.0 [8, 9] are presented. The model is designed by Embarcadero Describe Enterprise 6.1.7 [10] developmental environment that supports the UML 2.0.

2.1. Diagram of the cases of use *Student enrolment*

The use case diagram *Student enrolment* describes functionality of the system on the occasion of filling in the enrolment form (Figure 1). The student is enrolled at a university via the use case *Initial enrolment*, while a semester enrolment (other than the initial one) is done by the use case *Semester enrolment.*

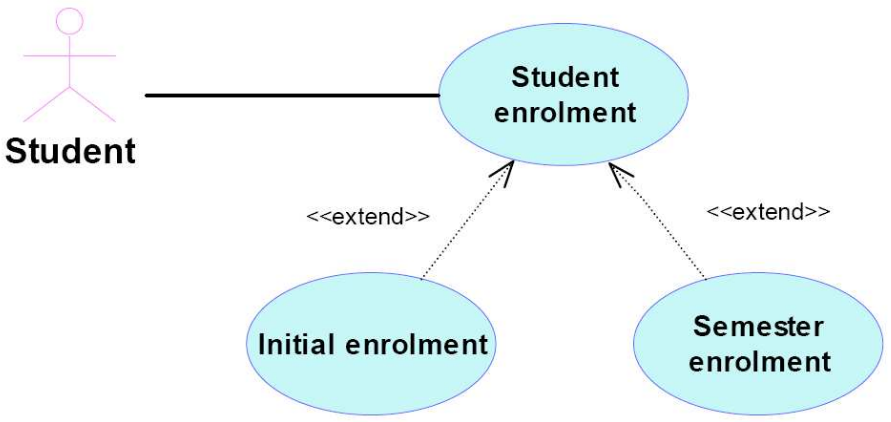


Figure 1: Student enrolment

Case of use: Student enrolment

Short description: Student enrols semester

Participant: Student

Conditions that must be fulfilled before performing: Student has fulfilled the condition for semester enrolment.

Description: Student enrols semester. If the student enrols at a faculty, the case of use *Initial enrolment* is carried out; otherwise, the case of use *Semester enrolment* is carried out.

Exceptions: None

Conditions that must be fulfilled after performing: Student has enrolled the semester.

Case of use: Initial enrolment

Short description: The student is enrolled at a faculty, mostly at the first semester for the first time. The student enters answers to the questions prescribed in the student enrolment form.

Participant: Student

Conditions that must be fulfilled before performing: Student has fulfilled the condition for enrolment at a faculty.

Description: Student is enrolled at a faculty, mostly at the first semester for the first time. Student gets the matriculation book number. Data on the school year and ordinal number of enrolment form are generated automatically. Register number, name of the faculty, university and place of school are generated automatically for the corresponding faculty. Student enters the following data: surname, name of mother or father and name, citizen's unique register number (JMBG), department, section, sex, year of birth, place of birth (place – settlement and municipality or foreign country), home address (place – settlement and municipality or foreign country, street and house number, and telephone number), place of residence during the studies (place – settlement and municipality or foreign country, street and house number, and telephone number), citizenship, nationality, previously gained formal education (full title of school, municipality or foreign country, foreign language learned at school and the year of school completion), year of studies that is being enrolled, type of payment of fees, answers to the questions if the student is enroling on this school year again, in which calendar year the student enrolled on this school for the first time, education of parents (father and mother), occupation of parent – guardian, job title (of parent – guardian or student) and answer to the question if the student is employed. [*Exception:* Wrongly entered citizen's unique register number (JMBG) of the student.], [*Exception:* Wrongly entered year]

Exceptions:
[*Exception*: Wrongly entered citizen's unique register number (JMBG) of the student] It is necessary that student enters his/her JMBG number again.
[*Exception*: Wrongly entered year] It is necessary that student enters requested year again.

Conditions that must be fulfilled after performing: Student has enrolled on the faculty.

Case of use: Semester enrolment

Short description: Student enrols on a semester (not at a faculty). This means, student modifies answers to the questions prescribed in the student enrolment entry form that were filled in when previous semester was enrolled.

Participant: Student

Conditions that must be fulfilled before performing: Student has fulfilled conditions for semester enrolment.

Description: Student enrols on a semester. Except for the academic year and ordinal number of the enrolment form that are generated automatically, student modifies the corresponding data in accordance with the data that the student filled in when the previous semester was enrolled.

Exceptions: None

Conditions that must be fulfilled after performing: Student has enrolled the semester.

2.2. Diagram of classes

Figure 2 shows the diagram of classes of a system for student enrolment. The class *Faces Servlet* represents the central servlet of the application. The class *StudentEnrolmentEntryFormDocument* represents an object model of the schema that is generated on the basis of XML schema of entry form for student enrolment by XML Beans [11], while the class *StudentEnrolmentEntryForm* represents the 'root' element of the object model. Classes *Heading*, *FilledInByStatistics* and *FilledInByStudent* represent the parts of the object model: part Heading, that is filled in by Statistics and part that is filled in by Student. Other classes represent questions from the previously mentioned parts of the object model with corresponding set and get methods.

2.3. Sequence diagrams

Based on the previously shown diagrams, the corresponding sequence diagrams that describe the specific scripts of the cases of use are created.

Sequence diagram *Initial enrolment.* Figure 3 shows a sequence diagram of the case of use *Initial enrolment.* Student communicates with the middle layer of the application via the user interface that is represented by screen forms. The messages being exchanged during the realization of the script of this sequence diagram are the following:

1. generateSchoolYear() – current school year is generated automatically;
2. generateOrdinalNumberOfEntryForm () – entry form ordinal number is assigned automatically;
3. enterSurname() – student enters surname;
4. enterNameOfMotherOrFather() - student enters mother's or father's name;
5. enterName() - student enters name;
6. enterJMBG() – student enters citizen's unique register number;

7. check() – the length and content of the entered JMBG is checked. If the length of the entered data is not 13, or if the entered data is not consisted only of Figures, the message 6 is exchanged again. Otherwise, the message 8 is exchanged.
8. enterDepartment () – student selects the name of the department;
9. enter Section() - student selects the name of the section;
10. enterSex() – student selects sex;
11. enterYearOfBirth() – student enters the year of birth;
12. check() – the length and content of the entered year is checked. If the length of the entered data is not 4, or if the entered data is not consisted only of figures, or if the first two figures of the entered data are not 19 or 20, the message 11 is exchanged again. Otherwise, the message 13 is exchanged.
13. enterPlaceOfBirth() – student enters the place of birth;
14. enterMunicipalityOrForeignCountryOfBirth() – student selects municipality or foreign country of birth;
15. enterHomeAddress() – student enters place of residence;
16. enterMunicipalityOrForeignCountry() – student selects municipality or foreign country of residence;
17. enterStreetAndHouseNumber() – student enters street and house number of residence;
18. enterTelephoneNumber() – student enters the telephone number registered at the address of residence.
19. enterPlaceOfResidenceDuringStudies() – student enters place of residence during studies;
20. enterMunicipalityOrForeignCountryOfResidenceDuringStudies() – student selects municipality or foreign country of residence during studies;
21. enterStreetAndHouseNumberOfResidenceDuringStudies() – student enters the street and house number of residence during studies;
22. enterTelephoneNumberOfResidenceDuringStudies() – student enters telephone number registered at the address of the residence during studies.
23. enterCitizenship () – student selects data on citizenship;
24. enterNationality () – student selects nationality;
25. enterPreviouslyGainedFormalEducation() – student selects previously finished school;
26. enterMunicipalityOrForeignCountryOfPreviouslyFinishedSchool() – student selects municipality or foreign country of previously finished school;
27. enterForeignLanguageLearnedAtSchool() – student enters foreign language learnt at school;
28. enterYearOfSchoolCompletion() – student enters the year of school completion;
29. check() – the length and content of entered year is checked. If the length of the entered data is not 4, or if the entered data is not consisted only of figures, or if the first two figures of the entered data are not 19 or 20, the message 28 is exchanged again. Otherwise, the message 30 is exchanged.

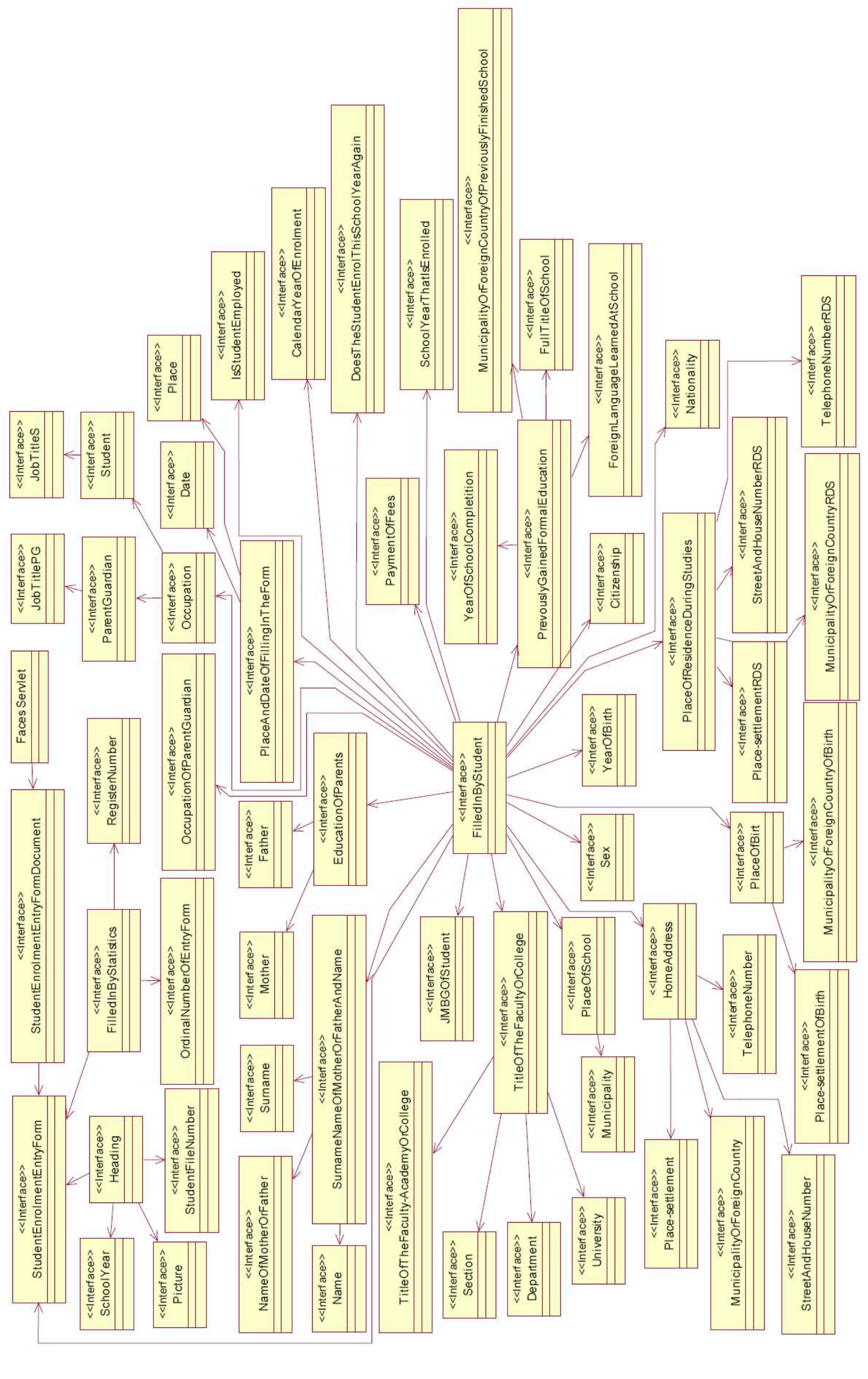

Faces Servlet
<<Interface>> StudentEnrolmentEntryFormDocument
<<Interface>> StudentEnrolmentEntryForm
<<Interface>> JobTitleS
<<Interface>> JobTitlePG
<<Interface>> Student
<<Interface>> ParentGuardian
<<Interface>> RegisterNumber
<<Interface>> FilledInByStatistics
<<Interface>> Heading
<<Interface>> SchoolYear
<<Interface>> Picture
<<Interface>> StudentFileNumber
<<Interface>> OrdinalNumberOfEntryForm
<<Interface>> Date
<<Interface>> Place
<<Interface>> Occupation
<<Interface>> OccupationOfParentGuardian
<<Interface>> PlaceAndDateOfFillingInTheForm
<<Interface>> IsStudentEmployed
<<Interface>> CalendarYearOfEnrolment
<<Interface>> DoesTheStudentEnrolThisSchoolYearAgain
<<Interface>> SchoolYearThatIsEnrolled
<<Interface>> MunicipalityOfForeignCountryOfPreviouslyFinishedSchool
<<Interface>> FullTitleOfSchool
<<Interface>> ForeignLanguageLearnedAtSchool
<<Interface>> PaymentOfFees
<<Interface>> YearOfSchoolCompletition
<<Interface>> PrevouslyGainedFormalEducation
<<Interface>> Citizenship
<<Interface>> Nationality
<<Interface>> TelephoneNumberRDS
<<Interface>> StreetAndHouseNumberRDS
<<Interface>> MunicipalityOrForeignCountryRDS
<<Interface>> PlaceOfResidenceDuringStudies
<<Interface>> Place-settlementRDS
<<Interface>> FilledInByStudent
<<Interface>> YearOfBirth
<<Interface>> Sex
<<Interface>> PlaceOfBirt
<<Interface>> MunicipalityOrForeignCountryOfBirth
<<Interface>> Place-settlementOfBirth
<<Interface>> EducationOfParents
<<Interface>> Father
<<Interface>> Mother
<<Interface>> SurnameNameOfMotherOrFatherAndName
<<Interface>> Surname
<<Interface>> NameOfMotherOrFather
<<Interface>> Name
<<Interface>> JMBGOfStudent
<<Interface>> TitleOfTheFacultyOrCollege
<<Interface>> TitleOfTheFaculty-AcademyOrCollege
<<Interface>> Section
<<Interface>> Department
<<Interface>> University
<<Interface>> PlaceOfSchool
<<Interface>> Municipality
<<Interface>> HomeAddress
<<Interface>> TelephoneNumber
<<Interface>> Place-settlement
<<Interface>> MunicipalityOrForeignCounty
<<Interface>> StreetAndHouseNumber

Figure 2: Diagram of classes of the system for student enrolment

30. enterSchoolYear() – student selects the school year that is being enrolled;
31. enterPaymentOfFees () – student selects the type of financing of the studies;
32. enterRepeatedEnrolment() – student selects the answer to the question if the same school year is being enrolled again;
33. enterCalendarYearOfFirstEnrolment() – student enters calendar year in which this school was enrolled for the first time;
34. check() – the length and content of entered year are checked. If the length of the entered data is not 4, or if the entered data is not consisted only of figures, or if the first two figures of the entered data are not 19 or 20, the message 33 is exchanged again. Otherwise, the message 35 is exchanged.
35. enterEducationOfFather() – student selects father's education;
36. enterEducationOfMother() - student selects mother's education;
37. enterOccupationOfParentGuardian() – student selects occupation of parent/guardian;
38. enterJobTitleOfParentGuardian() - student selects job title of parent/guardian (condition of message exchange: student is financed by the parent that is active);
39. enterJobTitleOfStudent() – student selects his own job title (condition of message exchange: student supports himself);
40. enterEmployment() – student selects the answer to the question if he is employed;
41. sequence diagram *Creation of XML file.*

Sequence diagram *Creation of XML file.* The messages that are being exchanged by the execution of the sequence diagram (Figure 4) script are the following:

1. form() – student requests XML file formation;
2. createStructure() – the instance of object schema model that is appropriate to the XML schema of student enrolment entry form is created;
3. FillInHeading() – school year and student matriculation book number are entered into the instance of object schema model;
4. FillInStatistics() – register number and ordinal number of entry form are entered into the instance of object schema model;
5. FillInStudent() – data on student are entered into the instance of object schema model;
6. Save() – instance of object schema model of student enrolment entry form is stored in respective data.

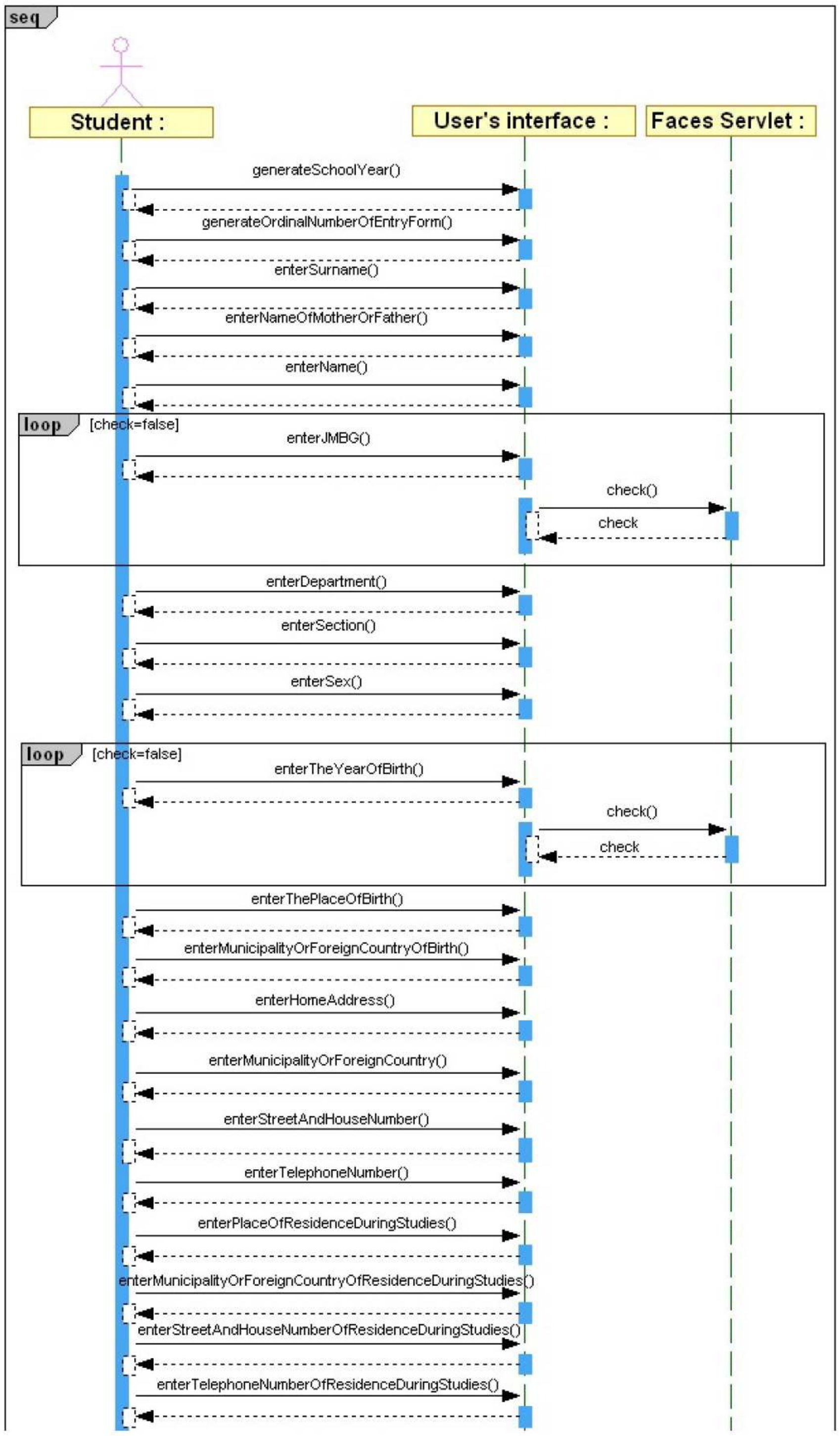

seq
Student :
User's interface :
Faces Servlet :
generateSchoolYear()
generateOrdinalNumberOfEntryForm()
enterSurname()
enterNameOfMotherOrFather()
enterName()
loop [check=false]
enterJMBG()
check()
check
enterDepartment()
enterSection()
enterSex()
loop [check=false]
enterTheYearOfBirth()
check()
check
enterThePlaceOfBirth()
enterMunicipalityOrForeignCountryOfBirth()
enterHomeAddress()
enterMunicipalityOrForeignCountry()
enterStreetAndHouseNumber()
enterTelephoneNumber()
enterPlaceOfResidenceDuringStudies()
enterMunicipalityOrForeignCountryOfResidenceDuringStudies()
enterStreetAndHouseNumberOfResidenceDuringStudies()
enterTelephoneNumberOfResidenceDuringStudies()

Figure 3: Sequence diagram *Initial enrolment*

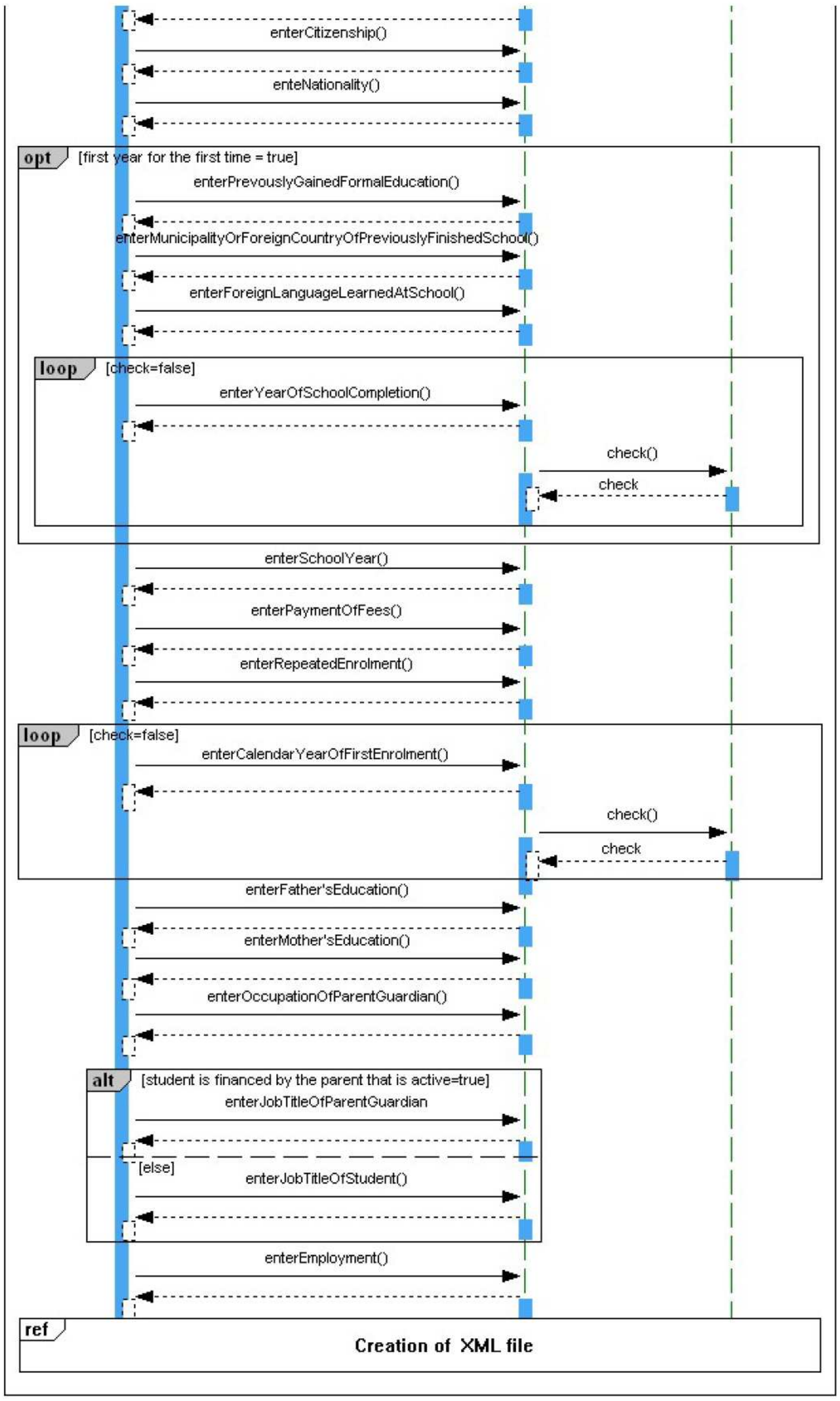

enterCitizenship()
enteNationality()
opt [first year for the first time = true]
enterPrevouslyGainedFormalEducation()
enterMunicipalityOrForeignCountryOfPreviouslyFinishedSchool()
enterForeignLanguageLearnedAtSchool()
loop [check=false]
enterYearOfSchoolCompletion()
check()
check
enterSchoolYear()
enterPaymentOfFees()
enterRepeatedEnrolment()
loop [check=false]
enterCalendarYearOfFirstEnrolment()
check()
check
enterFather'sEducation()
enterMother'sEducation()
enterOccupationOfParentGuardian()
alt [student is financed by the parent that is active=true]
enterJobTitleOfParentGuardian
[else]
enterJobTitleOfStudent()
enterEmployment()
ref
Creation of XML file

Figure 3: Sequence diagram *Initial enrolment* (continuation)

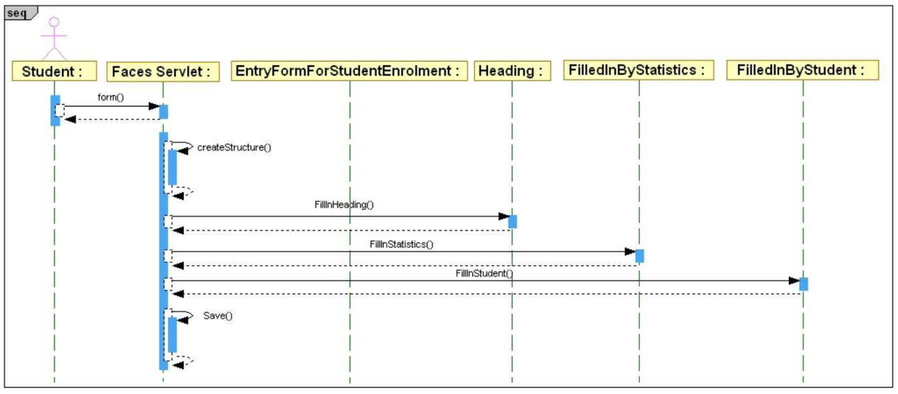


Figure 4: Sequence diagram *Creation of XML file*

Sequence diagram *Semester enrolment*. The messages that are exchanged when executing the sequence diagram *Semester enrolment* script are shown in Figure 5. The messages *generateSchoolYear()* and *generateEntryFormOrdinalNumber()* are identical with the same messages of the sequence diagram *Initial enrolment.* The modification of the data from the previous student enrolment entry form is done by the other messages. The names of these messages are in the form *modifyXXX*, which corresponds to the messages in the form *enterXXX* of the sequence diagram *Initial enrolment.* For example, the message *enterName()* of the sequence diagram *Initial enrolment* corresponds to the message *modifyName()* of the sequence diagram *Semester enrolment.*

3. Application for student enrolment

On the basis of the described modelling, the implementation of the Web application for updating and processing the enrolment form is done. IBM WebSphere Studio Application Developer 5.1.2 [12] environment, JavaServer Faces [13] technology, and Java [14, 15] and JavaScript 1.2 [16] program languages are used for the implementation of Web application for student enrolment.

The first screen form of the student enrolment entry form is presented in Figure 6. School year is automatically generated, while the matriculation book number is transferred from the entry screen form to the system. Register number is identical for all students of the Faculty of Sciences, while the ordinal number of the entry form is automatically generated for each student individually (algorithm: the highest ordinal number +1). Student enters surname, father's or

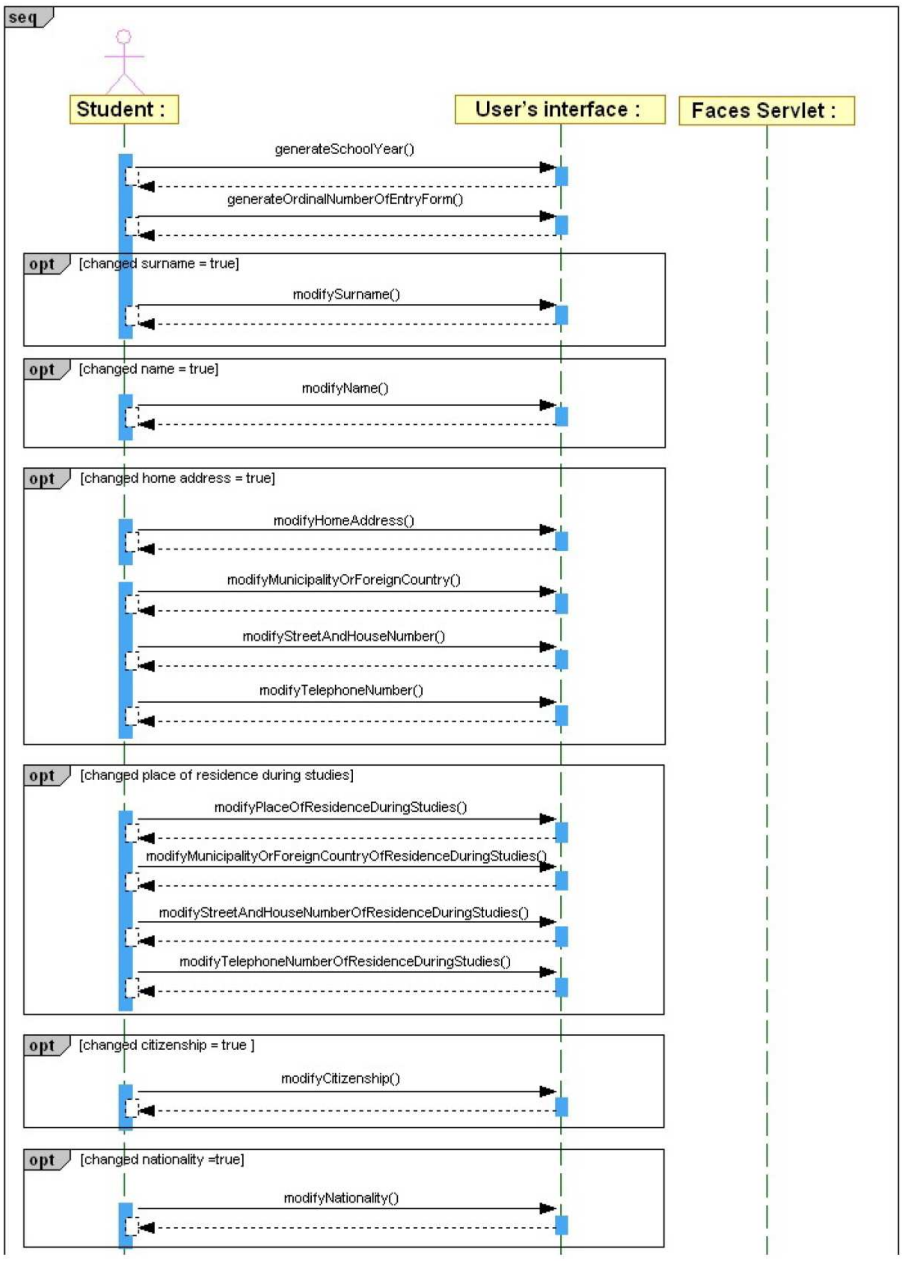

seq
Student :
User's interface :
Faces Servlet :
generateSchoolYear()
generateOrdinalNumberOfEntryForm()
opt
[changed surname = true]
modifySurname()
opt
[changed name = true]
modifyName()
opt
[changed home address = true]
modifyHomeAddress()
modifyMunicipalityOrForeignCountry()
modifyStreetAndHouseNumber()
modifyTelephoneNumber()
opt
[changed place of residence during studies]
modifyPlaceOfResidenceDuringStudies()
modifyMunicipalityOrForeignCountryOfResidenceDuringStudies()
modifyStreetAndHouseNumberOfResidenceDuringStudies()
modifyTelephoneNumberOfResidenceDuringStudies()
opt
[changed citizenship = true]
modifyCitizenship()
opt
[changed nationality =true]
modifyNationality()

Figure 5: Sequence diagram *Semester enrolment*

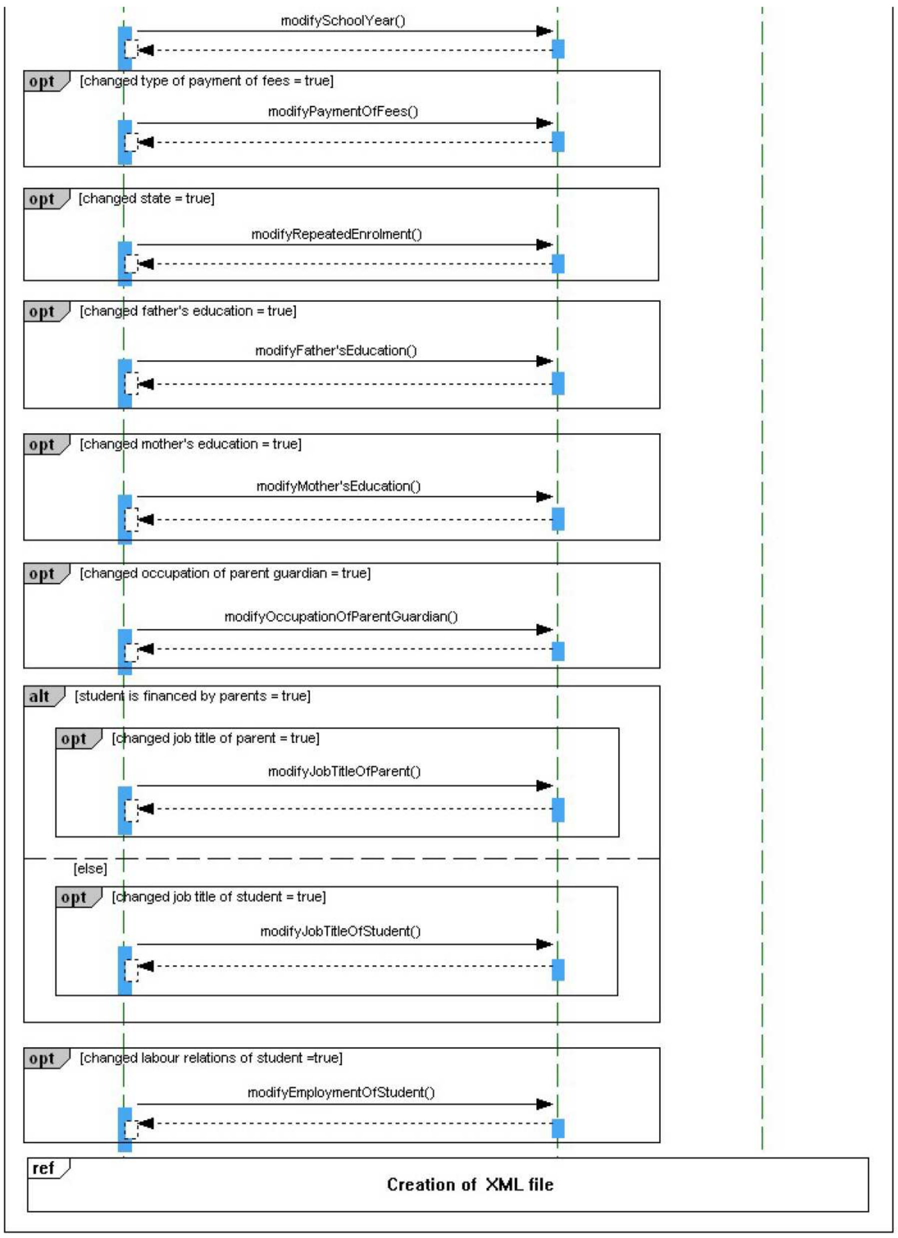
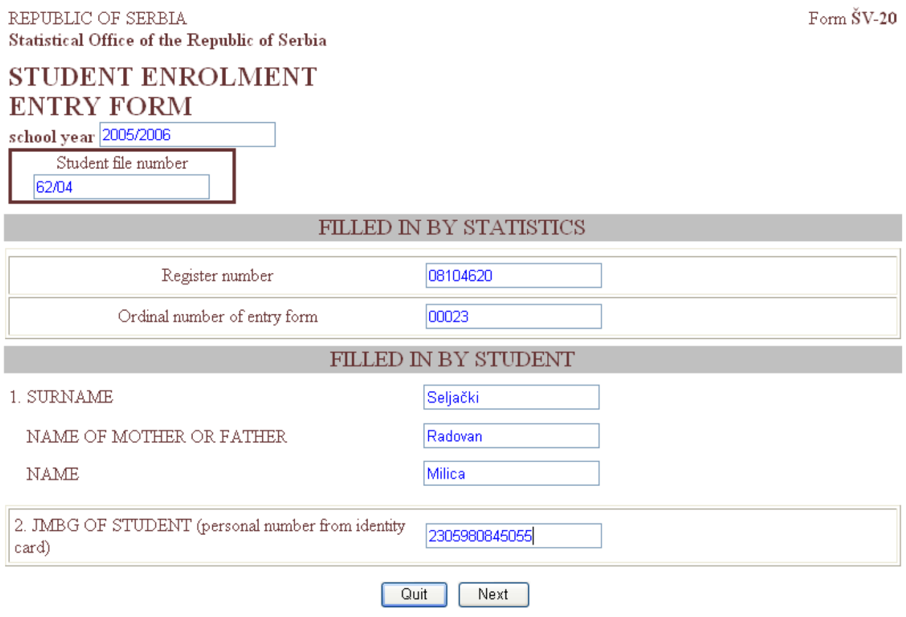

modifySchoolYear()
opt [changed type of payment of fees = true]
modifyPaymentOfFees()
opt [changed state = true]
modifyRepeatedEnrolment()
opt [changed father's education = true]
modifyFather'sEducation()
opt [changed mother's education = true]
modifyMother'sEducation()
opt [changed occupation of parent guardian = true]
modifyOccupationOfParentGuardian()
alt [student is financed by parents = true]
opt [changed job title of parent = true]
modifyJobTitleOfParent()
[else]
opt [changed job title of student = true]
modifyJobTitleOfStudent()
opt [changed labour relations of student =true]
modifyEmploymentOfStudent()
ref
Creation of XML file

Figure 5: Sequence diagram *Semester enrolment* (continuation)

mother's name, name and personal number in one line editors **SURNAME, NAME OF MOTHER OR FATHER, NAME** and **JMBG OF STUDENT (personal number from identity card).** The next screen form of student enrolment entry form is displayed by pressing the button **Next**, while one stops filling the form by pressing the button **Quit.**


REPUBLIC OF SERBIA
Statistical Office of the Republic of Serbia

Form ŠV-20

STUDENT ENROLMENT ENTRY FORM

school year 2005/2006

Student file number 62/04

FILLED IN BY STATISTICS

Register number	08104620
Ordinal number of entry form	00023

FILLED IN BY STUDENT

1. SURNAME	Seljački
NAME OF MOTHER OR FATHER	Radovan
NAME	Milica
2. JMBG OF STUDENT (personal number from identity card)	2305980845055

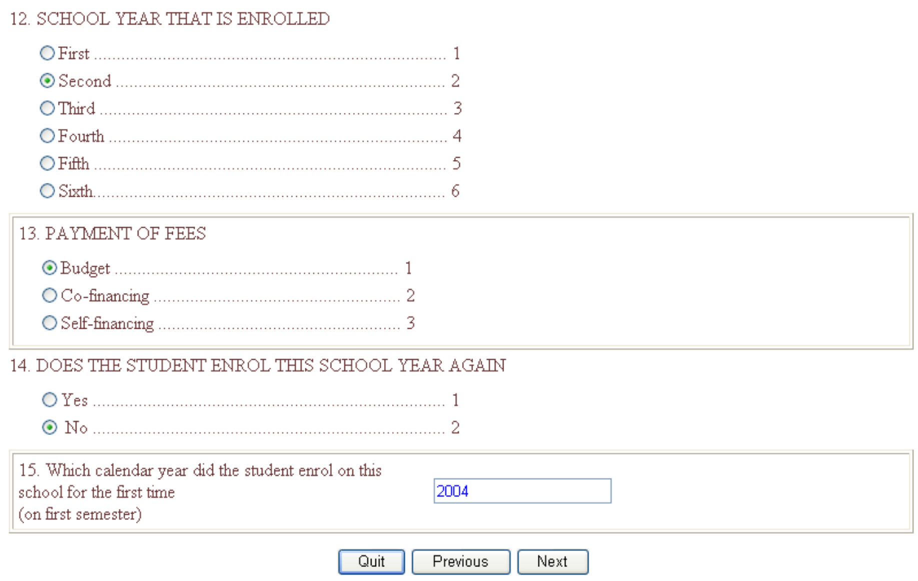
Figure 6: Entering basic student's data

Other screen forms contain the rest of the fields that student should fill in. Figure 7 shows one of these forms. By pressing the button **Save** after answering the last question, the student is enabled to save his/her data previously entered in entry screen forms into the database of student enrolment entry forms.

In the previous text we described in brief the initial student enrolment. On the occasion of every subsequent semester enrolment, screen forms are filled in with the student data entered at the time of the previous semester enrolment, whereby the modification of data is enabled if there was a change of it.

4. Conclusion

A student service information system based on XML documents has been developed at the Faculty of Sciences, University in Novi Sad. One of these documents is the student enrolment form (ŠV-20), which is described in accordance



12. SCHOOL YEAR THAT IS ENROLLED

First 1
[✓ Second .... 2]
Third 3
Fourth 4
Fifth 5
Sixth 6

13. PAYMENT OF FEES

[✓ Budget .... 1]
Co-financing 2
Self-financing 3

14. DOES THE STUDENT ENROL THIS SCHOOL YEAR AGAIN

Yes 1
[✓ No .... 2]

15. Which calendar year did the student enrol on this school for the first time (on first semester): 2004

Figure 7: Input of data about the enrolment of a school year

with XML schema so that all the information contained in the form is included in the schema. On the basis of this schema, it is possible to generate the identical look of the original printed form. This schema has served for implementing the Web application of the student service for work with XML document ŠV-20 subsystem. The application is implemented in Java environment. The architecture of the application is such that the updating of the data is implemented by the object model of XML schema of ŠV-20. This enables placing the data from the different systems into the same database. In addition, Web service can be defined, which can be used by the different software systems of the student service, as well as by other institutions, especially the Statistical Office of the Republic of Serbia.

A longer-term goal of this project is to introduce a public electronic service for updating and processing student's enrolment entry form. It is possible to establish such a service at various levels: university, regional or state. For establishing this service, the electronic form of ŠV20 needs to be prescribed by the law. In that case, all legally prescribed functions of ŠV20 could be carried out electronically.

References

[1] Jaksic, M., Modelling of Bibliographical Standards in XML tehnology, Softvare: Practice and Expirience, 34, 2004, pp. 1051 - 1064.

[2] Budimir, G., Surla, D., Quality control system of XML bibliographic records, Novi Sad Journal of Mathematics, Volume 34, Number 1, 2004, pp. 107-130.

[3] Vidaković, J., Rackovic, M., Modelling of Bibliographical Standards in XML tehnology, Softvare: Practice and Expirience, 36, 2006: pp. 513 - 524.

[4] Nenadic, K, Mapping faculty curriculum intoXML schema, Novi Sad Journal of Mathematics, Volume 34, Number 1, 2004, pp. 157-170.

[5] Sumic, D, Archiving XML Documents for the University Student Service, Master's thesis, Faculty of Science and Mathematics, Novi Sad, 2006.

[6] Cuk, D, Softver system for student's master archive book management , excepted Master's thesis, Faculty of Science and Mathematics, Novi Sad, 2006.

[7] Mirkovic, M., Surla, D., XML schema of entry form for students, Novi Sad Journal of Mathematics, Volume 37, Number 1, 2007, pp. 75-84.

[8] Unified Modeling Language, UML Resource Page, `http://www.omg.org/`, [January 15, 2006]

[9] Unified Modeling Language: Superstructure specification, v. 2.0, 8 October 2004, `http://www.omg.org/cgi-bin/doc?ptc/2004-10-02`, [January 17, 2006]

[10] Describe, Embarcadero, `http://www.embarcadero.com/products/describe/dedatasheet.html`, [January 15, 2006]

[11] The Apache XML Project, Apache XMLBeans, `http://xmlbeans.apache.org/index.html`, [January 30, 2006]

[12] U. Wahli, I. Brown, F. Ferraz, M. Schumacher, H. Sjostrand, *WebSphere Studio Application Developer Version 5 Programming Guide*, 2003, `http://www.redbooks.ibm.com`

[13] U. Wahli, G. Cohen, M. Perrins, L. S. Acera, M. Zandersen, *WebSphere Studio 5.1.2 JavaServer Faces and Service Data Objects,* 2004, `http://www.redbooks.ibm.com`

[14] E. Armstrong, J. Ball, S. Bodoff, D. Bode Carson, I. Evans, D. Green, K. Haase, E. Jendrock, *The J2EE^{TM} 1.4 Tutorial*, Sun Microsystems Inc., 2004, `http://java.sun.com/docs/books/j2eetutorial/index.html`, [February 20, 2006]

[15] Java programming language, `http://www.java.sun.com` [October 11, 2007]

[16] W3Schools - Full Web Building Tutorials, `http://www.w3schools.com` [January 20, 2006]

Received by the editors April 24, 2007