|  |  |
| --- | --- |
| Politehnica University of Bucharest | |
| Software Engineering Quiz-generator Software Design Document | |
| Team:  Teodora Mohai, gr. 1262E  Salvador Canas Moreno , gr. 1262E  Irina Talmaciu, gr. 1262E  David Suarez Caro, gr. 1262E |  |
| Coordinator:  As. Lucian Pestritu  As. Alexandra Todiruta | Date created:  Sunday, December 02, 2012 |

# Delivery Report

(will be delivered along with the project)

Name Group Project implementation [%, reason] Signature

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Delivery date:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Table of Contents

[Delivery Report 2](#_Toc254468674)

[Table of Contents 3](#_Toc254468675)

[System Design 4](#_Toc254468676)

[Document Change History 4](#_Toc254468677)

[1. Introduction 4](#_Toc254468678)

[1.1. Purpose 4](#_Toc254468679)

[1.2. Target Public 4](#_Toc254468680)

[1.3. Definitions, Acronyms and Abbreviations 4](#_Toc254468681)

[1.4. Document Structure 4](#_Toc254468682)

[2. References 5](#_Toc254468683)

[3. Decomposition Description 6](#_Toc254468684)

[3.1. Modules Description 6](#_Toc254468685)

[3.2. Description of Concurrent Processes 6](#_Toc254468686)

[3.3. Description of Data Modules 6](#_Toc254468687)

[4. Dependency Description 8](#_Toc254468688)

[4.1. Dependencies among modules 8](#_Toc254468689)

[4.2. Dependencies among processes 8](#_Toc254468690)

[4.3. Dependencies among data modules 8](#_Toc254468691)

[5. Interface Description 9](#_Toc254468692)

[5.1. Module Interfaces 9](#_Toc254468693)

[5.2. Processes Interfaces 9](#_Toc254468694)

[6. Detailed Design 10](#_Toc254468695)

[6.1. Modules detailed design 10](#_Toc254468696)

[6.2. Data Modules Detailed Design 10](#_Toc254468697)

[Appendices 11](#_Toc254468698)

[A1. Use cases diagrams 11](#_Toc254468699)

[A2. Class diagrams 11](#_Toc254468700)

[A3. Sequence diagrams 11](#_Toc254468701)

[A4. Document evolution 11](#_Toc254468702)

[A5. Conclusions on activity 11](#_Toc254468703)

# System Design

According to the IEEE STD-1016-1998, *IEEE Recommended Practice for Software Design Descriptions*.

## Document Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | The author/authors of the change | Details of changes |
| 1.00 | 28.11.2012 | Teodora Mohai  Salvador Canas Moreno Irina Talmaciu  David Suarez Caro | Basic version |

## 1. Introduction

### 1.1. Purpose

This Software Design Document provides the design details of University Politechnica Bucharest Quiz-generator application. It means this is the document which will guide all the developers in their work. All the technical details have to be written clearly in this document in order to be understood by all the developers.

The expected audience is University Politechnica of Bucharest, including all the teachers who work there, the student developers, the student assistants and people who will maintain the application.

### 1.2. Target Public

|  |  |  |
| --- | --- | --- |
| **Name** | **Last name** | **Role** |
| Teodora | Mohai | Student developer |
| Salvador | Moreno Canas | Student developer |
| Irina | Talmaciu | Student developer |
| David | Suarez Caro | Student developer |
| Lucian | Pestritu | Assistant |
| Alexandra | Todiruta | Assistant |
| Nicolae | Goga | Proffesor |

### 1.3. Definitions, Acronyms and Abbreviations

**Acronyms:**

**XML**- Extensible Markup Language

**SQL**-Structured Query Language

**UML**-Unified Modeling Language

**IEEE**-Institute of Electrical and Electronic Engineers

**SDD**-Software Design Document

**SRS**-Software Requirement Specification

**Tbn**-To be named

**Tbd**-To be decided

**Web Site**-A place on the world wide web

**Definitions:**

**Connection** **String** = string used to connect to a database, containing server name or IP, database name, database user name and password.

**Data** **Source** = data retrieved from the database based on an SQL query, used in displaying structured information.

**Query** = primary mechanism for retrieving information from a database and consist of questions presented to the database in a predefined format. Many database management systems use the SQL standard query format.

**Java** = object-oriented programming language originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform.

**Class =** a construct that is used to create instances of itself – referred to as class instances, class objects, instance objects or simply objects.

**Method** =subroutine (or procedure) associated with a class. Methods define the behavior to be exhibited by instances of the associated class at program run time.

**Entity – Relationship model** (**ER model**) = an abstract way to describe a database. Diagrams created to design these entities and relationships are called entity–relationship diagrams or ER diagrams.

**Pattern** = software solutions that solve a recurring problem within a given context. The use of patterns has many advantages. Patterns encapsulate a design expert's time and expertise to solve a software problem.

### 

### 1.4. Document Structure

The remaining chapters and their contents are listed below.

Section 2 is a reference list which shows all the sources used to develop this document.

All the references are numbered and dated. The authors are also named with the corresponding reference.

This allows a clear explanation where you can obtain more details of some part of the design.

Section 3 is a Decomposition Description that specifies the different modules we described previously in the SRS document and all their characteristics detailed. This chapter also contains the description of the concurrent processes and the data modules with all their specifications.

Section 4 is a Dependency Description where you can find all the dependencies firstly named in the previous chapter and detailed in this chapter.

Section 5 is an Interface Description which presents the external characteristics of the user interface showing all the different actions the users can do.

Section 6 is Detailed Design which shows all the modules previously named and described. This part contains diagrams and examples of the code that developers can follow.

Section 7 is the Appendices where you will find all the diagrams which describe the behavior of the application.

## 2. References

[1] IEEE STD-1016-1998, *IEEE Recommended Practice for Software Design Descriptions*

[2] [IEEE] The applicable IEEE standards are published in “IEEE Standards Collection,”

2001 edition.

[3][XML] All the tutorials to understand the XML language are in <http://www.w3schools.com/xml/>.

[4][HTML]The standars HTML documents are described in <http://www.w3.org/>.

## 3. Decomposition Description

### 3.1. Modules Description

#### 3.1.1. Description of Module 1

|  |  |
| --- | --- |
| Name | Quizzes generate |
| Type | Code module |
| Purpose | To generate randomly quizzes. |
| Way of operating | The user opens the command line and he will be asked to introduce the location of the xml file which contains the questions for the quizzes. If the xml file is not a valid one, an error message will displayed and the user must introduce once more the location.  Next, the operator will have to choose the number of questions he wants from the xml file. If this number will not be a valid one, he gets an error message and will have to enter again a number.  Afterwards the user will have to choose between: introducing a new location for another xml file or go one step forward.  The operator will be asked to enter the number of quizzes and then to write the output directory path where the quizzes which will be generated will be saved. |
| Resources | Teacher, Quiz, XMLSource, OutputHTMLFile |

### 3.2. Description of Concurrent Processes

#### 3.2.1. Description of Process 1

|  |  |
| --- | --- |
| Name | SelectXML |
| Type | Process |
| Purpose | To get the location of the xml file which contains the questions for the quizzes |
| Way of operating | Description of the operating way |
| Subordination | NumberQuestions |
| Dependencies | This process’ behavior depends on the input given by the teacher. |
| Resources | Necessary resources for this process (e.g. global timer (NTP)) |

#### 3.2.2. Description of Process 2

|  |  |
| --- | --- |
| Name | NumberQuestions |
| Type | Process |
| Purpose | To choose the number of questions from the xml file |
| Way of operating | Description of the operating way |
| Subordination | NumberQuizzes |
| Dependencies | SelectXML & and the input given by the teacher in this process. |
| Resources | Necessary resources for this process (e.g. global timer (NTP)) |

#### 3.2.3. Description of Process 3

|  |  |
| --- | --- |
| Name | NumberQuizzes |
| Type | Process |
| Purpose | To choose the number of quizzes |
| Way of operating | Description of the operating way |
| Subordination | ChooseOutputPath |
| Dependencies | NumberQuestions & and the input given by the teacher in this process. |
| Resources | Necessary resources for this process (e.g. global timer (NTP)) |

#### 3.2.4. Description of Process 4

|  |  |
| --- | --- |
| Name | ChooseOutputPath |
| Type | Process |
| Purpose | To choose the output directory path where the quizzes which will be generated will be saved |
| Way of operating | Description of the operating way |
| Subordination | This process do not have any subordination. |
| Dependencies | NumberQuizzes & and the input given by the teacher in this process. |
| Resources | Necessary resources for this process (e.g. global timer (NTP)) |

### 3.3. Description of Data Modules

#### 3.3.1. Description of Data Module 1

|  |  |
| --- | --- |
| Name | Quizzes generate |
| Type | Data module |
| Purpose | The purpose is to introduce the location of the question pool in order to extract questions to generate quizzes for students. The quizzes must be unique for each student – means: the order of the questions and that the multiple-choice questions must have randomize choices every time. |
| Way of operating | Introducing the data from the command line accordingly to what will be required to enter. |
| Resources | Java Virtual Machine, Teacher, Quiz, XMLSource, OutputHTMLFile |

## 4. Dependency Description

### 4.1. Dependencies among modules

This project has only 1 module.

### 4.2. Dependencies among processes

Processes:

* SelectXML
* NumberQuestions
* NumberQuizzes
* ChooseOutputPath

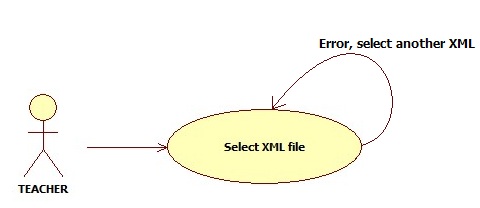


Fig. 1

In fig. 1, there is only 1 process, the first one. If any error occurs, we continue in it, otherwise, we advance to the next one.

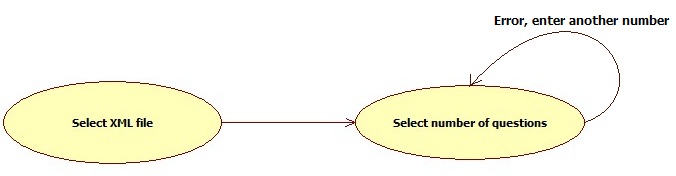


Fig. 2

In fig. 2, there are 2 processes. This figure represents the second step in this software. We come from the XML file selection and now we have to enter the number of questions we want.

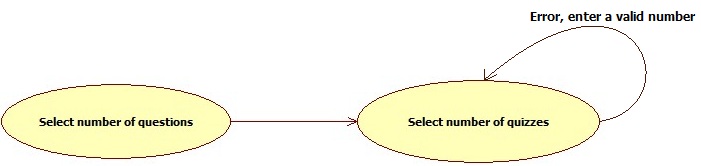


Fig. 3

In fig. 3, there are 2 processes. This figure represents the third step in this software. We come from the number of questions selection and now we have to enter the number of quizzes we want.

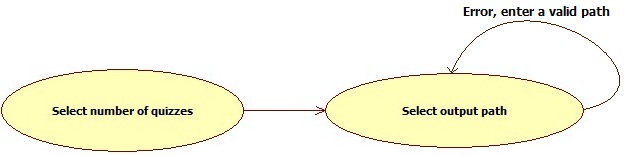


Fig. 4

In fig. 4, there are 2 processes. This figure represents the fourth step in this software. We come from the number of quizzes selection and now we have to enter the output directory path where the generated quizzes will be will be saved.

### 4.3. Dependencies among data modules

This project has only 1 data module.

## 5. Interface Description

This project will not use a graphical user interface, it will use the command line.

### 5.1. Module

|  |  |
| --- | --- |
| Name | Quizzes generate |
| Type | Code module |
| Purpose | The purpose is to introduce the location of the question pool in order to extract questions to generate quizzes for students. The quizzes must be unique for each student – means: the order of the questions and that the multiple-choice questions must have randomize choices every time. |
| Way of operating | The user opens the command line and he will be asked to introduce the location of the xml file which contains the questions for the quizzes. If the xml file is not a valid one, an error message will displayed and the user must introduce once more the location.  Next, the operator will have to choose the number of questions he wants from the xml file. If this number will not be a valid one, he gets an error message and will have to enter again a number.  Afterwards the user will have to choose between: introducing a new location for another xml file or go one step forward.  The operator will be asked to enter the number of quizzes and then to write the output directory path where the quizzes which will be generated will be saved. |
| Description | Introducing all the required data will be made opening the command line and using the keyboard. |

## 6. Detailed Design

### 6.1. Module detailed design

#### 6.1.1. Module

|  |  |
| --- | --- |
| Name | Quizzes generate |
| Type | Code module |
| Purpose | To generate randomly quizzes. |
| Way of operating | Introducing the data from the command line accordingly to what will be required to enter. |
| Classes | Teacher, Quiz, XMLSource, OutputHTMLFile |

**6.1.1.1. Module, class 1**

|  |  |
| --- | --- |
| Name | Teacher |
| Purpose | Introduces the required data. |
| Members | Name, NoOfStudents |
| Methods | SelectXML, ChooseQuizFormat, ChooseOutputPath |

**6.1.1.1.1. Module, class 1, method 1**

|  |  |
| --- | --- |
| Name | SelectXML |
| Purpose | Description of the purpose of this method within the class |
| Description | This method will allow you to write the path of the question pool. |

**6.1.1.1.1. Module, class 1, method 2**

|  |  |
| --- | --- |
| Name | ChooseQuizFormat |
| Description | This method will allow you to choose how many questions from a directory you want to have in the quiz. |

**6.1.1.1.1. Module, class 1, method 3**

|  |  |
| --- | --- |
| Name | ChooseOutputPath |
| Description | This method will allow to write the path of the output file. |

**6.1.1.1. Module, class 2**

|  |  |
| --- | --- |
| Name | Quiz |
| Purpose | Description of the purpose of this class within the module |
| Members | NoOfQuestions, OutputPath, NoOfQuizzes |
| Methods | GenerateOutputFile |

**6.1.1.1.1. Module, class 2 method 2**

|  |  |
| --- | --- |
| Name | GenerateOutputFile |
| Description | This medthod will contain the location where the output file will be generated. |

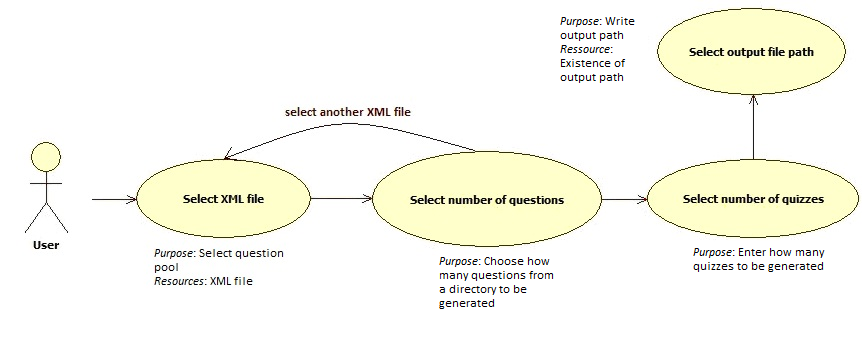
**6.1.1.1. Module, class 3**

|  |  |
| --- | --- |
| Name | XMLSource |
| Purpose | The location from where the questions will be extracted. |
| Members | Name, Path, QuestionPool |

**6.1.1.1. Module, class 4**

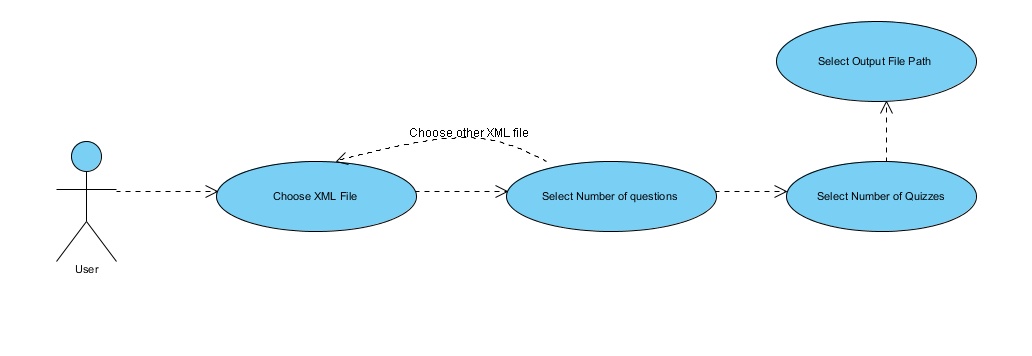
|  |  |
| --- | --- |
| Name | OutputHTMLFile |
| Purpose | Contained of the output file. |
| Members | SerialNumber, FileName, QuestionID, QuestionText, QuestionAnswer |

### 6.2. Data Modules Detailed Design

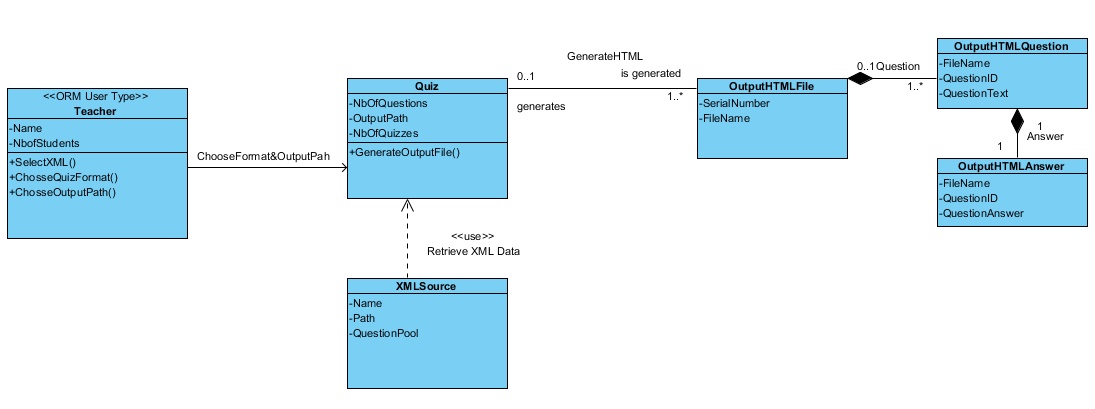
****

# Appendices

## A1. Use cases diagrams

A use case diagram is a graphical representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system.

## A2. Class diagrams

A class diagram in UML should describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

## A3. Sequence diagrams

## SequenceDiagramA sequence diagram in UML is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence.

## A4. Document evolution

Throughout this document, the developpers of the application intend to organize a structured plan of all the software requirements needed in order to develop a functional application. Here are discussed important details regarding :

* all the sources used to develop this document.
* the different modules described previously in the SRS document and all their characteristics in detail.
* the description of the concurrent processes and the data modules with all their specifications.
* a Dependency Description where we can find all the dependencies.
* an Interface Description which presents the external characteristics of the user interface showing all the different actions the users can do.
* a Detailed Design which shows all the modules previously named and described. This part contains diagrams and examples of the code that developers can follow.

This Software Design Document is meant to add several important details to the SRS document offering the design details for the Quiz-generator application. This document is intended to guide its developers in their work.

The SDD document comprises more specific details on how the application should be developped and its main functionalities. All the technical details are written very clearly and syntactically correct, in order to be well understood by all its users.

The expected users are from the Politechnica University of Bucharest, including all the teachers who work there, the student developers, the student assistants and people who will maintain the application.

## A5. Conclusions regarding the activity

The SDD document manages to offer a detailed description of the Quiz-generator application. We can observe that it has an easy to read structure, with additional diagrams and shemas, in order to make the requirements as clear as they can be, for an easy interpretation by its readers e.g. the programmers in the developping team.

This document provides a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective.

We can now conclude that the project and the developping team is ready for starting the next phase in the lifecycle of the application, which is the actual implementation of the theoretical capabilities and start writing the code.