Assessing Students’ Performance in Distance Education Courses

Larissa Vladinova, Milen Petrov, Trayan Iliev

Abstract: ARCADE is a project aimed to develop an integrated software platform for authoring and delivery of Internet-based distance courses covering the university needs. Unified Modeling Language (UML) and Unified Software Development Process were applied as a basis for the project development. The Assessments and Assignments package provides computer supported instructional feedback (tests, projects, essays, problems). The IMS Question and Test Interoperability specification (QTI) defines the basic structures (Assessment, Section, Item – ASI) for the Assessments module development. This module includes three main subsystems – ASI Structures Management, Tests Management, and Test Performance.

Keywords: E-Learning, Computerized testing, Learning Content Management Systems, Object-oriented analysis and design, UML, IMS QTI Specification

ARCADE PROJECT

ARCADE (Architecture for Reusable Courseware Authoring and Delivery) is a project developed by a team of researchers and students at the Department of Information Technologies, Faculty of Mathematics and Informatics of Sofia University “St. Kliment Ohridski”. The goal of the project is to develop an integrated software platform for authoring and delivery of Internet-based distance courses covering the university needs. [1]

The project was developed in two stages. The first stage was focused on providing web-based courses with a common delivery framework incorporating user and system management, course and curriculum management, communication facilities, and student assessment tools. It automated the work of four different user roles: Students, Instructors, Course Administrators (managing the curriculum and courses), and System Administrators (managing the system in a secure and stable state). The second stage was aimed to design and implement an open, platform independent set of tools, automating the process of courseware authoring.

OBJECT-ORIENTED DEVELOPMENT PROCESS

As described by Jacobson, Booch and Rumbaugh in [2] the Unified Software Development Process (USDP) is a “generic process framework that can be specified for a very large class of software systems, for different application areas, different types of organizations, different competence levels, and different project sizes”. The Unified Process is component-based, and uses Unified Modeling Language (UML) for preparing all blueprints of the software system. UML is an industry standardized “graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system” [7]. The Object Management Group (OMG) – an organization that “promotes the theory and practice of object-oriented technology in software development”, adopted it as a standard in 1997.

The USDP is use-case driven, architecture-centric, iterative and incremental [2]. There are five main workflows that take place in each project iteration – requirements specification, analysis, design, implementation, and testing. The main purpose of the requirements specification workflow is to develop a use-case model of the software system that describes the actors – “coherent sets of roles that users of an entity can play”, and the use-cases – “coherent units of functionality provided by a system” [7]. During the analysis workflow the functional and nonfunctional requirements are transformed in basic system architecture. The design workflow links the abstract analysis architecture with concrete implementation technologies – it provides a design model, a detailed blueprint for the system implementation. The construction and coding of different modules is the main purpose of the implementation workflow. During the testing workflow all implemented classes, subsystems, and the whole software system are tested according to a test plan.
Test cases are derived from existing use-cases. They are applied according to predefined test procedures, and the whole process can be automated using test components.

The USDP seems the most appropriate candidate for ARCADE software platform development (compared with other candidates like Extreme Programming - XP, and Dynamic Systems Development Method – DSDM) [1, 3] because it supports production of high quality, UML-based documentation inevitable for an open project with many participating developers. It is providing an opportunity for parallel development of different modules by concurrent teams of researchers and students, because of the well-specified workflow models. Though it is not ideal – implementing it in a full scope requires a lot of resources. That’s why – as proposed by the authors of the methodology – the process should be tailored to the individual project requirements.

**ARCADE ARCHITECTURE**

As presented in Fig. 2 there are four main packages developed during the first stage of the project:

- **Course and Curriculum Management** – provides different functions for the instructors and the course administrators, such as managing courses and course instances (courses tailored to particular student group), assigning students to student groups and creating curriculum. This package automates several student functions also – searching for courses and reading course materials.

- **User and System Management** – this package contains most of the system administrators functionality including user and user group management, assigning user permissions for different objects in the system, managing system event logs and so on.

- **Communications** – five different communication alternatives are provided: internal e-mail, virtual chat rooms, discussion boards, news, and shared/personal space (web-based virtual disks).

- **Assessments and Assignments** – the package is responsible for management of students’ performance evaluation. It supports the construction, modification, and management of tests by instructors, performing the tests by students, evaluating results according to built-in instructions, generating feedback to instructors and
students. It provides means for automated assignment and evaluation of individual and group course projects, and helps the instructor to form final grades based on different students’ assessments.

PRELIMINARY RESEARCH

One of the main design principles in ARCADE project is the compliance to existing learning technology standards and specifications. Related to this principle a research of existing best practices and specifications for computer-based testing was conducted. The following achievements are considered as most appropriate for our purposes: ARIADNE TestIT Questionnaire Tool Set [5], AICC CMI Guidelines for Interoperability [4], and IMS Question and Test Interoperability [6].

ARIADNE and ARIADNE II are two big projects connected to the “Telematics for Education and Training” sector under the 4th framework program of European Community. The ARIADNE system is based on the "core" tools, which allow indexing, storage, and distribution of different teaching documents. Various authoring tools are also proposed to help the teaching engineers in the creation of these documents. The TestIT Tool Set is meant for authoring and presenting questionnaires, analyzing and scoring student answers, providing feedback to learners, recording answers and producing result reports in a variety of formats (including HTML and spreadsheets). The tool is available for online and offline use and requires Java enabled operating system, Web browser and an Internet connection in case of online use. [5]

The AICC CMI Guideline for Interoperability outlines and defines a number of Computer Managed Instruction (CMI) principles and terms. This document provides a description of CMI data model. It consists of five components, and one of them is Testing. This component specifies the types of tests (mastery, performance checklists, and attitude questionnaires), strategies for scoring the test items, type of test assignment (as pre-test or post-test). The testing component can support data collection for item analysis. If tests are administered via traditional CBT lessons, there are mechanisms to dump item results into the CMI system for item analysis. Tests are administered either on-line (from the CBT system) or off-line (e.g. paper or performance). CMI systems support both norm-referenced and criterion-referenced tests. [4]

IMS Global Learning Consortium includes members from educational, commercial and government organizations. It is developing and promoting open specifications for facilitating online distributed learning activities such as locating and using educational content, tracking learner progress, reporting learner performance, and exchanging students’ records between administrative systems. Each specification created by IMS consists of package of documents – Information Model, XML Binding Specification and Best Practices and Implementation Guide. The IMS Question and Test Interoperability specification (IMS QTI) defines how to tag the questions and tests. It offers a core set of presentation and response structures, and variety of methods for collecting and scoring the results of questions. To represent these options the QTI Specification defines Assessment, Section, Item (ASI) structures. Items contain all the necessary data elements required to compose, render, score and provide feedback from questions. Thus there is a difference between Item and Question: Item contains Questions, layout rendering information, response processing information, and hints, solutions and feedback related to this item. Sections are composed of Items or other Sections, which are logically related. Assessments are assembled from Sections. The term “test” is defined as an instance of Assessment. In the Information model of specification Response-type taxonomy of items is described in details. [6]

After studying the aforementioned tools, guidelines and specifications the development team decided to base the description of test objects on IMS QTI
specification. It is detailed, platform independent and allows defining proprietary structures if necessary.

ASSIGNMENTS AND ASSESSMENTS PACKAGE

The Assignments and Assessments package includes development of two separate modules (see Fig. 1) – Assignments and Assessments. The former module automates the management of individual and group students’ projects, essays, and problems (assignment, checking current status, evaluation). The second one provides tools for web-based testing of learners’ performance. Describing the Assessments module is the main goal of this paper.

The main user roles (actors) supported by the Assessments module are Authors of ASI structures, Instructors and Students.

- Authors of ASI structures – subject matter experts, instructional designers, pedagogical specialists. They create the basic elements – Items, Sections, Assessments, but may not be involved in specific process of evaluation related to particular course.
- Instructors are the persons responsible for setting up the assessments, creating tests for specific groups of learners and taking account of the results.
- Students are the group of users that has to perform the test and receive information about the results.

The analysis of requirements led us to the following list of three main groups of functions that the testing system has to provide:

- ASI Structures Management – it is related to searching and retrieving existing elements (Assessments, Sections, Items) from Resource bank, modifying elements according to user’s rights, creating new elements and adding them to a Resource bank. Figure 2 represents the data structure of the assessment as defined in IMS QTI.

Four common types of Items are implemented in the Assessments module:

- **Multiple choice** – the examinee is offered several possible answers and only one is correct. Examples for this type are yes/no questions.
- **Multiple response** – the difference between this type and the previous one is that here it is possible to have more than one correct answer.
- **Fill in blank** – requires filling in the correct answer in specific place represented by Item.
- **Essay** (also called open ended question) – requires entering free answer of the defined question

Fig. 2. Assessment data structure

Currently the Assessment module supports a stand-alone import utility for the first Item type (multiple choice) without supporting nested sections and multiple sections in one assessment. The other three types are under development.

- **Tests Management** – includes creating instances of the assessments – tests and setting their properties (date and group of students to be assigned)
- **Test Performance** – allows performing the tests by students, evaluating results according to built-in set of rules, receiving feedback, and generating reports. The implemented evaluation algorithms allow setting different weights for different Items or equal weights for all Items.

The requirements are described in more than 30 use cases in terms of UML. Figure 3 presents a use case diagram for the Instructor activities.
During the analysis workflow a basic architecture supporting the functions defined in the use-case model was initially specified. The functionality was split into three main subsystems. The resource bank management subsystem realizes the use-cases that manage ASI structures (assessments, sections and items), their properties, structures and content. The second subsystem is concerned with test presentation and performance representing a group of use cases related to Student's activities when viewing and performing tests. The third subsystem is dealing with administration of tests – by implementing a group of use cases related to Instructor activities (assign test to students, manage test results, and export the results to other modules/systems). It provides connection to the Assignments module for exporting the tests’ marks in order to use them (combined with marks from other assignments) to form the final grade of the student.

The design workflow extends the abstract analysis architecture into concrete set of class hierarchies consistent with the chosen implementation technologies (Java™, JSP™). Three-layer Model-View-Controller (MVC) class architecture was used in order to separate the presentation, business logic and database persistence aspects of the system. The Model layer describes the data structures that represent the relations between the ASI structures. ARCADE is supporting a broad range of Relational Database Management Systems (RDBMS) by using the Java Database Connectivity (JDBC)™ technology. The View layer is responsible for visual presentation of the data. In ARCADE this functionality is implemented in a separate package – Visual Components. This package encapsulates html tags generation and makes the view layer (presentation) independent from data structures and business logic. The third layer in the Assessment module is the Controller layer. It is responsible for model data manipulation, connecting the View and Model layers and synchronizing their work. The controller classes in Assessments module handle events generated by the user and implement the business logic of the specific use cases defined in the use-case model.

In the scope of the implementation workflow a set of software components (Java classes, JSPs, CSS and SQL scripts for creating database tables) were developed based on the specified design class hierarchies. Integration tests were made with following configuration: MySQL™ and Oracle 8i™ as RDBMS; JSP™ container - Apache Tomcat Server, version 3.3.3, and Java 2 Platform, Standard Edition, version 1.3.1 as server-side software; Internet Explorer™ version 5 or higher as client-side software.

In order to facilitate the integration and testing processes the Assessments module functionality was implemented in several small steps – builds. The first build for the module
was focused on implementing and integrating the presentation subsystem. In the scope of this build were implemented the main Student functions including the abilities to: list assigned tests, view information about the tests, perform the planned tests, view runtime information about the performance process (e.g. planned and estimated time), pause and resume tests, automatically evaluate tests and receive feedback. The second build is related to implementation of authoring import/export utilities for different kinds of ASI structures. It includes also several common integration use cases for test generation, test evaluation process monitoring, custom scoring and advanced auto-evaluation rules. The third build will complete the basic functionality for the Assessment module. This build is focused on ASI structure creation and management that is independent from import/export facilities implemented in the second build. The coverage of ASI according to IMS QTI specification will be extended (mainly with different types of test questions).

The first build of the Assessments module is currently finished, and the second is under implementation.

The last workflow according to the development methodology accepted for the project is the testing workflow. Integration and usability tests were performed for the first build. The system was tested with two groups of 5-10 testers. As a result from these tests several user interface adjustments were made and some additional features were added (e.g. possibility to pause and resume tests during performance needed in case of Internet connection failure, malfunctions of hardware, etc.)

CONCLUSIONS

Some directions for future extension of the Assessments module include: creating ASI Resource Bank and supporting a wide range of import/export formats for the items, adding support for additional item types, result evaluation algorithms, item sequencing algorithms (computer adaptive testing).

The assessments authoring system may be improved by integrating it with the tools for courseware authoring under development in the second stage of the project.

REFERENCES


ABOUT THE AUTHORS

Larissa Vladinova, Milen Petrov, Trayan Iliev
Department of Information Technologies, Faculty of Mathematics and Informatics, Sofia University, 5 James Bourchier Str., 1164 Sofia, Bulgaria,
Tel. / fax: +3592 656 157, E-mails: {lary, milenp, t_iliev}@fmi.uni-sofia.bg