Specification of the Verity Learning Companion and Self-Assessment Tool

Sergiu Dascalu* Daniela Saru Ryan Simpson* Justin Bradley* Eva Sarwar* Joohoon Oh***

* Department of Computer Science
University of Nevada, Reno
1664 N. Virginia St.
Reno, NV, USA

** Dept. of Control and Industrial Informatics
Faculty of Automatic Control & Computers
University Politehnica of Bucharest
Sp. Independentei 313, Bucharest, Romania

Abstract

In this paper, the specification of Verity, a web-based instructional tool, is presented. Verity is intended to be used as a learning assistant and self-assessment tool, more than a web-based testing system. This represents a new approach to web-based instruction. Specifically, the purpose of the Verity software tool is twofold: first, to assist the students in their study by providing supporting information, sample examples, exercises and problems related to a given course and, second, to allow the students self-evaluate their knowledge of course material through sets of multiple-choice questionnaires. Both assistance from the instructor to the student and feedback from the students to the instructor are supported by a web-based implementation solution. Verity's main components are described in the paper and the tool's functionality is outlined. A discussion of possible extensions is also included.

Keywords: computer-aided education, software tool, requirements specification, use cases, scenarios, web application, UML.

1 Introduction

Verity[1], whose name is intended to suggest an accurate, objective evaluation of the students' level of knowledge and study progress, is a web-based learning companion and self-assessment tool designed to assist students in their study by providing course material for study as well as tests related to course topics.

In summary, the Verity software package consists of three main components: the instructor's module, the student's module, and an information repository for course-related material.

The information repository is the linking component of the package, a component that stores study material provided by the instructor, results of students' evaluations, as well as questions from the students and answers given by the instructor to questions.

The instructor's module allows the introduction of course-related study material in the system, material which can be used by students in their learning process and self-evaluation.

The student's module supports the following: retrieval of key concepts, principles and methodological instructions, retrieval of hints and sample solutions for exercises and problems, structured communication with the instructor, generation of questionnaires for self-evaluation, and calculation and archival of student test results. More details on Verity components and functionality are provided later in the paper.

Three observations are worth noting regarding this project. First, although computerized tests based on multiple-choice questionnaires already exist (e.g., GRE [1], TOEFL [2], various tests for professional certification [3, 4]) our proposed package is novel in that it allows for assistance and guidance from the instructor via a two-way communication mechanism. Verity's intended destination as learning assistant and self-assessment tool – as opposed to just generating

1) This project has been supported in part by an Instructional Enhancement Grant from the University of Nevada, Reno.

the questionnaires and evaluating the answers – also represents a new way of approaching the problem. Second, although initially we intend to use the Verity tool within the frame of one of the Computer Science capstone courses offered at the University of Nevada, Reno (UNR), the range of courses which can benefit from this tool is practically endless. Due to its flexibility and generality, the Verity package can be particularized for any given course. Third, the proposed package's possibilities for extension are numerous. For instance, within a larger context and time-span the Verity software can evolve into a Professional Study Companion and Self-Evaluator with high marketability potential. Other possible extensions of Verity are discussed later in the paper.

From a development point of view, the Verity software has been built following a simplified (due to time constraints) version of the Unified Process (UP) [5] and its model has been represented using constructs of the Unified Modeling Notation (UML) [6, 7]. Guidelines regarding both the develop-ment process and the modeling notation have been taken from [8].

The remaining of this paper is structured as follows: Section 2 presents Verity's most important functional and non-functional requirements, Section 3 provides details of use case modeling by showing the Verity's use case diagram and sample use cases, Section 4 highlights several aspects of Verity's high-level architecture, Section 5 reports on the current status of the tool and shows excerpts from the tool's user interface, Section 6 points to a series of future extensions, and Section 7 concludes the paper by summarizing our work and by presenting our acknowledgements.

2 Requirements Specification

Before starting the specification of Verity software, we have defined a series of functional and non-functional requirements that need be satisfied by the initial operational version of the tool. In the following, using the practical, efficient style proposed in [8], the most important requirements for Verity, both functional and non-functional, are presented.

2.1 Functional Requirements

The following is a list of functional requirements for the Verity learning companion and self-assessment tool:

R01 Verity shall keep a record of instructor and student accounts.

R02 Verity shall allow instructors to add, modify, and delete questions in the database.

R03 Verity shall support various types of memos, including but not limited to: video clips, audio clips, presentations, problem sets, hint sets, solution sets, and links to web sites with supplemental study material.

R04 Verity shall allow instructors to add, modify, and delete memos in the database.

R05 Verity shall support multiple-choice questions with a single correct answer.

R06 Verity shall support at least three levels of difficulty for questions. Each level of difficult shall be reflected in a number of specific points associated to a question of that difficulty.

R07 Verity shall generate tests based on student's specification of the desired number of questions in the test or of the desired total points of the test.

R08 Verity shall determine the allowable time for taking a test based on the difficulty of questions included in the test.

R09 Verity shall provide correct answers to questions.

R10 Verity shall calculate the scores of the student tests.

R11 Verity shall record the scores of the student tests.

R12 Verity shall provide a non-timed test mode in which memos are available, and a timed test mode in which memos are not available.

R13 Verity shall allow instructors to see individual student weighted test score averages.

R15 Verity shall allow students to see their weighted test score averages.

R16 Verity shall allow both instructors and students to see weighted test score averages for the class.

It is useful to note that in the above requirements the term memo has been use in a general way. In essence, a *memo* encapsulates a piece or set of pieces of study-supporting material presented in various formats (specifically, text, audio, video or combinations of these) and having various structures, for example PowerPoint presentation, web-based tutorial, web article, and so forth.

2.2 Non-Functional Requirements

Some of the most relevant non-functional requirements for Verity are listed below:

T01 Verity shall be a web-based application.

T02 Verity shall be written in HTML, JavaScript, Flash, MySQL and PHP4.

T03 Verity shall support at least 100 student accounts.

T04 Verity shall support at least one instructor account.

3 Use Case Modeling

Early in the modeling process, Verity's functionality has been defined using use cases and scenarios. The entire functionality of this instructional tool is represented in the use case diagram shown in Figure 1. A correspondence between the functional requirements listed in subsection 2.1 and the use cases shown in the use case diagram can be easily established.

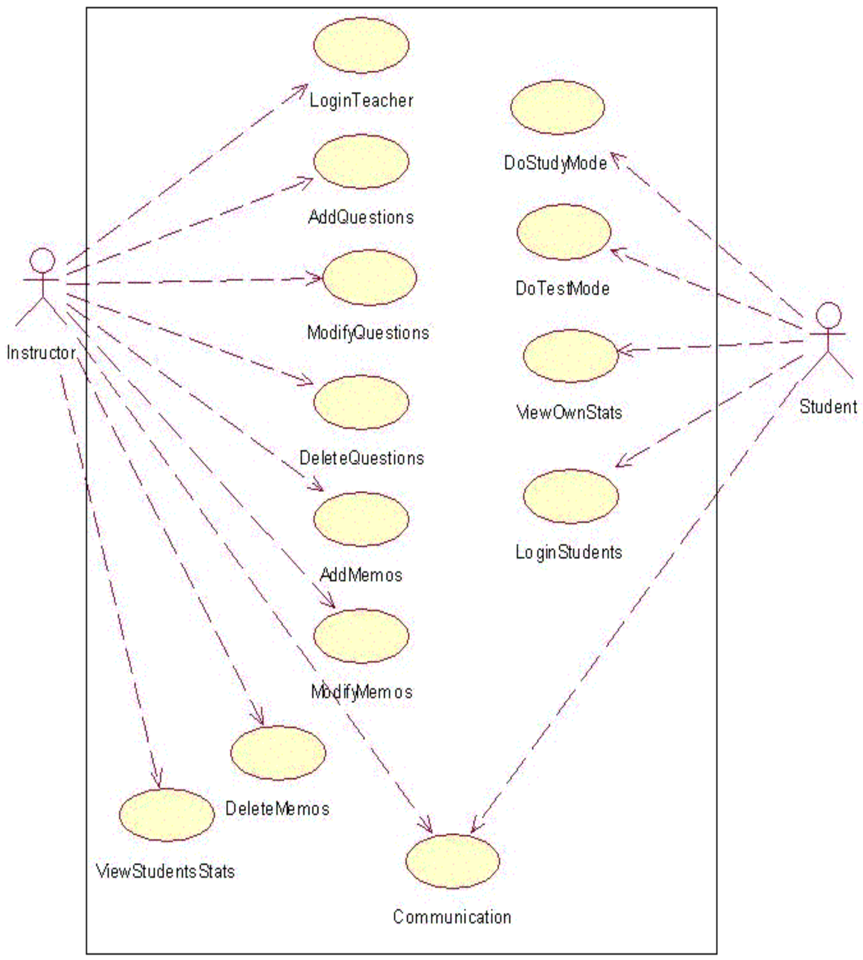


Fig. 1 Verity's Use Case Diagram

For illustration purposes two of Verity use cases are shown in a simplified form in Figs. 2 and 3.

Use case: AddQuestion

Use Case ID: UC02

Actors: Instructor

Preconditions:

1. The Instructor is logged on the Verity instructor's website.
2. The instructor has clicked on the link "Add Question" from the main page of the instructor's website or has clicked on the button "Reset Form" of the Add Question page of the instructor's website.

Flow of events:

1. The use case starts when the above preconditions are met.
2. The system displays the Add Question page with an entry form with fields for the text of the question and the texts of four possible answers. In addition, four radio buttons are displayed for the specification of the correct answer and a drop-down menu is available for the specification of the question's level of difficulty.
3. The instructor fills out the texts of question and of the four possible answers, selects the radio button corresponding to the correct answer, and selects the question's level of difficulty using the drop-down menu.
4. The instructor either clicks on the "Submit Form" button, or clicks on the "Reset Form" button, or closes the Add Question page.
   - 4.1 If the instructor clicks the "Submit Form" button the question is saved in the database.
   - 4.2 If the instructor clicks the "Reset Form" button all the text fields are emptied and the correct answer as well as the question's level of difficulty become unspecified. The overall effect of this action is to return to step 2 of the present use case.
   - 4.3 If the instructor closes the Add Question page then any information entered or selected is not saved and the browser returns to the main page of Verity's instructor's website.

Fig. 2 The AddQuestion Use Case

Use case: DoTestMode [SetupTest only]

ID: UC10

Actors: Student

Preconditions:

1. The student has logged on the Verity student's website.
2. The student has clicked on the "Test" link or has canceled the taking of a test.

Flow of Events:

1. The use case starts when the above preconditions are met.
2. The system displays the Setup Test page with an entry form with fields for the student to select a type of test, based on either the total number of questions or the total number of points desired.
3. The student enters either the total number of questions or the total number of points desired for the test.
4. The student either clicks the "Start Test" button, or clicks the "Go Back" button, or closes the Setup Test page.
   - 4.1 If the student clicks on the "Start Test" button then the system verifies the entry provided.
     - 4.1.1 If the entry is valid then the system displays the Test page and the taking of the test can start.
     - 4.1.2 If the entry is invalid the system displays a message indicating why the student's entry is invalid. The student must then click the "OK" button and return to step 2 of the current use case.
   - 4.2 If the student clicks the "Go Back" button or closes the Setup Test page the system returns to the main page of Verity's student web-page.

Fig. 3 The SetupTest Segment of the DoTestMode Use Case

Due to space limitations only a very limited description of the use cases developed for Verity has been provided in this section. For the same reason, scenarios, which are instances of use cases, have not been presented either. However, for specifying Verity's behavior we have relied significantly on both use cases and scenarios.

4 Architectural Details

Because Verity is a web-based application we have included in its specification several modeling elements that depart somewhat from the traditional, UML-supported representation of object-oriented systems. In particular, we have found useful the inclusion of a site map that details the pages used in the Verity environment (Fig. 4) and have provided descriptions for each of these pages (program units). The description of one of these pages is shown in Fig. 5. In addition, given the data-intensive nature of this application, the definition of all the tables used in Verity to store information was necessary.

In our view, all these supplementary components of the model fall under the classification of specification, given that we use the term *specification* in Alan Davis' sense, that of "a document containing a description" [9]. In Verity's case, the specification of the tool encompasses descriptions pertaining to both requirements analysis and software design.

5 Current Status

Although the focus of this paper has been on the specification of Verity's model, we note that this instructional tool has reached recently its implementation and testing stages, and a first operational –albeit not fully complete– version is currently available. There are still several components of Verity's software that need to be finalized, in particular the generation and the taking of tests, yet as proof of concept Verity has largely achieved its initial goals. We estimate about three more months of implementation, testing, and data entry until a fully functional version will be completed.

Snapshots from the current version of Verity "in action" are provided in Fig. 6 (the student's interface) and Fig. 7 (the template for adding questions). The first snapshot provides an overview of Verity's functionality (student side) and thus complements the use case diagram shown in Fig. 1, while the second snapshot supplies a visual illustration for the use case described in Fig. 2. For more information on the Verity project the reader is referred to [10].

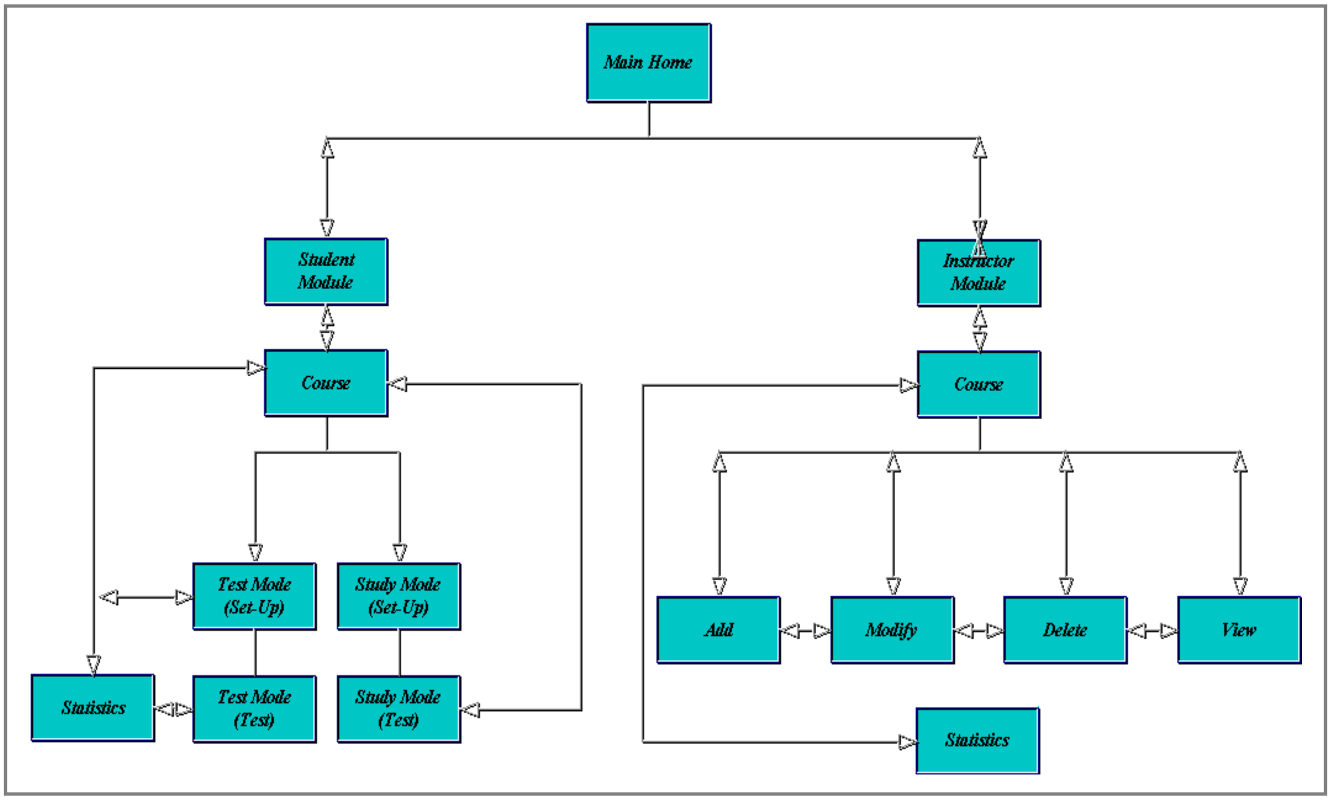


Fig. 4 Verity Site Map

Module	Instructor
Unit	Statistics
Type	MYSQL for stored database on students progress
Description	The instructor has to be logged on the instructor's website, and needs to click on the ViewStudentStats link. The system displays the page that asks which student the instructor would like to view statistics for. Alternatively, the instructor can select to view statistics for the entire class. The system calculates the selected type of statistics and displays them graphically. If applicable, the statistics also indicate how a particular student performs as compared with the class.

Fig. 5 Description of the Statistics Unit

6 Possible Extensions

As stated in the introduction of this paper, we believe that the Verity tool has great potential in terms of applicability and possible extensions.

In what regards short term design and implementation refinements, the Verity tool can be expanded with numerous features and options. In particular, we are looking at enhancing the study and test modes of operation in principal by expanding the types of memos (study material) and by providing means for a more elaborate definition of tests, for example via controlled selection of questions from various sections of the course. Also, there are many aspects that we intend to address regarding the definition and presentation of statistics on tests taken and on the usage of the Verity tool.

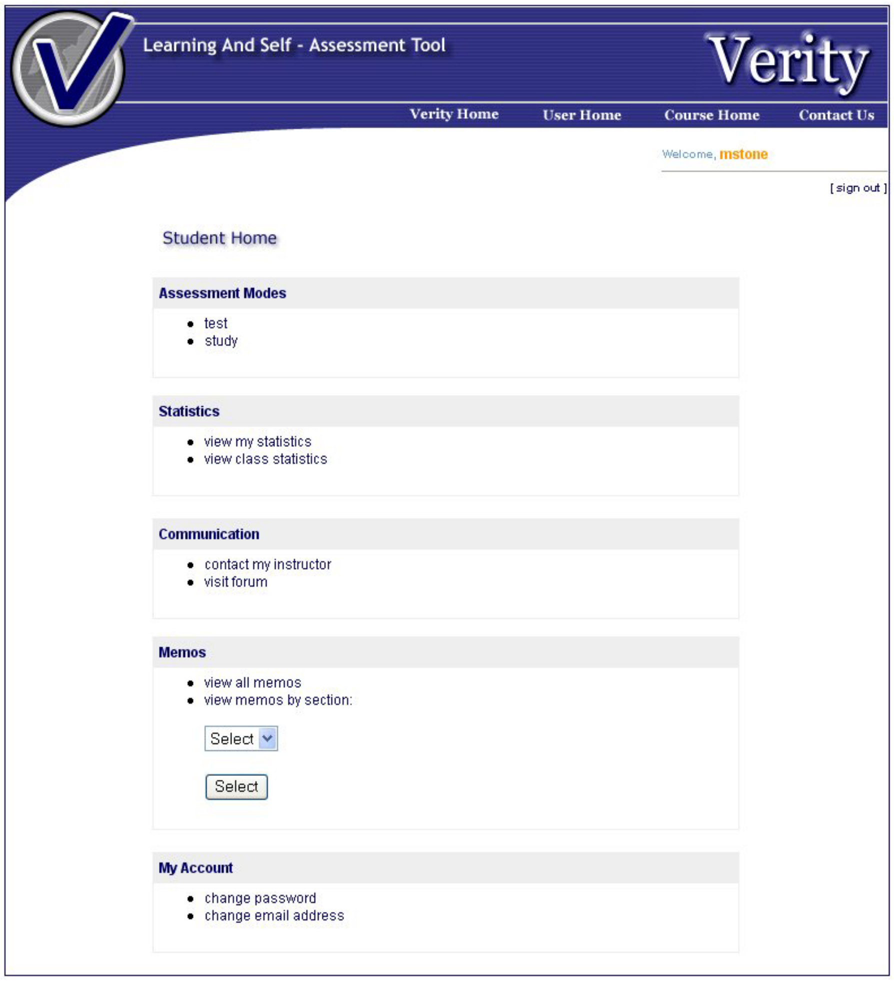


Fig. 6 Verity: The Student's Main Page

These are directions of investigation from which the monitoring of the students' study progress as well the assessment of the instructor's contribution to off-class support of the students' learning could significantly benefit.

On a larger scale, the Verity tool offers the basis for several more complex extensions, which demand intensive research and development. Such extensions encompass the inclusion of an automated, intelligent learning assistant based on an expert-system engine, the development of a multi-user version of Verity for collaborative study and assessment, and the addition of a recommender system for web-wide retrieval and classification of course-related material.

7 Conclusions

In this paper the Verity tool has been introduced via descriptions of requirements, use case modeling, and high-level design. The main goal of this instructional software package, that of providing learning and self-assessment support –as opposed to just offering a new alternative for taking computerized tests– has been emphasized and both Verity's current status and possible extensions have been pointed out. Our belief is that significant practical benefits can be obtained through further enhancement of this tool dedicated to assisting students in their learning process.

Acknowledgement: We would like to thank the University of Nevada, Reno for the Instructional Enhancement Grant that has supported the development of Verity.

References

[1] Graduate Records Examination website, http://www.gre.org/ttindex.html, accessed May 4, 2003

[2] free-TOEFL.com website, http://www.free-toefl.com/, accessed May 4, 2003.

[3] Sun Certification: Java Technology website http://suned.sun.com/US/certification/java/java_progj2se.html, accessed May 4, 2003.

[4] Microsoft Training and Certification website http://www.microsoft.com/traincert/default.asp, accessed May 4, 2003.

[5] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, 1999.

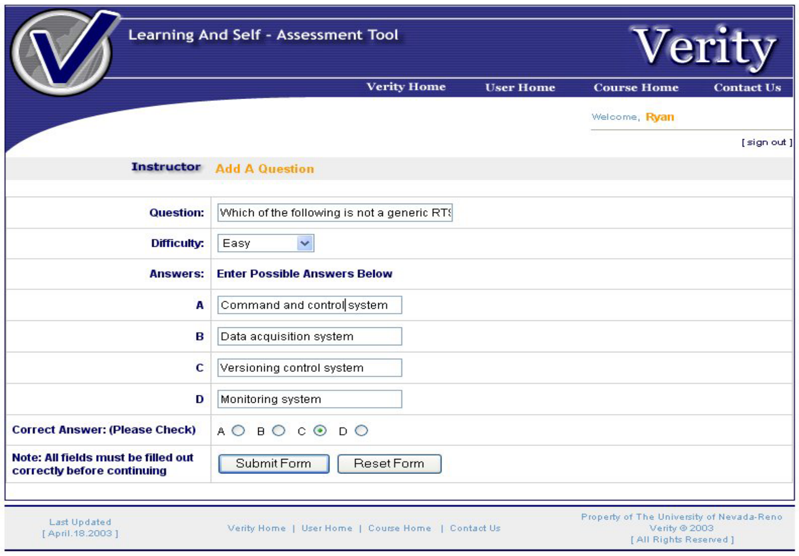
[6] OMG's UML Resource Page, accessed April 25, 2003, http://www.omg.org/uml

[7] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1998.

[8] J. Arlow and I. Neustadt, *UML and the Unified Process: Practical Object-Oriented Analysis & Design*, Addison-Wesley, 2002.

[9] A. Davis, *Software Requirements: Objects, Functions & States*, Prentice Hall, 1993.

[10]Team 9, Verity project, accessed May 2003, www.cs.unr. edu/~rsimpson/home.html



Question:	Which of the following is not a generic RTS
Difficulty:	Easy
Answers:	Enter Possible Answers Below
A	Command and control system
B	Data acquisition system
C	Versioning control system
D	Monitoring system
Correct Answer: (Please Check)	A B [✓ C] D
Note: All fields must be filled out correctly before continuing	Submit Form Reset Form

Last Updated
[April.18.2003]
Verity Home | User Home | Course Home | Contact Us
Property of The University of Nevada-Reno
Verity © 2003
[All Rights Reserved]

Fig. 7 Verity: Instructor's Add Question Page