Online Marking System for Vula

I-feng Yang
Department of Computer Science
University of Cape Town
iyang@cs.uct.ac.za

Choene Rammutla
Department of Computer Science
University of Cape Town
crammutl@cs.uct.ac.za

Tembela Godongwana
Department of Computer Science
University of Cape Town
tgodongw@cs.uct.ac.za

ABSTRACT
The University of Cape Town is in the process to migrate its course contents from a disparate number of learning management systems to the new Vula system. As the management of course contents is often done using different systems through various interfaces, Vula will be providing a common solution to all e-learning needs at UCT.

However, Vula does not currently provide a tool that allows the creation of marking guides and the subsequent use of such a marking guide for inputting marks and comments as feedbacks to students.

This report describes the process of creating such a tool - the 'Vula Marking System', as it aims to solve the above problem, using J2EE and JavaScript Technologies, and essentially acts as a single entry point for all input of marks into Vula.

Keywords
Software Engineering, user interface, marking guide, user evaluation, Vula, Web services, Java Script, JBOSS, SOAP

1. INTRODUCTION
Vula Marking System is a Web-based interface developed to help instructors and tutors mark and grade student assignments online. The application is developed for the Center for Higher Education of the University of Cape Town. It is aimed to improve the marking of assignments and feedback for all departments.

Users of the system are allowed to access courses they are members of. Under each course the user can view all the assignments and students registered in the course. Instructors have rights to create a marking guide for a particular assignment, while tutors may only view the marking guide that they use to grade student assignments.

In this paper we start by briefly describing the background of the technologies and processes used to develop the system. Then we describe the software development methodology that we used, explain the three components of the project and finally conclude the document.

2. BACKGROUND
2.1 Developing Tools for Vula
The complexities and intricacies of Vula, or any learning management system makes it 'tricky' for developers to develop modular components that could be plugged in and out of the system. As Vula conforms to the Sakai framework, the best approach to develop and provide tool support for Vula is to conform to the Sakai architecture and develop components with specifications agreed upon by the consortium.

Therefore, to develop a modular and usable system within the time scope given, a separate web-based system, which communicates with Vula via Web Services while running on its own server, will be developed. This approach has many advantages as it has minimum impact on the existing Vula system while also allows the technologies used for the implementation of the tool to be independent of those used by Vula.

2.2 Usability
The outcome of the project is a system that interacts with users e.g. lecturers and tutors. The system interface therefore needs to be usable and hence serve the purpose that it was created for. Thus during the development of the system usability guidelines and practices were followed and used to ensure that the interface is usable.

3. DEVELOPMENT METHODOLOGY
3.1 Software Engineering Methodology
As a software engineering project that centers on usability, incorporating design methodologies during the process of software development is important. Parts of Extreme Programming (XP) will be adopted as the main software engineering methodology, as they provide a set of principles that facilitate the production of high quality software in minimum time.

Participatory design techniques will also be applied to facilitate the design of user interface and the gathering of user requirements.

Note that some practices of XP will be less applicable; techniques such as 'pair programming' would only be applied during the integration of the different components, and '40-hour work week' is clearly not suitable for the nature of this project. The key emphasis will be, however, on the identification of user stories (use cases), following the simple design strategy and iteratively developing small releases or prototypes with added features.

3.2 Division of System into components
To make the process of development easier, to thus allow implementation to be carried out simultaneously by the development team, the system is divided into three components each with a distinct purpose. The 'EJB & Web Services' component is the backend of the system that provides Web Services functionalities for accessing the Vula database; the

'JavaScript' component focuses on allowing the user to interactively creating and using a marking guide; while the 'User Interface' component makes use of both components and glues them together and provides other general usability features such as user log in, browsing of courses and assignments, etc.

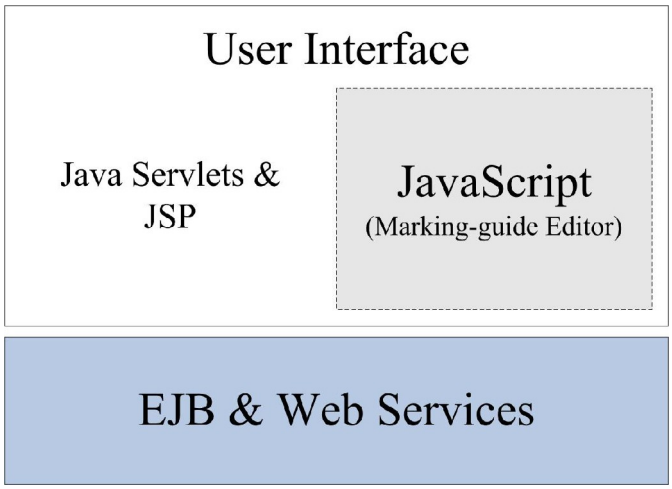

Figure 1. Components of Vula Marking System

The design and implementation of the system will be carried out on the identified components, each following the XP methodology, before they are finally ready to integrate. The next three sections will describe the process of developing each of the components.

4. USER INTERFACE

4.1 Design

The methodology used to design the Vula Marking tool is User-Centered Design. A participative design form of this methodology was applied in the design and usability of the interface.

4.1.1 Low fidelity prototypes/ paper prototyping

We held meetings with tutors in order to develop a pencil and paper design of the interface. Tutors were driving the development of the interface as they suggested ways of better developing the interface. The low fidelity prototype was subjected to scrutiny by the developer and tutors during a series of follow-up meetings.

4.1.2 Card Sorting

Card sorting is a technique for exploring how people group items, so that you can develop structures that maximize the probability of users being able to find items [3]. See figure 2 for an illustration of Card sorting that was done to complement and confirm the paper design developed by the tutors and developers. Five users were given cards and asked to group cards that belong together. We approached users individually and explained the exercise. These users were randomly selected on campus and the results of sorting cards were compared to the flow of the paper prototype. The adjustments were made depending on the differences that arose between the prototype and the card sorting mechanism.

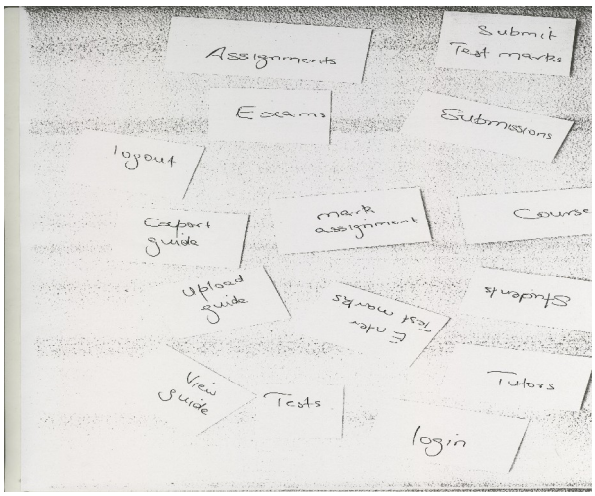

Figure 2. Vula System card sorting

4.2 Implementation

4.2.1 JSP and Servlet Component

We now turn into the interaction between the JSP and the Servlet component of the system. Figure 3 below shows some key JSP components i.e. login, courses, assignments, submissions and marking guide. The marking guide component is further divided into sub parts and is discussed in Marking-guide Editor section. All the JSP pages interact directly with the Servlet component in order to get information to be displayed on the browser.

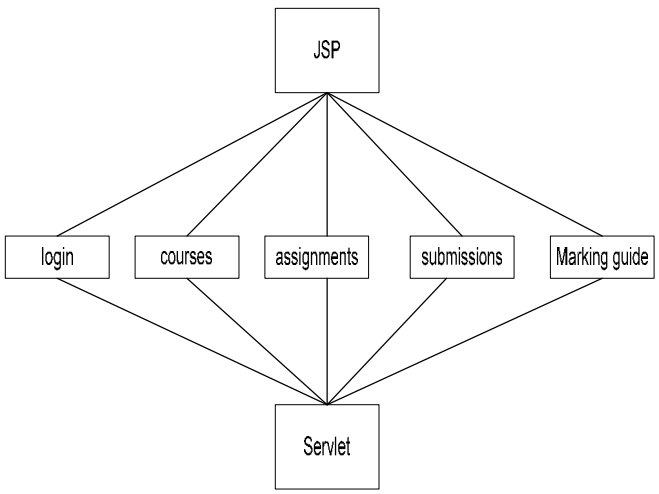

Figure 3. Interaction between JSP and Servlet components

The Servlet component is also currently divided into objects corresponding to the JSP component. The Servlet component gets information by using java beans from Vula via web services and the information is then dispatched to the Java Server Pages.

4.3 Evaluation

4.3.1 Scenario Testing

Scenario testing is a form of usability testing whereby participants use the Web site or prototype to do tasks. During testing users are given tasks or goals to do either in writing or verbally. Twenty users were involved in testing the interface prototype of the system as a way scenario of testing. These users were split into two equal groups. One group was asked to perform this scenario: "Create a marking guide for CSC115F". Scenario question for group two: "Mark one student's submitted assignment for CSC115F". At this point of testing, the prototype was incredibly simple and users understood the flow of the system quite easily and therefore was usable.

5. MARKING-DUIDE EDITOR

5.1 Design

5.1.1 Initial Requirement Gathering

As the aim of the project is to provide a tool for Vula, it is essential that the developers understand what feature(s) is needed as far as Vula is concerned. As a user centered software project, it is also imperative that the team has a good understanding of the user requirements in the beginning of each XP iteration before implementation.

Initial Meetings with Vula

The first two meetings with the Vula team established a basic project specification. It was understood that the current Vula system (built on the Sakai framework) lacks a feature that facilitates the creation and use of a marking guide.

Study of Existing Marking Guides

An example of a currently-used marking guide was obtained from a tutor and the structure of which was identified. This is important as the goal of the component is to create such a marking guide.

Interview with Teaching Assistants

An interview session was set up with two teaching assistants. Teaching assistants are responsible for creating the marking guides for assignments and are therefore the potential users of the marking-guide editor. The current process of creating marking guides and using marking guides were demonstrated and understood.

User Requirements

The initial requirement study set the scope for the project as well as each component of the system. Here are the key points identified that are relevant to the marking-guide editor:

- The process of marking-guide creation and inputting marks should be visual and facilitated through effective user interactions.
- A marking guide is essentially a tree with each node containing question text and mark allocation.
- Teaching assistants should be allowed to edit a marking guide after submission; and tutors should be allowed to make changes to marks and comments after submission.
- Automated summing of marks, handling of bonus marks and penalty deductions are 'bonus' features that could facilitate teaching assistants and tutors' job.
- The system will produce a HTML formatted marking-guide for viewing by the students.

User Stories

As Extreme Programming (XP) requires the use of user stories, the user stories for the first iteration of XP were drawn up after the initial requirement analysis. Additional features, or user stories, were identified during the user feedback sessions in the beginning of each of the subsequent iterations. Three iterations were used during the development of the marking-guide editor.

'Tree' as Structure for Marking Guides

It was identified during the requirement analysis that a marking guide has the structure of a tree with each node of the tree holding 'some' values. Therefore the 'marking-guide editor' is essentially a tree creator. Figure 4 below illustrates the correlation between a tree and a marking guide.

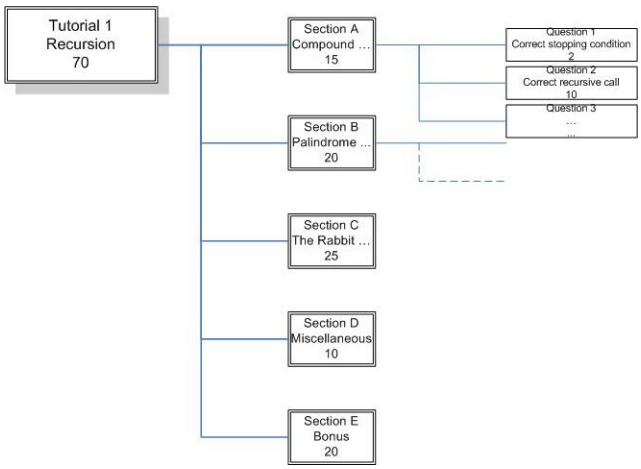

Figure 4

5.2 Implementation

5.2.1 Building an Interactive Environment on Web

There are three popular technologies that allow the creation of, web-based, dynamic and interactive user experience. JavaScript, Macromedia Flash and Java Applets are all possible choices that provide powerful tools for the creation of the marking guide editor.

However, in addition to being an open source technology like Java, JavaScript is the only technology that does not require any additional browser plug-ins. JavaScript runs directly in a browser, supports object-oriented programming and integrates easily with HTML forms are its advantages and reasons why it was chosen as the technology for creating a web interactive tool.

5.2.2 Creating a Marking Guide Editor

The following values were identified as the values necessary for each node: question number, question text, mark allocation and mark obtained. In addition, a comment field is added to allow the input of comments into each node (section or question) by the tutors during marking; and to uniquely identify a node, a node-id field is added where the value is uniquely assigned depending on the level of the node and its position under its parent node. Lastly, since each question has a section which it belongs to (except for the root node), each parent node, including the root node, must hold a list of nodes (questions) which falls under it.

The Very First Prototype

Following the XP methodology, the very first 'self-contained' prototype that allows the addition, deletion and automatic id-assigning of nodes was developed. Figure 5 demonstrates the two basic operations the prototype supports: addition and deletion (with reassigning of node-ids) of nodes in particular.

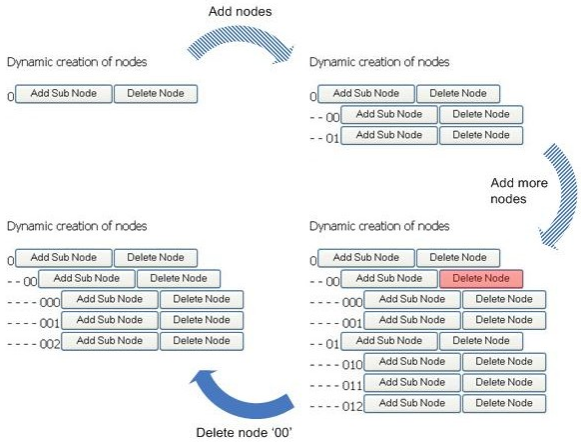

Figure 5

XML Formatted Marking Guide

XML was a chosen as the language for storing the marking guides after they are created. The popularity of XML, the ease of defining formatted documents and the community support for it made it a trivial choice. However, XML formatted marking guides will not be visible to the users as they are merely used for storage. In other words, users will be interacting with a web interface to manipulate the underlying tree. The tree is then stored as an XML document for later retrieval.

Implementing a Complete Editor Using a Library: XML for <SCRIPT> [1]

Although the prototype was created without the help of any particular library, the possibility of using a JavaScript library to manipulate an XML document was explored and XML for <SCRIPT> was found to be a viable option. As the library is W3C Dom Level 2 Compliant (essentially supports generation 5 and above browsers) [2], we are assured that the compatibility between browsers would not be an issue with all modern browsers.

Making use of the library to implement a marking-guide editor makes things simpler as the library essentially provides all functions necessary for the creation of an XML document.

Creating a Mark Editor

Creating a mark editor follows the same process as the marking-guide editor. The two major differences between the marking-guide editor and mark editor are:

1. Mark editor will not manipulate the structure of tree; but instead allow the input of values (marks and comments) into the tree.
2. The output of the mark editor will be in HTML format to allow viewing.

5.3 Testing & User Evaluation

5.3.1 Testing

There are two types of testing that XP uses [4]:

1. Unit Tests are automated tests written by developers to test functionality as they are written. Each *unit* tests only a single class or a small cluster of classes.
2. Acceptance Tests (or Functional Tests) are specified to test that the overall system is functioning as specified. This is either automated or follows a scripted user process.

Both types of tests were utilized, each with a specific purpose, during the development of the marking-guide component. Unit tests were used to test all the 'utility' classes which the main code relied on; while acceptance tests were conducted to test the overall features of the system.

Automated Unit Tests

JsUnit [5] was used as the tool for unit testing during the development of the marking guide component. JsUnit is essentially a tool that unit tests in-browser JavaScript. Since the utility objects, such as the history object (a stack-like object that provides the undo / redo feature) and other string filters and methods are completely self-contained, it was simple to write tests for those classes.

Scripted Acceptance Test

5.3.2 User Evaluation

Although the functional requirements of the component may have been satisfied; much of the success of the component will depend on its usability. Therefore, the outcome of user evaluations on a prototype (or the final system) is important as it could be a measure of success or achievement.

Subjects

The user evaluation of the system was conducted by 10 randomly selected students. The subjects were not limited to the users of the current marking scheme (including paper-based) and Computer Science students, as essentially all students are potential tutors and teaching assistants.

Evaluation Process

A brief introduction about the purpose of the project with key emphasis on the marking guide component was given to the student prior to his or her testing. Then the user would follow the instructions to carry out tasks given; and would then require to answer a questionnaire and rate his or her experience on a scale of 1 to 5 based on the questions. Comments and possible improvements on the features were also requested.

Result of Evaluation

The usability scores obtained for the 'marking-guide editor' and 'mark editor' were divided into current users (teaching assistants and tutors) and potential users and results were analyzed.

- **Transfer of Knowledge and Experience**
  Based on the outcomes of the small sample, the current users found the marking-guide editor simple to use, giving an average score of 4.8 to the component; compare to the 3.4 the potential user gave. This may perhaps be an indication that the knowledge and experience of a user gained from the current system has been carried over, resulting in an easy transition to the new system.

- **Simplicity of Usage**
  Although the potential users did not 'strongly agree' and tend to stay neutral on whether the interface was easy to understand; and that the behavior of the marking-guide editor was of what they had anticipated, they gave a highly favorable score of 3.88 when asked whether they agreed that the editor was simple to use.

  As 6 of the 8 inexperienced users gave 4 or 5 for the simplicity of usage, we may conclude that although the interface of the system was not the easiest to understand and has room to improve, the system was after all simple to use.

6. EJB & Web Services

6.1 Design and Implementation

Secure communication is always an issue amongst the web developers. Web services technologies present with it self sophisticated mechanisms to invoke client /server like enterprise applications.

The communication module was implemented using JBOSS 4.04 with Netbeans 5.0 as the IDE. Both these tools provide intense support for EJB 3.0 which is and SOAP architecture which are useful in client server distributed computing. Extreme programming methodology was highly followed in producing a highly effective communication module.

6.2 Testing

Stress / Load tests, amongst other testing strategies for web services we done using a dynamic test simulation tool. Since it was not possible to employ 1000 people for this project under normal conditions, Siege tool was used for simulations.

7. Conclusions

We will now draw the following conclusions based on the project experience as a whole, and with a greater perspective.

7.1 Extreme Programming (XP) promotes user participatory – essential for developing successful user-centered software

Although due to time constraint, there have only been two formal XP iterations during the development of the marking guide component, the user evaluation performed on the last prototype could essentially be seen as the beginning of the last iteration before the release of the final system.

It is worth noting that repeatedly involve user into the design process has been effective, as during each feedback session, users were able to give meaningful responses on the features implemented, as well as identify new features that could potentially facilitate their tasks. This is clearly visible during the feedback sessions on the two prototypes as new features were identified that could otherwise be missed by developers.

Therefore, it could be said that XP or part of XP that promotes user participation makes it a good choice of methodology for user centered software development where requirements can not be clearly laid out at the commencement of the project.

7.2 Unit testing is important for building complex systems

Although unit tests were only written on a small scale for the purpose of this project, it was not hard to see its advantages in a more complex environment. In cases where the implementation had to change, or codes had to be refactored as to eliminate duplicate codes, unit tests become really handy. As it ensures the flow of logic or coding does not *break* when changes are made, some significant debugging time would be greatly saved.

7.3 JavaScript is a powerful language for browser-based applications

Although there are some intrinsic weaknesses to programming in JavaScript, its inefficiencies, and debugging issues (mentioned in 4.4.1) etc, are some of those; developers should not overlook the capabilities of JavaScript; and the tool support it has.

During the development of the marking guide component, a JavaScript library was used to help with the manipulation of an XML document; and JsUnit was also utilized for unit testing in JavaScript. As the final product of the component demonstrates the dynamic environment JavaScript can create on web, the emerging popularity of Ajax (asynchronous JavaScript technology and XML) confirms the usefulness of combining JavaScript and XML in a web environment.

As compatibility between browsers become less and less of an issues, we will certainly see more JavaScript on the Web.

8. FUTURE DEVELOPMENT

There are essentially two directions where the marking system as a whole could be heading. One, the likely short term possibility, is that it will remain as a standalone system (running on its own server while communicating with Vula via Web Services). Here are some possible features to add in this, short term, scenario:

- Sending email 'reminders' to tutors when the deadline of marking is approaching and there are still unmarked assignments.
- Add other marking tools to the system. Such an example would be a marking tool that facilitates the marking of an essay by allowing the marker to highlight the text within an essay and attach color tags with comments. The color tags highlighting the text will then be visible to on retrieval of the essay. To view the attached comments, the user could simply click on the tags.

Another possible direction of the marking system, which is likely to be the long term solution, is to use this current system as a temporary solution or a prototype; and develop a similar marking system under the Sakai framework (or alternately, replace the current marking system when such a tool is available from the Sakai community).

This has the advantage of eliminating the need for exposing unnecessary Web Services (reducing likely security issues) and the running of a separate system on a separate server (which could both be costly and hard to maintain). More importantly, developing a tool under the Sakai framework will allow it be shared among the members of the Sakai community.

9. ACKNOWLEDGMENTS

Our thanks to Donald Cook our project supervisor for his guidance though-out the project. We also extend our thanks to tutors for their involvement in the application development. Finally our thanks to Vula for the opportunity to develop the marking system.

10. FERENCES

[1] XML for <SCRIPT> http://xmljs.sourceforge.net Last accessed: 2006-10-25

[2] Documentation – FAO *XML for <SCRIPT>*, http://xmljs.sourceforge.net/website/documentation-faq.html Last accessed: 2006-11-10

[3] Information & design http://www.infodesign.com.au/ftp/CardSort.pdf Last accessed: 2006-10-12

[4] John Brewer, & Jera Design. Extreme Programming FAQ. http://www.jera.com/techinfo/xpfaq.html Last Accessed: 2006-10-25

[5] JsUnit http://www.jsunit.net Last Accessed: 2006-11-10