Identifying, Analysing and Testing of Software Requirements in Learning Management System

Sengupta Souvik¹, Dasgupta Ranjan²

BIT, kolkata, India¹, NITTTR, Kolkata, India²
mesouvik@hotmail.com¹, ranjandasgupta@ieee.org²

Abstract

The requirement analysis of e-Learning software is a challenging job considering the diversity of its users, standards and models followed in education. The overall software engineering process behind e-Learning software depends heavily on the correct requirement identification from different perspectives. In this paper we have tried to figure out the all possible user requirements in an e-learning system and then related those requirements to the corresponding stake holders. Then we have tried to group these requirements based on the qualities of a software product namely, reliability, usability, portability etc. We have used different UML diagrams to model these requirements and their mapping.

Keywords: Requirement Analysis, Software quality, UML modelling, e-Learning

1. Introduction:
Understanding the requirements is the most critical process to the success of interactive systems² such as e-Learning software. It is even more important to classify these requirements from different perspective for better understanding of their importance on design. Typically, an e-Learning software provides an author with a way to create and deliver content, an instructor (tutor) to monitor student participation, learning and assess student performance. It may also provide students with the ability to use interactive features such as threaded discussions, video conferencing, and discussion forums. We have tried to identify themajor requirementsfrom various possible perspective.

2. Requirements in e-Learning:
From study on different available e-Learning software we classify requirements into five main components¹ as mentioned below:

   i) curriculum design ii) content management iii) learning activity iv) communication and v) organizational (administrative) perspective.

We further group the requirements as per the classification made.

Curriculum Design:

R1. The contents should be properly sequenced
R2. All the learning contents should maintain metadata
R3. The contents should be maintained in repository
R4. The contents should be reusable
R5. Prerequisites for a course should be well defined
R6. Duration should be set for a course work
R7. There should be formative evaluation technique
R8. There should be offline summative evaluation technique
R9. Policies should be set for individual course
Content management:
R10. Content should be verified before publishing
R11. The learning content should have full multimedia support
R12. The contents should be properly linked
R13. The content should comply to globally accepted e-Learning standard
R14. The contents should be fully or partly importable and exportable
R15. The contents should be updatable
R16. Content should be properly packaged

Learning activity:
R17. Learner should be able to personalize the system according to their preference
R18. The contents should be fully or partly easily downloadable by the learners
R19. Learner should go through an enrolment process so that she can be identified
R20. There should be feedback system for the learners
R21. The contents should be searchable

Communication:
R22. There should be adequate support of hardware to host, run and view the e-Learning software
R23. Certain browser properties should be set to work all the functionalities properly
R24. Chatting provision should be there to obtain synchronous interaction
R25. Blogging/Forum should be there to obtain asynchronous interaction
R26. There should be provision for live sessions [video conferencing]
R27. There should be an emailing system/bulletin board to display the event information and announcements.

Organization:
R28. The contents should be easily uploadable by the authors
R29. Each course should have question bank which should be able throw random questions at runtime.
R30. Session management is required
R31. Records of individual students learning time on a particular chapter should be maintained
R32. Records of individual students learning time on a course should be maintained
R33. Records of student-teacher interaction should be maintained
R34. Records of students performance evaluation should be maintained
R35. Feedback records should be maintained
R36. There should be different levels of evaluation
R37. There should be student-student and student-tutor interaction interface
R38. Result analysis reports should be generated
R39. Feedback analysis reports should be generated
R40. Trend analysis reports should be generated
R41. There should be a periodic review mechanism
R42. The overall architecture should follow a globally accepted standard

3. Mapping the requirements with the stakeholders:
For an e-learning system, there exist different stakeholders and every stakeholder has some business with the system. Thus a clear picture should be maintained among the stakeholders and the requirements. This will not only identify the stakeholders’ role in the system, but at the same time will clarify the requirements and this in turn will help in designing the UML diagrams. We declare the stakeholders in next section and in sec 5 we map the requirements with the stakeholders in the Use Case diagram.
The stakeholders in an e-Learning system:

A. Learner: Learners are students who are the end users of the system. They are the direct beneficiary of the system. They learn and use the content, personalize and participate in course, get assessed and evaluated by the system.

B. Author: Authors are responsible for developing learning content and uploading it to the software.

C. Tutor: Tutors monitor the learning process and help students via chat, blog or mail. There may be live sessions of video conferencing by the tutor. The evaluation process is controlled by tutors.

D. Coordinator: A coordinator creates a course, selects and verifies contents, sets policy, duration and control student enrolment in it. He may assign tutor for learning sessions.

E. Administrator: Administrator is the overall in-charge of the entire system. He looks after the interest and role of all the other users. He accepts feedback and analyse different records for the improvement of the system.

UML diagrams to map requirements with stakeholders: In our approach we have used use-case modelling to represent the relationships between these requirements and the stakeholders.
The above UML diagrams show that though there are five different stake holders, not all have contribute in the requirements of all the components. The requirements in curriculum design have nothing to relate with the learner as the learner does not take part in this process. Similarly the actors in the content management use case are limited to the administrator, author and coordinator. The requirements of the learning activity are set by learner, coordinator and administrator. All except the author have a say in the requirements of communication. However from the organizational perspective all of the stake holders have their own requirements.

6. Mapping the requirements with software quality attributes
Identification of necessary software quality is of immense importance in the design process. At the beginning of the design process all the necessary software quality should be identified and appropriate measures should be taken so that those quality factors are properly embedded in the software. In order to identify the influence of these requirements in the quality of the e-Learning software we tried to analyse these from the perspective of ten different quality attributes.

I) Correctness: The content shown should always come from a repository of verified and trusted learning content and reused in different courses as it increases correctness of the software. The search engine should properly find the contents on a valid input query. The evaluation policy is always a measure for the correctness of the e-Learning software. The standard policy of a course and the two fold evaluation technique helps to evaluate the learner correctly. Questions should be set dynamically and randomly form a question bank that covers the entire course. There should be a feedback system from which recommendation to improve correctness should come.

II) Reliability: The learning contents should be properly sequenced and linked, so that the learner can navigate smoothly through the learning path. This ensures reliability. The support of minimum software and hardware requirements should be satisfied. The scope of modification of the system based on the feedback from the users makes it more reliable.
III) Robustness: The system should be able to behave reasonably in an unexpected circumstance like failure at client or server end. The session management is crucial because at any point of time the system should be able to identify the learner’s state and maintain a log so that it can again start over from the same point. As the learning content should come from the repository it can be possible to restore a learning session instead of restarting from the beginning.

IV) Portability: The e-Learning software should be portable from one system to another. An XML based meta data system is very much helpful in this regard. The learning content should come in a package in order to ease portability.

V) Usability: The learning system should be easy to use by the users. A proper sequence of topics with correct linking is the prime objective. The learner should be able to search the right content. The learner should be able to download and the author should be able to upload easily. The learner should have provision of interaction with teacher and also among themselves.

VI) Reusability: The contents should be always verified and if required updated before publishing and so it should be reused. The packaged content should be exportable and should also be able to import other contents.

VII) Maintainability: The e-learning software should be easy to maintain, proper packaging and use of metadata are crucial in this regard. Students’ learning paths and all the interactions should be recorded in order to increase maintainability.

VIII) Availability: The learning content should have full multimedia support. The proper linking and synchronization among the different component increases availability of the software. The individual personalization feature increases the availability. The contents should be always available when searched. The student should have the facility to communicate with others and tutor at any time, this will increase availability.

IX) Interoperability: The learning contents should be able to be shared with different e-Learning software that follow the same standardization process. The metadata is used to exchange information between two systems.

X) Conformance: The e-Learning software should conform the established standard in e-Learning[3,8]. The learning content should comply with Shareable Content Object Reference Model (SCORM)[10] and the overall architecture should comply specifications of Learning Technology System Architecture (LTSA)[11].

<table>
<thead>
<tr>
<th></th>
<th>Course Management</th>
<th>Content Management</th>
<th>Learning Activity</th>
<th>Communication</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>R3, R4, R7, R9</td>
<td>R10, R15</td>
<td>R21</td>
<td>R25, R38, R39</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>R1</td>
<td>R12</td>
<td>R20</td>
<td>R22, R23</td>
<td>R35, R36, R40</td>
</tr>
<tr>
<td>Robustness</td>
<td>R3</td>
<td></td>
<td></td>
<td></td>
<td>R30, R31, R32</td>
</tr>
<tr>
<td>Portability</td>
<td>R2</td>
<td>R14, R16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td>R1</td>
<td>R12</td>
<td>R17, R18, R21</td>
<td>R24, R25, R26, R27</td>
<td>R28, R37</td>
</tr>
<tr>
<td>Reusability</td>
<td>R2, R4</td>
<td>R16</td>
<td>R21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>R2, R3</td>
<td>R18, R19</td>
<td></td>
<td>R31, R32</td>
<td>R33, R34</td>
</tr>
<tr>
<td>Availability</td>
<td>R11, R12</td>
<td>R17, R30, R23</td>
<td></td>
<td>R37</td>
<td></td>
</tr>
<tr>
<td>Interoperability</td>
<td>R2</td>
<td>R11, R12</td>
<td></td>
<td>R42</td>
<td></td>
</tr>
<tr>
<td>Conformance</td>
<td>R15</td>
<td></td>
<td></td>
<td>R43</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Mapping requirements with the software quality attributes.
7. Software testing for LMS:
For an e-learning system features are required to be tested against the functional requirements discussed in sec. The most relevant characteristics to assess the quality of Web applications are usability, functionality, reliability, efficiency and maintainability. As LMSs are mostly web based applications, the complexity of testing web based applications are adherent to it. The main challenge in web application testing is its heterogeneous nature. These are component based applications that run on different layers, Client, Server and Middleware. Moreover these components use different communication protocol and issues like Session handling, Cookie handling, Security etc. makes the job of testing more difficult.

7.1 The V model of testing:
The basic objective behind testing of web applications is same as that of conventional software testing. Biezer described six levels of software testing namely, unit test, module test, integration test, subsystem test, system test and acceptance test. Testing of software is a continuous process throughout the development life cycle. The V model shows how different types of testing can be linked with different phases of software development.

7.2 Testing Strategies:
There are two general testing approaches: black box testing and white box testing. Black box testing approaches create test data without using any knowledge of the structure of the software under test, whereas white box testing approaches explicitly use the program structure to develop test data\[12\]. Black box testing approaches are usually based on the requirements and specifications, while white box testing is usually based on the source code. So white box testing is typically done during unit test, and black box testing is typically applied during integration test and system test.

7.3 Model based Testing for Web Applications:
Model-based testing these days has gained immense importance because of the proliferation of the model based software development model. This can be applied in two different types of scenarios. One type of scenarios considers the shared model for testing and code generation purpose. Such models are not always suitable for testing purpose, since models used for code generation purpose need to be very detailed \[15\]. Another type of scenarios considers the model for testing purpose only, which is exclusive model for testing purpose. Such testing specific models require certain level of abstractions on the System Under Test (SUT), which is more suitable for model-based testing\[16\]. Though there are many models but FSM and UML have emerged as two mostly accepted MBT tools for testing of web applications.
7.4 FSM based Testing of LMS:
Finite state machines (FSM) model for testing is used for modelling complex behaviour of software without the need to consider the software’s underlying implementation. Theoretically, a web application’s behaviour could be modelled using FSMs and then test cases could be automatically generated by traversing the paths through the FSM model of the application, with each distinct path comprising a single test case[16].

FSM can be described formally as quintuple $(I, S, T, F, L)$, where
- $I$ is the set of inputs of the model
- $S$ is the set of all states in the model
- $T$ is a function which determines whether a transition occurs when an input is applied to the system in a particular state
- $F$ is the set of final states where the model can end up
- $L$ is the state where the software is launched

7.4.1 For example we consider the following scenario:
The user can use searching to find out the downloadable links at any time but can download from the link only if she is logged in; otherwise she is redirected for login. One can login only if she is registered. Now both logging in and registration process can be failed if given wrong input.

A Finite state machine (FSM) of this behavior model, which consists of states and transitions between those states, can be modeled as shown in Figure 2.

Now considering $S_0$ as starting state and $S_3$ as terminating state, test cases can be created like:
- $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3$
- $S_0 \rightarrow S_1 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3$
- $S_0 \rightarrow S_1$ etc.

The problem is however, arises due to the vast number of choices most web applications provide to a user. A single page of a web application can be designed to accept numerous different pieces of data as well as allowing the data to be entered in arbitrary order. As a consequence more states and transitions are required for FSM and ultimately will suffer from state space explosion.
So, while FSMs provide a solid groundwork for modelling the complex behaviour of web applications, unhampered by the implementation complexities of web applications, a technique is needed which reduces the size of the FSM while still providing enough detail to generate a sufficient number test cases.

7.5 UML based testing:
The Unified Modelling Language (UML) has emerged as an industrial standard for modelling software systems these days. The UML is a visual modelling language that can be used to "specify, visualize, construct, and document the artifacts of a software system [13]. The behavioural aspect of UML modelling can be dealt with use case diagrams, interaction diagrams, activity diagrams, state chart diagrams. Table 4 [shows how different diagrams are used for different approaches of testing [13]. In this paper we will explore operational behaviour of the said system [see 7.4.1] using use case and activity diagram.

<table>
<thead>
<tr>
<th>Test type</th>
<th>UML diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit</td>
<td>Class, State chart</td>
</tr>
<tr>
<td>Functional / Module</td>
<td>Interaction, Class</td>
</tr>
<tr>
<td>System</td>
<td>Use case, Activity, Interaction</td>
</tr>
<tr>
<td>Regression</td>
<td>Same as Functional</td>
</tr>
<tr>
<td>Solution</td>
<td>Use case, Deployment</td>
</tr>
</tbody>
</table>

Table: 4

Use case diagrams present relationships between actors and use cases. Use cases describe the behaviour of the system by showing how actors interact with the system. An use case diagram for representing the above problem is given below.

In UML, an activity diagram extracts the core idea from flowcharts, state transition graphs and Petri nets [11]. It is a flow chart that explains the use case. In an activity diagram, an activity state is shown as a box with containing a description of the activity; simple completion transitions are shown as arrows; branches are shown conditions on transitions or as diamonds; fork or join of control is used for multiple branching. The activity diagram of the above said requirement is as follows.
The basic flow contains the most popular sequence of actions, the steps that happen when everything goes correctly. Alternative flows represent variations of the flow, including less usual cases and error conditions. Now to create test cases from the UML diagrams we need to identify all of the scenarios for the given SUT including basic flow and alternative flow. A scenario describes one specific path through the flow of events. Form figure 9 we can produce different scenarios like

S₀ → S₁ → S₂ → S₃
S₁ → S₂ → S₃
S₂ → S₁ → S₃ etc.

8. Conclusion:
The requirements we have identified are at the first level of abstraction which does not include the classification in terms of functional and non-functional requirements. These requirements are from the stake holders’ point of view and we have accordingly classified them on software quality. As a result in our classification of the requirements we have often found some requirements overlapping in different clusters which we think is quite obvious. We have analysed the quality of web based LMS with respect to these requirements. Finally we have explored two main techniques of model based testing, namely Finite State Machine and UML based modelling and sample test cases are prepared from a case study.

References:
Jannicke Baalsrud Hauge, Heiko Dün1, Manuel Oliveira, Klaus-Dieter Thoben(2006) “User Requirements Analysis for Educational Games”
www.manubuild.net/projects/408/ICE%202006/Training.../p49-47.pdf
“Shareable Content Object Reference Model (SCORM)" URL: http://www.adlnet.gov/scorm
Clay E. Williams (2006), "Software Testing and the UML",1st workshop on model-based testing and object-oriented systems
Peter Zielczynski, (2003)“Traceability From Use Cases To Test Cases”, Rational Users' Conference
Jim Heumann(2001),"Generating Test Cases From Use Cases", The Rational Edge