2011 International Conference on Information Communication and Management
IPCSIT vol.16 (2011) © (2011) IACSIT Press, Singapore

On Integrating User Acceptance Tests Generation to Requirements Management

Chavalid Ieamsaard [1] and Yachai Limpiyakorn [2 +]

[1, 2] Department of Computer Engineering, Chulalongkorn University, Bangkok 10330, Thailand
[1] Chavalid.cpe@gmail.com, [2] Yachai.L@chula.ac.th

Abstract. Once the phase of requirements analysis has finished, user acceptance tests can be derived from the user requirements. Since testing is resource consuming and it is inevitable for requirements changes in a software project, this paper presents an automation approach to generating user acceptance tests from use case descriptions. Additionally, the relationships between use cases and their test cases are exported to traceability matrix automatically. The efforts of this research contribute to the improvement of software development process. That is, it could reduce the cost of testing and provide the ability to better manage requirements changes.

Keywords: Use Case, User Acceptance Tests, Requirements Management, Software Process Improvement.

1. Introduction

The V model or the extended model of the waterfall model serves as a guide for the integration of quality control activities (i.e. reviews and testing) throughout the software development life cycle. The horizontal and vertical axes represent time or project completeness (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively.

During the first phase of software development life cycle called Requirements Analysis, the needs of the users are elicited and transformed to the requirements of the proposed system. This phase is concerned with establishing what the ideal system has to perform. However it does not determine how the software will be designed or built.

Prior to the release of the product to the customer, user acceptance test or UAT will be performed. UAT is the phase of testing used to determine whether a system satisfies the requirements specified in the requirements analysis phase. The acceptance test design is derived from the requirements document which typically describes the system's functional, interface, performance, data, security, etc. In general, the requirements document is used by business analysts to communicate their understanding of the system to the users and the user acceptance tests are designed in this requirements analysis phase. The user acceptance tests aim at verifying and validating the implemented system in the real world by the intended audience according to the original needs.

In addition to the fact that testing is a resource consumption activity, requirements changes aggravate the cost of testing. It is inevitable for requirements changes in a software project. Automating test case generation could lessen the cost of testing as well as reduce the flaws in test cases due to human errors. This research therefore presents an approach to generating test cases from requirements document, and serving as black-box tests used in user acceptance test.

2. Background

+ Corresponding author. Tel.: (+66) 2-218-6968; fax: (+66) 2-218-6955.
E-mail address: Yachai.L@chula.ac.th.

2.1. Use case [1]

Unified Modeling Language provides a comprehensive notation for communicating the requirements, architecture, implementation, deployment, and states of a system. UML standardizes several Diagramming methods for embracing object-oriented analysis and design. The UML 9 diagrams can be categorized into Structural diagrams and Behavioral diagrams. Use Case diagram, which is a type of behavioral diagrams, is used as a source for deriving the test cases for user acceptance testing in this research work.

In UML, use case diagrams model the behavior of a system and help to capture the requirements of the system. The purpose of use case diagrams is to present a graphical overview of the functionality provided by a system in terms of actors, their goals represented as use cases, and any relationships between those use cases. Four relationships among use cases are used often in practice**,** namely *include, extend, generalization, and associations*.

Use cases treat the system as a black box. One or more system requirements describe the functionality needed to meet a user need or user requirement. Each system requirement can be analyzed into one or more use cases. Each use case should be traceable to its originating system requirement, which in turn should be traceable to its originating user need.

Each use case describes a particular goal for the user and how the user interacts with the system to achieve that goal. Some development processes require detailed use cases to define requirements. Accompanied with a use case, use case description is the documentation that details the behavior of objects and flow of events in a use case. Example of use case description is shown in Table 1.

Table1 Example of Use Case Description

ID:	Use case type:	Importance level:
Use Case Name:		
Primary actor:		
Stakeholders and interests:		
Brief description:		
Trigger:		
Type:		
Relationships:		
Association:		
Include:		
Extend:		
Generalization:		
Normal flow of Events:		
Subflows:		
Alternate/Exceptional flows:		

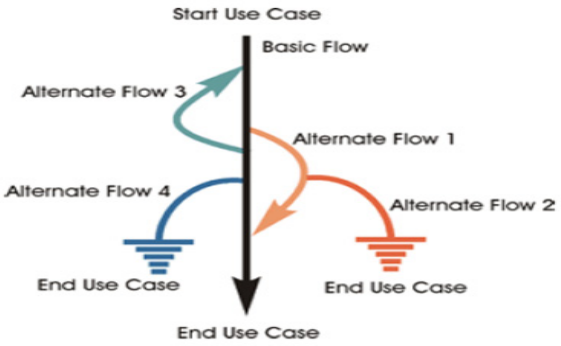


Fig. 1: Flow of Events in Use Case.

Flow of events [2] can be divided into three types: 1) Basic flow, 2) Alternate flow, and 3) Exception flow as illustrated in Fig.1. Basic flow depicts a perfect situation where nothing goes wrong. Alternate flow is a pathway that is not the most heavily travelled one. Exception flow is a path for an error condition.

One of the purposes of use cases is to provide a basis for testing to ensure that the system works as intended. It is thus possible to automatically generate black-box tests from use cases.

2.2. Extended finite state machine [3]

Finite State Machine (FSM) and Extended Finite State Machine (EFSM) are commonly used in Model Based Testing. An EFSM is the enhanced model based on the traditional finite state machine or finite-state automation. It is defined by the tuple M= {Q, I, O, D, T}, where Q is a set of control states, I is a set of inputs, O is a set of outputs, D is a *n* Dimensional space $D_1 \times \ldots \times D_n$, T is a transition relation, T: Q × D × I → Q × D × O. An EFSM can be represented by the control flow graph which illustrates the transformation of events from state to state. Therefore, Extended Finite State Machines are useful and help the derivation of test paths or test scenarios from use case descriptions in this work.

3. Research Methodology

Once the requirements analysis phase has been committed, the user acceptance tests can be derived from the requirements document. This research presents an automation approach to generate test cases for UAT from Use Case Descriptions. It is assumed that Use Case Descriptions (UCD), Functional Requirements (FR), and Non-functional Requirements (NFR) are well-defined in the Software Requirements Specification (SRS)

as the input document to the system. Example of the Use Case Description used in this work is depicted in Fig. 2. The input text file, SRS, will be transformed into XML format and the details inside each UCD associated with the list of FR and NFR will be extracted by the component implemented in the research work [4], which is part of our Requirements Management system. Fig.3 illustrates the overview of the process including the following steps:

ID:	UC02
Use case type:	Real Use Case
Importance level:	High
Use Case Name:	Register
Primary actor:	AC1:Student
Brief description:	When students log in to Registration System. A course for students to register. System can be check course and student groups. Upon completion, all subjects will be invoiced registration form to the student registration.
Trigger:	Student (Actor) Login to Online Registration System website. Students specify year and semaster . Select the course student want to register. (Student can select more than one subject).
Normal flow of Events:	1. Student requests for register. 1.1 Creating a registration form. 2. Students specify the course you wish to register. 2.1 System check the Course and a group of subjects (UC03). 3. Students choose courses. 4. Student register confirmation. 4.1 System must additional courses, students register on the Registration Form. 4.2 Create a list of register (Register Transaction). 5. Students print the invoice registration form. 5.1 calculate the invoice registration form. 5.2 The system generates an invoice and print it.
Alternate flows:	2a. System can not find the course you wish to register. 2a1. Students find information on courses offered and check the code out of it again. 2a2 to step 2. 3a of the seat is not enough. 3a1 warning system to notify the student in mind. 3a2 to step 3 again.
Exceptional flows:	-
Pre-condition	The following information must have been students, more courses. And courses offered.
Post-Condition	Registration and the course is completed. Print invoices will be done. Number of students register in the group must have adequate Course registration is related to the student.

Fig. 2: Example of Use Case Description.

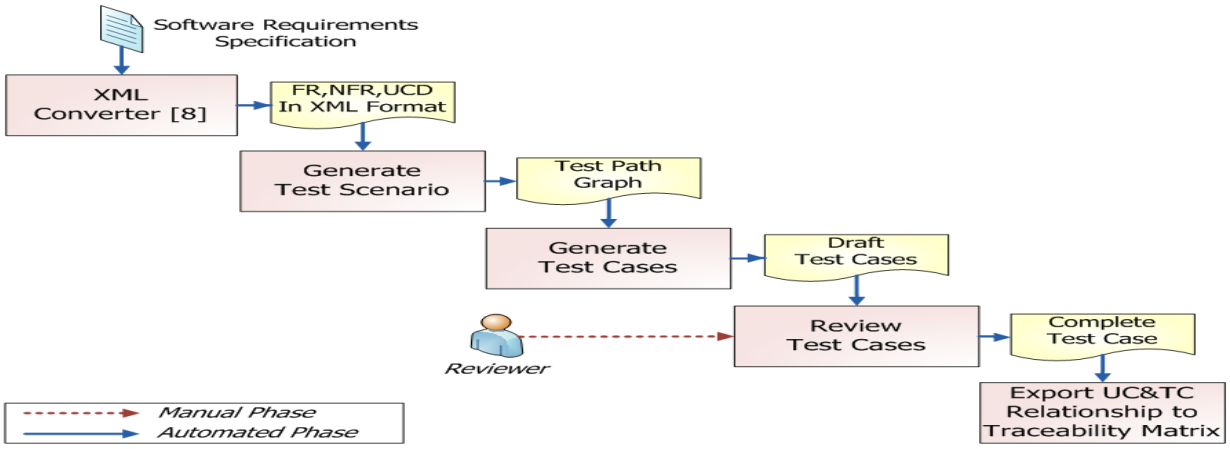
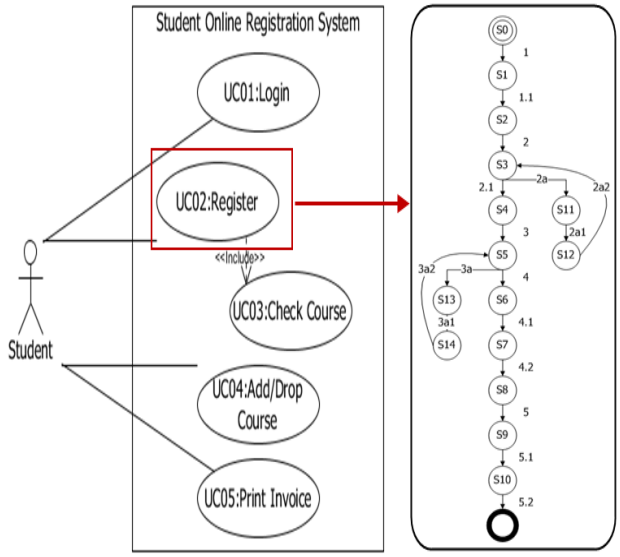


Fig. 3: Overview of the System.

3.1. Generate test scenarios

Extended Finite State Machine (EFSM) presented in [3] is applied to generate the test path graph as illustrated in Fig. 4. The use case "UC02" contains four test paths as listed in the right-hand-side table of Fig. 4. Each test path represents each test scenario or each flow of event residing a Use Case Description. Each UCD contains a set of test scenarios, and in turn one or several UCDs may fulfill a Functional Requirement/

Non-Functional Requirement. The information of the associations between UCDs and a FR/ NFR, as well as those between a UCD and its test scenarios are stored in the repository.



Path	Use Case Register Scenario path	Path Type
1	1 - 1.1 - 2 - 2.1 - 3 - 4 - 4.1- 4.2 -5 - 5.1 - 5.2	Normal Flow
2	1 - 1.1 - 2 - 2a - 2a1 -2a2 - 2 - 2.1 - 3 - 4 - 4.1- 4.2 -5 - 5.1 - 5.2	Alternate Flow 1
3	1 - 1.1 - 2 - 2.1 - 3 - 3a - 3a1 - 3a2 - 3 - 4 - 4.1- 4.2 -5 - 5.1 - 5.2	Alternate Flow 2
4	1 - 1.1 - 2 - 2a - 2a1 -2a2 - 2 - 2.1 - 3 - 3a - 3a1 - 3a2 - 3 - 4 - 4.1- 4.2 -5 - 5.1 - 5.2	Alternate Flow 3

Fig. 4: Example Set of Test Paths or Test Scenarios Derived from UCD—"UC02".

3.2. Generate test cases

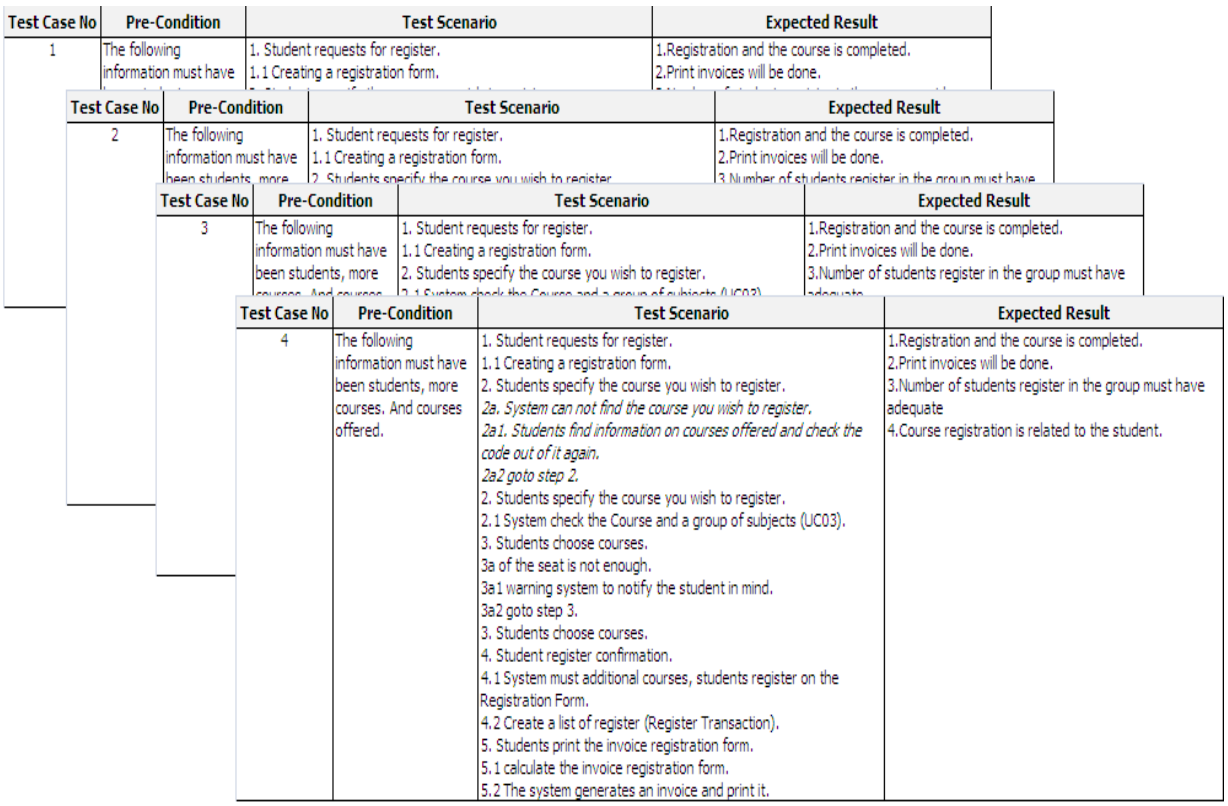
Each test scenario represents a test case (TC). The details of a test case will be automatically fulfilled in this step, i.e. Pre-condition, Test Scenario, and Expected Result. According to Fig. 4, four test cases associated with "UC02- Register" will be generated. That is, TC No.1 represents the normal scenario of successful student registration. TC No.2 represents the first alternate scenario when students miss the course ID they would like to register. They need to check for the valid course ID ("UC03"), then back to the normal path again. TC No.3 represents the second alternate scenario when the number of students registering a certain course over limit. The system will notify the situation and back to normal path. TC No.4 represents the third alternate scenario which is the combination of the first and second alternate scenarios before back to the normal path.

3.3. Review test cases

This step is performed by a reviewer or test team to verify the correctness and completeness of test cases generated from the previous step. Example of a set of test cases associated with a Use Case—"UC02" is illustrated in Fig. 5.

3.4. Export data to traceability matrix

The relationships between use cases and functional requirements/ non-functional requirements, as well as the associations between uses cases and their test cases are exported to the component implemented in the research work, which will extract the details and input them into the traceability matrix implemented in the research work [5].



Test Case No	Pre-Condition	Test Scenario	Expected Result
1	The following information must have	1. Student requests for register. 1.1 Creating a registration form.	1.Registration and the course is completed. 2.Print invoices will be done.
2	The following information must have	1. Student requests for register. 1.1 Creating a registration form.	1.Registration and the course is completed. 2.Print invoices will be done.
3	The following information must have been students, more	1. Student requests for register. 1.1 Creating a registration form. 2. Students specify the course you wish to register.	1.Registration and the course is completed. 2.Print invoices will be done. 3.Number of students register in the group must have
4	The following information must have been students, more courses. And courses offered.	1. Student requests for register. 1.1 Creating a registration form. 2. Students specify the course you wish to register. *2a. System can not find the course you wish to register.* *2a1. Students find information on courses offered and check the code out of it again.* *2a2 goto step 2.* 2. Students specify the course you wish to register. 2.1 System check the Course and a group of subjects (UC03). 3. Students choose courses. 3a of the seat is not enough. 3a1 warning system to notify the student in mind. 3a2 goto step 3. 3. Students choose courses. 4. Student register confirmation. 4.1 System must additional courses, students register on the Registration Form. 4.2 Create a list of register (Register Transaction). 5. Students print the invoice registration form. 5.1 calculate the invoice registration form. 5.2 The system generates an invoice and print it.	1.Registration and the course is completed. 2.Print invoices will be done. 3.Number of students register in the group must have adequate 4.Course registration is related to the student.

Fig. 5: Example Completed Test Cases of UCD—"UC02".

4. Conclusion

This paper presents a method and a system implemented to generate user acceptance tests from use case descriptions once the requirements analysis phase has been committed. It is assumed that the well-defined use case descriptions are contained in the Software Requirements Specification as the input text file to the system. The Software Requirements Specification will be converted to XML format so that the details of use case descriptions can be extracted to generate the test paths from the flow of events inside each use case description. Each test path represents a test case which will be fulfilled and reviewed the details. Finally, the relationships between use cases and their test cases will be exported to the traceability matrix to serve the requirements management purpose. The benefits of this research are to reduce the cost of testing and to improve the performance of software process.

5. References

[1] Object Management group. (2010). *Unified Modeling Language: Superstructure version 2.3* [Online]. Available: http://www.omg.org

[2] H. Jim. (2001). *Generating Test Cases from Use Cases* [Online], IBM Journal, Available: htttp://www.ibm.com/developerworks/rational/library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01.pdf

[3] A. Sinha, A. Paradkar, and C. Williams, *On Generating EFSM Models from Use Cases*, Proceedings of the Sixth International Workshop on Scenarios and State Machines (SCESM'07: ICSE Workshops 2007), Minneapolis, MN, USA, May 20-26, 2007, pp.1-1.

[4] Y. Phopan, and Y. Limpiyakorn, *Approach to Automating Input Data for Requirements Traceability Matrix*, Proceedings of the National Graduate Research Conference 2011, Buriram, Thailand, Aug 11, 2011, pp. 1033-1042.

[5] S. Soonsongtanee, and Y. Limpiyakorn, *Enhancement of Requirements Traceability with State Diagrams*, Proceedings of IEEE International Conference on Computer Engineering and Technology, Chengdu, China, Apr 16-18, 2010, pp. 248-252.