Journal of Theoretical and Applied Information Technology
15th April 2017. Vol.95. No 7

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A STUDY OF GENERATING ABSTRACT TEST FOR REQUIREMENTS VALIDATION AMONG REQUIREMENTS ENGINEERS

[1]NOR AIZA MOKETAR, [1]MASSILA KAMALRUDIN, [1]MOKHTAR MOHD. YUSOF, [1]SAFIAH SIDEK, [2]MARK ROBINSON

[1]Innovative Software System and Services Group, Universiti Teknikal Malaysia Melaka, MALAYSIA

[2]Fulgent Corporation, Texas, USA

E-mail: [1]nor.aiza09@gmail.com, {massila, mokhtaryusof, safiahsidek}@utem.edu.my
[2]marcos@fulgentcorp.com

ABSTRACT

Requirements testing or requirements-based testing (RBT) is one of the software testing techniques that is found effective to test requirements' completeness and accuracy. This technique involves systematic way of test case generation from the model of the requirements specification. This technique has been applied in the requirements analysis phase to detect and eliminate requirements defects before the next stage of software development project. Although this technique is useful, it is tedious and time consuming to manually generate abstract test from the requirements model. However, we argue that the tedious process can be minimised if the requirements engineer have the good ability (skill) to generate abstract test from requirements models for requirements validation. This paper described a study of requirements engineer manually generate abstract tests from requirements model: Essential Use Cases (EUC) model. From the result, we discover that software requirements engineers are not well equipped with the skill and technique to generate abstract tests from requirements model.

Keywords: *Requirements Validation, Requirements-Based Testing, Abstract Tests, Test Requirements, Test Cases*

1. INTRODUCTION

Capturing correct and consistent requirements from client-stakeholders determines the production of a quality software development. However, it is often considered to be difficult, time consuming and error prone [1][2]. Requirements validation has been recognised as an important process to produce quality software as it determines whether client-stakeholders' needs and expectations of a product are sufficiently correct and complete [3][4][5][6]. Further, the common practice of conducting requirements validation at late stages of product development is costly and time consuming [7][8]. One way to overcome this problem is to perform requirements validation early in software development. This proactive approach, known as the test-driven development allows early detection and prevention of errors in requirements specifications as it begins with writing the test cases and then follows with implementation, hence avoiding the need to rectify errors at later stages of product development [9][10].

At present, various requirements validation techniques, such as requirements review, inspections, prototyping, model-based, requirements testing and viewpoint-oriented requirements validation [3][11][12] have been used to evaluate the correctness and quality of requirements. Each of these techniques has their own strengths and weakness depending on the purpose of their usage. Studies [13][14][15] have recognized requirements testing or requirements based testing as an effective technique to identify requirements defect. Requirements testing or also known as requirements-based testing (RBT) is a software testing technique that is use for the purpose of verification and validation (V&V) of a developed software application by deriving test cases from the requirements [13][14]. It is a specification-based (black-box) testing technique or input-output driven testing techniques as the software is viewed as a black-box with input and output that solely derived from the specifications, without concern for the internal structure of the program [16]. This technique use systematic way to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

derive test cases from a model (formal or informal) of the requirements specification. Designing tests with users' involvement in the earliest stage will help the users to understand what they really want the system to do. This will help to discover and eliminate requirements defects before the design and development phase, which will also help to cut the project cost.

Although this technique is found useful and effective to test requirements' completeness and accuracy, it is costly, time consuming and challenging to manually generate the test cases from requirements model [14][17]. As a result, most development organizations are reluctant to invest their time and effort in designing test cases in requirements phase. Furthermore, it is difficult to get client-stakeholders' cooperation to get involve in the process due to time-constraint and other responsibility. However, we argue that the tedious process can be reduced if the software practitioners have a good skill in generating abstract tests from requirements. We also found there is almost no similar study that investigate the challenges faced by requirements engineers in generating/defining test cases from requirements model. Herein, we present the user study to measure the software requirements engineers' ability (skill) to generate abstract test from requirements model as well as to understand the difficulties in the process.

The remainder of this paper is organized as follows: Section 2 outlines the background of the study; Section 3 present the research design of our user study; Section 4 discusses the result of the experiment; Section 5 described the discussion and lesson learn from the study; Section 6 describes the threats to the validity of our study and Section 7 conclude this work.

2. BACKGROUND OF STUDY

We use the term 'abstract tests' to refer to our test requirements and test cases that are generated from the semi-formalised abstract model, called the Essential Use Cases (EUC) and the Essential User Interface (EUI) model. An abstract test is a high-level test requirement and test case that represents a requirements scenario. In contrast to concrete tests, an abstract test does not contain any details of the test environment, test protocol, or configuration for the test component.

2.1 Essential Use Case (EUC) and Essential User Interface (EUI)

EUC is a structured narrative, expressed in the language of the application domain and the users. It is composed of a simplified, abstract, technology-free and implementation-independent description of a single task or interaction [18][19]. EUC is a complete, meaningful, and well-designed interaction from the point-of-view of the users. It represents a particular role in relation to a system and embodies the purposes or intentions underlying the interaction. EUCs enable users to ask fundamental questions, such as "what's really going on" and "what do we really need to do" without letting implementation decisions get in the way. These questions often lead to critical realisations that allow users to rethink, or reengineer the aspects of the overall business process. Figure 1 shows an example of natural language requirements (left hand side) and an example of an EUC (right hand side) while capturing the requirements (adapted from [19]). The natural language requirements from which the important phrases are extracted (highlighted) are shown on the left hand side of Figure 1. From the natural language requirements, a specific key phrase (essential requirement) is abstracted and is shown in the EUC on the right hand side of Figure 1. As shown in Figure 1, the EUC depicts two interrelated sets of information: the user intentions and the system responsibility.

An EUI prototype is a type of abstract prototype or paper prototype that is a low-fidelity model. Also known as a "UI prototype" for a software system, it represents the general ideas rather than the exact details of the UI [19][20]. An EUI prototype represents the user interface requirements in a technology independent manner; just as the EUC models do for the behavioural requirements. An EUI prototype is particularly effective during the initial stages of user interface prototyping for a system. It models user interface requirements that are evolved through analysis and design to the final user interface of a system [8]. It also allows some exploration of the usability aspects of a system. Figure 2 shows an example of an EUI prototype developed from EUC model. The possible UI functionality at a high level of abstraction is captured from the user intention/system responsibility dialogues.

Both EUC and EUI play important roles in our work. The EUC provides a simpler and shorter form of dialogue between the user and the system compared to the conventional use case. This dialogue provides the key information of the input and output (expected results) for our test cases. An interaction (input and output) between the user and the system can generate one or more test requirements. This dialogue also provides information for the test procedures/steps in our test cases. The EUI prototype model provides a guide

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

for the important elements to be included in our mock-up UI prototype. These two models are crucial to ensure the correctness, completeness and consistency of generated abstract tests and mock-up UI prototypes for users' requirements.

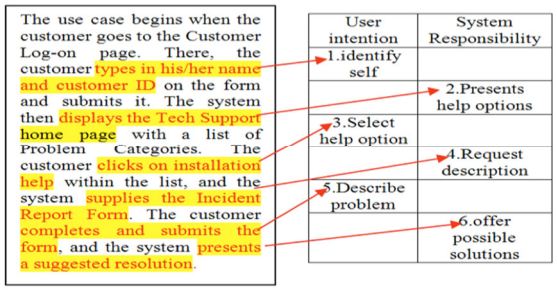

Figure 1: Example of EUC Model (Right-hand Side) Extracted from Natural Language Requirements (Left-hand Side)

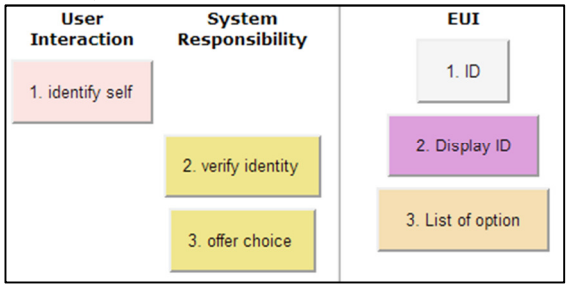

Figure 2: Example of EUI Prototype Extracted From the EUC Model

3. STUDY GOAL AND METHOD

This section describes the design of our user study. Our user study seeks to test the skill (ability) of requirements engineers to correctly generate abstract test (test requirements and test cases) from requirements model.

3.1 Experiment Subjects

For this user study, the subjects were 30 undergraduate students from the course of software testing and quality assurance, in their final year of degree in computer science at the Universiti Teknikal Malaysia Melaka (UTeM). The student previously attended courses on software engineering and project management. In general, they had similar knowledge and expertise level in software engineering and software testing topics. The sample of subjects participating in the experiments was on a voluntary basis and agreed to participate to the experiment. The participants were provided with a written informed consent form. They were informed that: (i) the experiment is not mandatory, (ii) they will be observed while performing the task, (iii) they were not evaluated on their performance and (iv) data collected will be used only for research purposes.

3.2 Experiment Materials

The study material consisted of a tutorial and a set of requirements sample. The tutorial explained the EUC and EUI model that are used as the requirements model in this experiments. We also provide theoretical and practical lesson on how to generate abstract test from EUC model. We provide the participants with requirements sample for a Patient Management System (PMS) in the form of use case scenario. This system targeting the hospital or clinic administrators who manage the patient information. The admin can log in to the system, view the list of patients and add the patient's medication information list. We also provide the associated EUC model derived from the sample requirements as shown in Table 1.

Login Module:
Pre-condition: User must register to the system.
1. *Anonymous user needs to login with their User ID and password to use the system.*
2. *System will verify and validate the User ID and password.*
3. *Upon successful authentication, system will display the menu choices.*

View Patient Details:
Pre-Condition: User must be logged-in to the system.
1. *User chooses the option to view the list of patient from the menu choices.*
2. *System will display the list of patients stored in the database.*
3. *User chooses a patient ID from the list.*
4. *System will display the information of the selected patient ID.*

Update Medication:
Pre-Condition: User must be logged-in to the system.
1. *User adds a new medication to the patient's con-med list.*
2. *System will update the new medication to the database.*

Table 1: The EUC Model Generated From PMS Use Case Scenario

Module	EUC Model	
	User Intentions	System Responsibility
Login	Identify self	
		Verify Identity
		Offer choice
View Patient Details	Select option	
		Display Information
Update Medication	Add option	
		Update Information

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.3 Variable Selection

The dependent variables of our study are the participant's comprehension level and time taken to generate/define the abstract test from EUC model. The comprehension level is to measure the participants' ability and skills, meanwhile the time is to measure the effort need to generate abstract test from requirements model. The comprehension level has been measured by checking the correctness of the generated abstract test to the EUC model. Our perspective of correctness in this study is the combination of consistency and completeness of the generated abstract test. Table 2 displays our correctness measurements in this experimentation. As described in Table 2, we have three (3) test requirements and five (5) test cases for the login module in our pattern library. We say the participant has a *correct* abstract test when he or she has generate the same number of abstract test with the similar definition of abstract test from our pattern libraries. A participant response is *partially correct* if he or she has defined fewer or more abstract test. A participant response is considered *incorrect* if none of the defined abstract test matches with our pattern library. We give one point to correct answer, half a point for partially correct answer and zero point for incorrect answer. Table 3 show an example of the abstract test generated from our pattern library.

3.4 Experiment Procedure

We have defined and followed a simple procedure to carry out the experimentation. We asked the participants to manually generate abstract test from EUC model. Prior to that, they were given a short description of the experimentation. We provide a tutorial that explained the theory of EUC model in detail and give an example on the process of generating abstract test from the model. We give them 15 minutes to understand the concept and hands-on the example as given in the tutorial. Then, the experiment went through the following steps:

(1) Subject had 10 minutes to read the sample requirements of PMS system.
(2) Subject had to write their matric card number and start time.
(3) Subject had to write the abstract test on the provided sheets. To reduce the complexity and time taken, the subject only need to write down the test requirements, test case description, input/test data and the expected output of the abstract test.
(4) Once completed, subjects had to write down the stop time and call the researcher.

3.5 Data Analysis Protocol

Upon completion of the task, the participants need to call the researchers and submit the handouts. There were two researchers involved in the data analysis for this study. To measure the manual effort, we calculated and averaged the time taken of the participants to finish the task. Then, we checked and compared each of the abstract test written by the participants with our abstract test's pattern to measure the comprehension level of the participants. For this, we checked the consistency and completeness of the participants' responses with our abstract test pattern library as shown in Table 3. We gave the relevant point for each responses following the correctness measurements as described in Table 2. We calculated the points and get the percentage for each comprehension level: good, moderate and poor. The results is discussed in Section 4.

4. RESULTS

4.1 User Study: Manual Extraction of Abstract Test

Table 4 summaries the result for comprehension level of the participants. We classified the comprehension level as good, moderate and poor based on the correctness of the abstract test. The result shows that only 10.13% of the participants have good skill (ability) to generate the correct abstract test from the EUC model. More than half of the participants have moderate skill, which is 57.38% who generate partially correct abstract test and 32.49% of the participants have poor skill to generate the abstract test. Based on the results, participants were most likely to generate incomplete abstract test, as they tend to miss few test cases that were associated with specific test requirements. For example, in module 1 (login), most participants only create positive (pass) test cases and miss identifying the negative (fail) test cases. Thus, from this result we conclude that software requirements engineer is *not* able to generate abstract test from requirements model. Further, we also have average the completion time of all the participants. The mean time taken to accomplish the task was 1 hour and 15 minutes (75 minutes). The shortest time taken was 55 minutes to complete the task. This study also demonstrates that it is time consuming and tedious for participants to generate correct, complete and consistent abstract test from the requirements model. Our study thus supports the claims [14][17][21][22] that writing test cases as part of the requirements-based testing process are tedious and time consuming.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: Correctness Measurement

Module	Pattern Library		Participant Response	
	# of Test Requirements	# of Test Cases	# of Test Requirements	# of Test Cases
Login	3	5	3 matches = correct 2 or only 1 matches = partially correct 0 match = incorrect	5 matches = correct 4 or 3 or 2 or 1 matches = partially correct 0 match = incorrect
View Patient Details	2	2	2 matches = correct 1 matches = partially correct 0 match = incorrect	2 matches = correct 1 matches = partially correct 0 match = incorrect
Update Medication	2	2	2 matches = correct 1 matches = partially correct 0 match = incorrect	2 matches = correct 1 matches = partially correct 0 match = incorrect

Table 3: The Abstract Test Generated From Our Pattern Library

EUC	Test Requirements	Test Case				
		TC Description	Pre-condition	Test Data	Steps	Expected Output
Identify self	Validate that user can login with valid username and password	Valid username and password	User should be registered to the system.	Username: Admin001 Password: Admin00!	1. Key in the username and password. 2. Click on "Login" button.	User should be able to login to the system.
	Validate that user can not login if username and password is invalid.	Valid username and invalid password.		Username: Admin001 Password: Admin002	1. Key in the username and password. 2. Click on "Login" button.	User should not be able to login to the system.

Table 4: The Comprehension Level Results of the Participants

Module	Correct (%)	Partially Correct (%)	Incorrect (%)
Login	5.06	56.96	37.97
View Patient Details	12.66	58.23	29.11
Update Medication	12.66	56.96	30.38
Mean	10.13	57.38	32.49

5. DISCUSSION AND LESSON LEARN

From the results of the experiment we found that only 10.13% of the participants have a good skill to generate abstract test from requirements model. From our observation, we found that the participants took a long time to think and figure out the correct, complete and consistent abstract test from the requirements model. This is a big concern as in a real software development environment the requirements can be more complex compared to the sample requirements provided in the experiment. Further, we also found that it was difficult for us (researchers) to read and understand the participants' responses (the generated abstract test). Such difficulties includes: (1) inconsistent terms to explain the same thing, (2) the abstract test statements are unclear and ambiguous, (3) grammar and typographical errors. This have motivate us to develop an automated tool that able to generate abstract test from semi-formalised model: EUC and EUI models in order to assist in requirements validation process. There is also a need to have a proper authoring template to help the requirements engineers to write correct tests to avoid the difficulties as mention above. Moreover, we learn that with a proper tool support, it may help to reduce human effort and time in the process, which eventually will help to cut the production cost.

6. THREATS TO VALIDITY

In this section, we discussed the threats to the validity that can have affected our study, which include the internal and external validity.

Internal validity measures the cause-effect relationship identified in a study [23]. The participants in this study were final year undergraduate students majoring in Software Engineering. They were properly trained to deliver the task before the experiment as described in Section 3. The students were informed that their response was treated anonymously and they were not evaluated on their performance. They were also not aware of the main objective of experimentation.

External validity refers to the degree to which the results of an empirical investigation can be generalised to and across individuals, settings and times [23]. Our results may be generalized to novice requirements engineer who were not well trained to generate abstract test from requirements

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

model. To draw any conclusions for more experience requirements engineer, empirical studies with professional experts are needed. Other threat to external validity is related to the sample requirements that were used in this experimentation. Although the sample requirement is simple, it is realistic to represent the requirements of small management system. Furthermore, we also minimised the component of the abstract test to be generated to reduce the complexity and the time taken to complete the task.

7. CONCLUSION

Requirements engineers' are responsible to ensure the requirements meet the users' need and expectation. They need to perform proper validation to ensure the requirements interpreted correctly by reviewing and validating the design, code as well as the test cases based on the requirements specification. Requirements testing is one of the effective requirements validation technique that involved test case derivation from the requirements. For this, requirements engineers need to be equip with proper technical skills to generate/define the correct and consistent test cases compliance to the elicited requirements.

In this paper, we present a user study that assesses the ability (skill) of software requirements engineer to generate/define abstract test from requirements model. From the study, we found that the software requirements engineers are not well equipped with the skill to generate abstract test from requirements model. The result was quite alarming as in real software development environment the requirements can be more complex compared to the sample requirements provided in the experiment. Yet, as the experimentation was conducted with final year undergraduate student, we summaries and generalised our finding that it applied to novice requirements engineer. Therefore, we need to replicate the experiment with senior software practitioner from the industry to confirm or contradict our hypothesis. For future work, we intended to replicate the experiment with a small group of IT professional from industry to confirm or reject our hypothesis.

ACKNOWLEDGEMENT

This research is funded by Ministry of Higher Education Malaysia (MOHE), Universiti Teknologi Mara (UiTM), Fulgent Corporation, FRGS grant: FRGS/2/2013/ICT01/FTMK/02/2/F00185.

REFRENCES:

[1] M. Kamalrudin and J. Grundy, "Generating essential user interface prototypes to validate requirements," *2011 26th IEEE/ACM Int. Conf. Automated Software Engineering ASE 2011*, 2011, pp. 564–567.

[2] M. Kamalrudin, J. Grundy, and J. Hosking, "Tool support for essential use cases to better capture software requirements," *Proc. IEEE/ACM Int. Conf. Automated Software Engineering - ASE '10*, 2010, p. 255.

[3] U. A. Raja, "Empirical Studies of Requirements Validation Techniques," in *2nd International Conference on Computer, Control and Communication, IC4 2009*, 2009, pp. 1–9.

[4] N. Condori-Fernandez, S. España, K. Sikkel, M. Daneva, and A. González, "Analyzing the effect of the collaborative interactions on performance of requirements validation," in *20th International Working Conference on Requirements Engineering: Foundation for Software Quality, REFSQ 2014*, 2014, vol. 8396 LNCS, pp. 216–231.

[5] D. Aceituna, H. Do, and S.-W. Lee, "SQ^(2)E: An Approach to Requirements Validation with Scenario Question," *2010 Asia Pacific Software Engineering Conference*, 2010, pp. 33–42.

[6] D. Aceituna, H. Do, and S.-W. Lee, "Interactive requirements validation for reactive systems through virtual requirements prototype," in *2011 Model-Driven Requirements Engineering Workshop*, 2011, pp. 1–10.

[7] S. Sukumaran, a. Sreenivas, and R. Venkatesh, "A Rigorous Approach to Requirements Validation," *Fourth IEEE Int. Conf. Softw. Eng. Form. Methods*, 2006 pp. 236–245.

[8] S. Maalem and N. Zarour, "Challenge of validation in requirements engineering," *J. Innov. Digit. Ecosyst.*, vol. 3, no. 1, 2016, pp. 1–7.

[9] T. Hammel, R. Gold, and T. Snyder, *Test-Driven Development: A J2EE Example*. New York, USA: Apress, 2005.

[10] H. Erdogmus, G. Melnik, and R. Jeffries, "Test-Driven Development," *Encycl. Softw. Eng.*, 2010, pp. 1211–1229.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[11] S. B. Saqi and S. Ahmed, "Requirements Validation Techniques practiced in industry : Studies of six companies," Blekinge Institute of Technology, Sweden, 2008.

[12] F. Yousuf, Z. Zaman, and N. Ikram, "Requirements Validation Techniques in GSD : A Survey," in *IEEE International Multitopic Conference*, 2008, pp. 553–557.

[13] T. Sarwar, W. Habib, and F. Arif, "Requirements based testing of software," *2013 Second Int. Conf. Informatics Appl.*, 2013, pp. 347–352.

[14] U. A. Raja, "Empirical studies of requirements validation techniques," in *2009 2nd International Conference on Computer, Control and Communication, IC4 2009*, 2009.

[15] P. Skoković, M. Rakić-Skoković, and Teaching, "Requirements-Based Testing Process in Practice," *Int. J. Ind. Eng. Manag.*, vol. 1, no. 4, 2010, pp. 155–161.

[16] G. J. Myers, *The Art of Software Testing*. 2004.

[17] C. Wang, F. Pastore, A. Goknil, L. Briand, and Z. Iqbal, "Automatic generation of system test cases from use case specifications," in *Proceedings of the 2015 International Symposium on Software Testing and Analysis - ISSTA 2015*, 2015, pp. 385–396.

[18] R. Biddle, J. Noble, and E. Tempero, "From Essential Use Cases to Objects," *forUSE 2002, Second Int. Conf. Usage-Centered Des. forUSE 2002*, vol. 1, no. 978, 2002, pp. 1–23.

[19] L. L. Constantine and L. A. D. Lockwood, "Structure and Style in Use Cases for User Interface Design," vol. 1, no. 978. Addison-Wesley Longman Publishing Co., Boston, MA, 2001.

[20] S. W. Ambler, "Essential (Low Fidelity) User Interface Prototypes," 2003. [Online]. Available: http://www.agilemodeling.com/artifacts/essentialUI.htm.

[21] B. Haugset and G. K. Hanssen, "Automated Acceptance Testing: A Literature Review and an Industrial Case Study," *Agil. 2008 Conf.*, 2008, pp. 27–38.

[22] S. Khandkar, S. Park, Y. Ghanam, and F. Maurer, "FitClipse: A Tool for Executable Acceptance Test Driven Development," *Agil. Process. Softw. Eng. Extrem. Program. - 10Th*, vol. 31, 2009, pp. 259–260.

[23] R. K. Yin, *Case Study Research: Design and Methods Fourth Edition*, vol. 5. 2009.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

APPENDIX

User Study of Generating Abstract Test for Requirements Validation among Requirements Engineers

Statement

	I have read the Participant Information Sheet and have understood the nature of the survey and I agree to take part in this survey. (please tick √)

Matrix No. : _____

Start Time : _____

End Time : _____

Task 1: Identify Test Requirements and Test Cases from use case scenario and EUC models

1. Given the use case scenario and the corresponding EUC model.
2. Identify and list the appropriate test requirements (TR) and test cases (TC) from the EUC model.
3. Use the following table to list your TR and TC.
4. Please note your start and end time to execute this task.
5. Upon completion please return this sheets to the researchers.

Login Module:
Pre-condition: User must register to the system.
1. *Anonymous user needs to login with their User ID and password to use the system.*
2. *System will verify and validate the User ID and password.*
3. *Upon successful authentication, system will display the menu choices.*

View Patient Details:
Pre-Condition: User must be logged-in to the system.
1. *User chooses the option to view the list of patient from the menu choices.*
2. *System will display the list of patients stored in the database.*
3. *User chooses a patient ID from the list.*
4. *System will display the information of the selected patient ID.*

Update Medication:
Pre-Condition: User must be logged-in to the system.
1. *User adds a new medication to the patient's con-med list.*
2. *System will update the new medication to the database.*

Module	**EUC Model**	
	User Intentions	**System Responsibility**
Login	Identify self	
		Verify Identity
		Offer choice
View Patient Details	Select option	
		Display Information
Update Medication	Add option	
		Update Information

Test Requirements	Test Case Description	Test Data	Expected Output