A Systematic Review of Use Cases based Software Testing Techniques

Asad Masood Qazi¹*, Adnan Rauf² and Nasir Mehmood Minhas³

¹, ², ³University Institute of Information Technology – PMAS – Arid Agriculture University Rawalpindi, Pakistan

¹asadmasood1@hotmail.com, ²mindz.xpert@gmail.com, ³nasirminhas@uaar.edu.pk

Abstract

Use case based Software testing concerned with testing an entire system’s functionality and its constraints. User’s requirements can be drawn in form of use cases to show the internal and external behaviour of a system. The use case has been used to test software system for different levels as well as through many ways according to nature of software. Rather than proposing a new technique, it is necessary to synthesize the existing techniques of Use case based software testing which is available in the literature. A Systematic Literature Review has been performed to investigate all the current approaches of Use case based software testing techniques. A question-wise analysis has also been presented for the researchers to investigate the effectiveness of these techniques. In results, we have discussed all the existing approaches of Use case based software testing along with its strengths & weaknesses and measure the characteristics of these approaches based on some key parameters. We have concluded the current state of the art of all the existing approaches of Use case based testing. Results are based on both types of analysis; Qualitative as well as Quantitative.

Keywords: Software testing, systematic literature review, use cases based testing, test cases

1. Introduction

Software testing refers to executing a program with the intent of finding errors [1]. Producing testability in software is a complex task. Testing is also a time consuming and costly procedure. Software testing is mainly categorized into two types; white box and black box. White box testing refers to structural testing and black box testing refers to functional testing of any Software under Test (SUT) [2] [3]. Some researchers added grey box testing [3] in these types, which is an integration of white box and the black box or mixing of the specification and implementation or design testing. [4]

In this survey, our main focus is on use case based software testing. Use case models can test the system’s specification, design, architecture and behavior etc. In Literature, many researchers applied use case based testing to integrate with different UML diagrams to support the model based testing. A Model-based testing (MBT) is a type of testing that relies on explicit behavior models that encode the intended behaviors of an SUT and/or the behavior of its environment. Test cases are generated from one of these models or their combination and then executed on the SUT [5].

Use case and other modelling languages like UML are used to visualizing through graphical components of a system which helps the software engineer to develop software in an easy way. It reduces the complexities of requirement engineering process as well. Use case are used to transform the requirements specifications of any software to show its

* Corresponding Author
functionality and constraints. Through these Use case models, a specification based testing is also possible. Use case diagrams are used to show that how users will perform any task with their system or website. Use cases are used to show an internal and external behavior of a system. Also, it describes the relation between components as well as with sub-components by using "extend", "include" and "generalization" property. Use cases can be written in descriptive form as well as showed in graphical and visual form by showing actors and their association with a specific component of a system. Use case also helps to integrate product line or reuse features in a software development by improving Meta concepts of Use case. [7]

A major contribution of existing Use case based software testing approaches [6] - [11] deals with Object Oriented and component-based software domain. These techniques can be applied in System Oriented and component-based software domain. These techniques can be applied in System Testing [8], Black box testing [9], Requirements based testing [10], integration Testing [11], system testing GUI and testing of web based systems.

The rest of the paper is organized as follows: in Section 2, we present a Review of related surveys in which Use case, model based testing or other related work has been performed. Section 3 illustrates a complete Systematic Literature Review to gather thoroughly all the existing approaches of Use case based software testing techniques. Section 4 is the analysis of existing approaches according to criteria set in SLR. Finally, Section 5 concludes the research work and discuss some future work.

2. Related Work

The quality of software systems needs to improve, and for this purpose, testing must be performed. Different tools, mathematical techniques/formulas and models are available in the field of testing to ensure the overall quality of software product. Model-based or specification based testing mostly works on Use case and other UML (Unified Modelling Languages) diagrams like activity, collaboration, sequence etc.

Authors in [12] presented a survey related to model-based testing up to the current year 2015. This survey includes the current challenges and research gaps related to model-based testing in the field of computer science, software engineering, electrical engineering and other fields. This paper also focuses on the validation techniques of functional and non-functional testing techniques. Requirements gathering and specifications are important because testing depends upon the requirements. If requirements are gathered and specified correctly, testing will be easy, and gives correct results. This paper also covers the elements which are required for using a specific testing technique.

Another survey was conducted by Dr. Arvinder Kaur and Vidhi Vig given in [13]. The scope of that survey was based on testing through UML diagrams. Different authors’ uses a specific diagram like Use case, class diagram, collaboration diagram; sequence diagram etc. to generate auto test cases for software model based testing. In this survey the existing work is merged, and its research questions also answers the most popular technique (in terms of its usage in research field or in the industry) of model-based testing. Later on, it also discusses the type of testing used in each study.

Javier J. Gutierrez et. al., (2006) [14] presented existing techniques of Use case-based approaches and then proposed an approach to generate test cases from Use case for web applications. Because existing approaches are not delivering the feasible results; they expressed that Use case are more appropriate to start the generation process of test cases for web systems. This research followed case study approach for web applications and check that how a new link is added and stored into the system. The generation of test cases will invoke three different elements i.e., the test values, the interactions with the system under test and the expected results. The aim of his research is to validate that all the Use case have been successfully implemented in the web system under test.
Noraida Ismail et al., (2007) [15] in their study, highlighted some of the issues of model-based testing because these approaches were not fully automated. To tackle such issues author(s) has developed a tool for automatically generating test cases according to the system’s requirement. The researcher classified his process into two steps. First, the system requirements were transformed into Use case diagram and secondly the test cases will be generated according to the Use case respectively. These test cases are important to be used in analyzing and validating the requirements of the system as software testing is a key part of Software Development Life Cycle (SDLC) and delaying this process will definitely increase the number of errors and also it will be more tough and complex to fix these errors at the end. The researcher explained that once the use case diagram has been finalized from system requirements, it can be saved for further use and modification.

Clementine Nebut et. al., (2006) [16] emphasized on the barriers in model-based testing approaches using Use case and proposed an approach for generating concrete test cases by building a transition system from which the researcher create the test cases. This approach eliminated ambiguities encountered at the time of writing the requirements in a natural language simulation model of the use case. The simulation model is also used to explicitly build a model of all the valid sequences of use case, and from it to extract relevant paths using coverage criteria. These paths are called test objectives. The test objectives generation from the use case constitutes the first phase of their approach. The goal of this approach is to relieve of the automation work in test generation and to shift the effort to the specification activity. The aim of their research is to generate test cases for efficiently detecting faults in embedded software. The researcher performed an empirical study to validate his approach by using several case studies and deploy their approach for validating the results.

According to Anjali Sharma and Maninder Singh [17], the generation of test cases is an important activity in testing. For this purpose, authors defined an algorithm which takes UML diagrams as an input and generates test cases. The procedure of the proposed framework can work in any given UML diagram includes use case, class diagram, sequence diagram, data flow diagram etc. and convert them into a control flow graph (CFG). Then they apply test case generator and hit cost generator algorithm on it which will give an output in the form of test cases and coverage file. These test cases and coverage file is then use as an input to report generator tool and it gives a complete report to user or tester.

Shafique et. al., (2010) [18] presented a systematic literature review of tools supported by model-based testing technique. The scope of this systematic literature review is limited to identification of only tools and their features. Some of the tools from the findings of this literature review are Test Designer, Qtronic, MOTES, GOTCHA-TC Beans and others. The comparison criteria of these above-mentioned tools are States, Transitions, Transition Pair, Sneak Paths, Parallel Transitions and All scenarios.

A review of some of the model based testing approaches are given in [19] and [20], but the review is not systematic, so the comparative analysis is not been done systematic way. Secondly, there is no proper analysis approach of these MBT tools. Also, the study contains the review of approaches up to 2008, so to tackle the advancements later from 2008, there is a need to conduct a survey or systematic literature review.

In [21], the authors summarize the generation of test cases techniques for object oriented software in their review study. The study focuses on all three aspects related to generation of test cases which includes the generation, execution and evaluations of test cases. The scope of this study is gathering all possible techniques either UML based or not. This study also gives us the motivation to be a specific and in-depth analysis of doing any model based testing analysis related to a specific diagrammatic model.

It can also be observed from the study given in [22] a model is used to describe the behavior of the system, and to model the software systems, UML is one of the modeling languages. Test cases are generated through these UML based techniques, but how to
make them effective? This issue has been discussed in this review study. In this study, the scope of the test cases is very broad and focuses on the complete software development life cycle. This study found various types of testing through UML based approaches, includes goal oriented techniques [23], specification based techniques [24], sketch diagram based techniques [25] and source based techniques.

In [26], a recent work of literature review related to a model based testing by using UML state chart diagram is presented. A survey table presented in this study is well organized and presents complete details of the findings. The Table included the fields: Author / Technique Name, Input Model, Method, Intermediate model and coverage criteria. Most of the techniques used DFS algorithm. According to authors, the purpose of generating test cases through UML is to verify the relation between behavior, state, action and event. The correctness of test cases is totally based upon the correctness of the model. If the model is wrong and unreliable, the output will also be unreliable. Another survey related to software testing techniques was also presented in [27]. According to authors, software testing is not an easy task; it takes time as well as a proper sequence of steps. Which includes: planning, analysis, design, construction, testing cycle, final testing cycle and post implementation? Any software application which is under test can be tested through activity diagrams and other UML diagrams which help to reuse the components of a system. Model-based testing approach takes system under test as an input, generate system model, defining test coverage criteria, analyses using CPM (Category Partition Method) [28] and then generate test cases. In this survey paper, a review of work related to this particular approach is presented, and also complete UML diagrams are not covered.

In [29] a comparative evaluation of test cases generation through UML diagrams was presented. The comparative analysis showed that for unit level testing, state diagrams give better results and for integration level testing, sequence diagrams show better results in the generation of test cases. An experiment to evaluate the results followed their sequence of steps: Analyze the specific UML diagram of any particular system, Implements in Java, Generation of test cases, Design faults and run each test case of a system. It can be observed from another survey on automatic generation of test cases given in [30] that there were many techniques to generate the test cases automatically. Model-based testing is one of the techniques; and in model-based techniques, UML modeling is one of the popularly used approaches. These can be categorized in specification based test case generation [31], model based test case generation [32], path oriented test case generation [33] and intelligent techniques test case generation [34]. There is no main focus on any specific UML modeling diagram.

The above mentioned related works are presented to find the existing reviews and surveys related to use case based software testing. Some of the related works are not properly systematic, and some of them are too old to be considered. By presenting the related work, we set an objective of this study is to tackle all the approaches of use case based software testing in a systematic way.

3. Systematic Literature Review of Use cases based Testing

For the purpose of synthesizing the approaches described for Use case based software testing, the guidelines have been tailored from the systematic literature review proposed in [55]. In this regard, review protocol has been developed. The purpose of our study will be described in this protocol which may include the need of review protocol for our study, the definition of research questions, search process, inclusion / exclusion criteria, and primary study selection and finally analyzing the approaches used for Use case based software testing on the basis of different parameters. A detailed systematic literature review process of this paper is presented below in graphically in Figure 1.
3.1. Research Questions

The research questions to be addressed in this study are:

**RQ1:** What are the published Use case based software testing techniques used for test case generation?

**RQ2:** Which parameters are used to measure the characteristics of Use case based software testing techniques?

3.2. Search Process

In order to determine the similar work already performed in this area, we have to generate a search string. There are two ways to perform search strategy; one is manual search and the other is automatic search. Our search process is an automatic search of specific conference proceedings and journal papers. We had seen lots of papers published in different conferences, the nominated journals and conferences are as follows: Science Direct, Springer Link, IEEE Explorer, ACM and others.
In order to create the search string, the Boolean Operator “OR” was used to join alternative terms and synonym; and the Boolean Operator “AND” was used to join all the main terms.

We have identified some search terms and verify the keywords from major search terms of our Research Questions and by using the Boolean search expressions, we have constructed the following search string for our research questions:

(("test case generation from Use case diagrams" OR "approaches to generate test cases from Use case diagrams" OR "testing techniques to generate test cases from Use case diagrams" OR "testing approaches for test case generation using Use case diagrams") AND ("characteristics of Use case-based approaches" OR "characteristics of Use case-based techniques") AND ("systematic review" OR "systematic literature review" OR "systematic map" OR "systematic mapping" OR "mapping study" OR "evidence-based study" OR "mapping study").)

A trial search was conducted using this search string on Science Direct Digital Library and IEEE Transactions. The results of search terms proposed here were found sufficient after checking the above mentioned two electronic databases. So we concluded that Systematic Review is the best way to finding that what kind of similar work had been already performed in this area.

3.3. Results of Search String

The list of Journals / conference proceedings, keyword searched and number of papers found on different above mentioned sources. Approximately 384 papers are extracted from different sources when search string is triggered. Detail of the results of search string from different data sources is given in the following table, Table 1.

<table>
<thead>
<tr>
<th>Data Sources</th>
<th>Keywords Searched</th>
<th># of Papers Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Science Direct</td>
<td>&quot;test case generation from use case diagrams&quot; OR &quot;approaches to generate test cases from use case diagrams&quot;</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>&quot;testing techniques to generate test cases from use case diagrams&quot; OR &quot;MBT approaches for test case generation using use case diagrams&quot;</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>&quot;characteristics of use case-based approaches&quot; OR &quot;characteristics of use case-based techniques&quot;</td>
<td>14</td>
</tr>
<tr>
<td>Total - Science Direct</td>
<td></td>
<td>81</td>
</tr>
<tr>
<td>Springer Link</td>
<td>&quot;test case generation from use case diagrams&quot; OR &quot;approaches to generate test cases from use case diagrams&quot;</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>&quot;testing techniques to generate test cases from use case diagrams&quot; OR &quot;MBT approaches for test case generation using use case diagrams&quot;</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>&quot;characteristics of use case-based approaches&quot; OR</td>
<td>11</td>
</tr>
</tbody>
</table>
3.4. Study Selection Criteria

Initially, 384 papers were collected from results of our search string found in above mentioned electronic databases and some from other journals as well. These papers are analyzed on the basis of following inclusion-exclusion criteria:

3.4.1. Inclusion Criteria: Inclusion criteria for selection of studies is given below:

- Articles/papers which belong to use case-based software testing techniques.
- Articles/papers which belong to use cases based test cases generation.
- All published papers that have the potential of answering at least one research question.
3.4.2. Exclusion Criteria: Exclusion criteria for findings is:
- Topic who’s abstract and title not matching to our area.
- Multiple papers on the same topic will be excluded.
- Papers that do not have any link with our research questions.
- Duplicate papers (only the most complete, recent and improved one is included).
  The rest are excluded.

3.4.3. Selected Primary Study: The appropriately selected studies are enlisted by a single researcher. The rejected studies are checked by another researcher. We maintain a list of papers that were rejected with reasons for the rejection. Further, we executed the process of title and abstract screening of 384 searched papers in study selection process and discovered 45 relevant papers as a primary study.

4. Analysis on Findings

The purpose of this section is to analyses the literature found from our search process which will guide us to examine the approaches described for software testing in terms of UML Use case. The data collected from this stage will help us to answer our research questions. The summary of techniques found from our selected studies is given below:

4.1. Analysis on RQ 1

In this section, the analysis on existing techniques has been presented. To answer the RQ1, we have gone through a systematic review of Literature and identify the existing approaches of Use case based software testing and we enlist them in tabular form along with their strengths and limitations below in Table 2.

Table 2. Use Case based Software Testing Techniques along with their Strengths and Weaknesses

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Name of Author(s) and Year of Publication</th>
<th>Approach</th>
<th>Strengths</th>
<th>Limitations</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Shinpeiojata and Saeko Matsuura, (2010)</td>
<td>In this study, RA model is proposed which includes following steps for model-based testing: Interaction Activity Analysis, Concrete I/O Data analysis, UML Based scenario analysis, Scenario based prototype generation, Definition process of test case for Integration test stage</td>
<td>Requirements based testing, to meet the user’s requirement</td>
<td>No test coverage criteria is properly defined. Also the approach has no supporting tool available.</td>
<td>[37]</td>
</tr>
<tr>
<td>Sr. No</td>
<td>Name of Author(s) and Year of Publication</td>
<td>Approach</td>
<td>Strengths</td>
<td>Limitations</td>
<td>Reference</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------</td>
<td>----------</td>
<td>-----------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
</tbody>
</table>
| 3.     | Matthias et.al, (2003)                    | The proposed approach is based upon black box or specification based testing. The proposed approach consists 4 steps, which includes:  
* Domain understanding,  
* Model it in UML,  
* Convert it into Usage model and finally  
* Converted into Test model | Reliability rather than fault tolerance. And proper verification process through XML-based model transformation | Lengthy process of defining, refining and finalizing the Usecase models | [35] |
| 4.     | Basanieri et.al, (2000)                   | The proposed approach has following steps:  
* UML design analysis & search of relevant Usecase  
* Analysis of Sequence & class diagrams  
* Test Units  
* Research of setting & Interactions categories  
* Test specific construction  
* Search of message sequence & Test case  
* Usecase test suite & incremental construction of Test | Fast Analysis approach  
Success experiment in Industry level projects | Multiple Diagrams analysis involved | [38] |
<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Name of Author(s) and Year of Publication</th>
<th>Approach</th>
<th>Strengths</th>
<th>Limitations</th>
<th>Reference</th>
</tr>
</thead>
</table>
| 5.     | L Briand and Y Labiche (2002)            | Authors proposes TOTEM model, which includes following steps:  
- Analyse model to check completeness & correctness  
- Derive Usecase, class diagram & sequence diagrams  
- Derive variant sequences  
- Derive requirements for system testing  
- Derive test cases for system testing  
- Derive test oracles | Successfully applied in Case study / Experiment | Multiple UML diagrams are involved to analysis | [39] |
| 6.     | JJ Gutiérrez et.al, (2006)               | Authors proposes following sequence of steps to generate test cases from Usecase:  
- Generation of instances of Usecase  
- Generation of test values  
- Generation of test cases  
- Generation of expected results | Simple & applied in a case study | No proper objective is presented in study | [14] |
| 7.     | Li, Liping, and Huaikou Miao. (2008)    | This model is based upon two steps:  
- Convert simple Usecase model to hierarchical UCTMs (Usecase transition model)  
- Generate test cases by using CSC (Constraint Scenario Coverage) | Only focuses on Usecase | Abstract level technique. The approach needs simple and complete Usecase models. | [41] |
| 8.     | J Heumann. (2001)                       | Following steps to generate test cases are defined by Rational Rose Team:  
- Generate Scenario  
- Identify Test cases  
- Identify Data values to Test | Simple Approach and generate test cases by using only Usecase models | Small projects validation only | [42] |
<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Name of Author(s) and Year of Publication</th>
<th>Approach</th>
<th>Strengths</th>
<th>Limitations</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.</td>
<td>R Ibrahim et.al, (2007)</td>
<td>This paper presents an automatic tool for generating test cases using Use case. Sequence diagrams and flow of events used for checking consistency and validity of the generated test cases.</td>
<td>Automated Tool Support</td>
<td>No Operational Description</td>
<td>[43]</td>
</tr>
<tr>
<td>10.</td>
<td>L Chen et.al, (2010)</td>
<td>Authors propose IFA (Interaction Finite Automation) state based model to perform Use case based testing. The steps are: Transformation of Use case in to state models, Interaction Finite automation and transformation of algorithms, Test cases generation. The rules are: Each state in IFA should traverse at least once, Each transition in IFA should traverse at least once, Each path in IFA is traversed at least once.</td>
<td>Experiment / Case study validation</td>
<td>Complex Algorithm</td>
<td>[44]</td>
</tr>
<tr>
<td>11.</td>
<td>Nebut et.al, (2006)</td>
<td>A Global methodology of requirement based testing is applied in Thales Airbone System. Which generate test cases using following sequence of steps: Requirement</td>
<td>Validation through Case study Only use Use case models</td>
<td>Involve multiple tools Complex scenario and suite for complex and large projects</td>
<td>[16]</td>
</tr>
<tr>
<td>Sr. No</td>
<td>Name of Authors and Year of Publication</td>
<td>Approach</td>
<td>Strengths</td>
<td>Limitations</td>
<td>Reference</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------</td>
<td>----------</td>
<td>-----------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>12.</td>
<td>Bill Hasling et.al, (2008)</td>
<td>Gathering • Model the Usecase as Usecase scenarios • Convert Usecase into simulation model • Identify Test objectives • Generate test scenarios with the help of Usecase scenarios and test objectives • Finally, Set of test scenarios.</td>
<td>Validation through Experiment / Case study of Medical system. Improved Requirements and test cases</td>
<td>Extra Training is required for testers</td>
<td>[45]</td>
</tr>
<tr>
<td>13.</td>
<td>Serguei Roubtsov and Petra Heck. (2006)</td>
<td>Authors proposes a simple approach to generate test cases but by using multiple tools involvement through multiple tools. Ideally</td>
<td>Good for large scale projects to perform acceptance testing Small number of levels, to</td>
<td>Too much rely on tools, so extra training of tools may be required for testers. Heavily</td>
<td>[46]</td>
</tr>
<tr>
<td>Sr. No</td>
<td>Name of Author(s) and Year of Publication</td>
<td>Approach</td>
<td>Strengths</td>
<td>Limitations</td>
<td>Reference</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------</td>
<td>----------</td>
<td>-----------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>14.</td>
<td>Santosh Kumar et.al, (2010)</td>
<td>approach is based upon: • Test plan • Test design specification • Test case specification • Test Procedure</td>
<td>generate test cases Validation through real projects</td>
<td>documentation procedure as well.</td>
<td>[47]</td>
</tr>
<tr>
<td>15.</td>
<td>Bertolino et.al, (2003)</td>
<td>Authors include multiple UML models to generate the test cases. It is a complex approach which includes the transformation of UML diagrams into different intermediary models includes CFG, UDG, CCFG and others. The steps includes: • Deriving Usecase dependency sequences • Deriving test requirements from sequence diagrams • Deriving test cases.</td>
<td>Implemented in ComTestTool</td>
<td>Complex procedure and involve multiple UML diagrams to generate test cases for one case study.</td>
<td>[51]</td>
</tr>
<tr>
<td>16.</td>
<td>M. Sarma and R. Mall (2007)</td>
<td>The Proposed approach is based on the integration some existing tools which supports the UML based testing CDT and Cow_Suite. The proposed approach have following phases: • Test of the single virtual component • Test of a group of integrated virtual components</td>
<td>Automated Tools supported approach, which is easy to apply in different scenarios</td>
<td>Merger of UML diagrams and tools, so additional training may be required to testing team</td>
<td>[52]</td>
</tr>
<tr>
<td>Sr. No</td>
<td>Name of Author(s) and Year of Publication</td>
<td>Approach</td>
<td>Strengths</td>
<td>Limitations</td>
<td>Reference</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------</td>
<td>----------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
</tbody>
</table>
| 17.    | Chen et al, (2007)                      | Proposed approach is designed for generation of test cases for web based applications. Authors have proposed following steps to generate test cases:  
- Modelling of web application using page flow diagram  
- Definition of logical components and partitioning of web application.  
And to generate the test cases, authors proposes following steps:  
- Composition of logical components using intermediary diagrams like control flow graph etc.  
- Reuse of logical component interactions and supporting of concurrent access and interaction.  
A standard methodology, which can be applied in any web application. It is also a comprehensive approach of generation of test cases through Usecase and other UML diagrams. | approach of generation of test cases from UML diagrams. Also supports tools for verification and generation of test cases, and proposed approach follows proper algorithms. | Proposed approach has no tool support and not even validated through any case study or experiment | [53] |
| 18.    | Vieria et al.                           | In this paper, authors Proposed | Discussion | [54] |
Table 1: Use case supported testing methodology based on the use case study

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Name of Author(s) and Year of Publication</th>
<th>Approach</th>
<th>Strengths</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2006)</td>
<td>presented the importance of model-based testing techniques and GUI based system testing is performed by using Use case models of the system. The proposed methodologies includes following phases to generate test cases:</td>
<td>methodology is a simple methodology and can be easily implemented in various GUI based applications.</td>
<td>based evaluation of proposed methodology is presented in this study. Also the approach includes multiple modelling languages for a single program.</td>
<td></td>
</tr>
</tbody>
</table>

4.2. Discussion on RQ 1

Use case has been involved in different levels of software testing like system testing, requirements testing, integration testing and others. In this systematic review, a comparison of all findings is discussed and categorized the techniques according to their strengths and weaknesses. The strength of any technique depends upon its proposed steps, use of tool, an algorithm applied and also the level of testing.

Similarly the weaknesses of any Use case based software testing techniques refers to its complexity in terms of number of diagrams or models involved in each technique, like many techniques do not use directly Use case, also they includes other UML diagrams as well, so such techniques are complex in terms of time, cost as well as the training and accuracy required in the modelling techniques.

Model driven techniques are automated testing techniques, whereas model based techniques are not fully automated and the generation of test cases from Use case or other models are manual. Use case supported both types of testing model based as well as model driven testing. Other model based or model driven testing usually focuses on control flow graph, algorithms etc. but our findings show that there are many Use case based testing techniques which are simple, which generate test cases directly from the Use case without involving intermediary modeling language.

To analyses the RQ1 or the Use case based software testing, our results also focus on the acceptance level of any specific technique. Because the overall goal of any software testing technique is the acceptance of users, so in the strength of all the techniques, we try to find the level of acceptance of the results after testing the system through the use case.
4.3. Analysis on RQ 2

In this section, analysis on RQ2 has been presented. To answer the RQ2, we have gone through different literature repositories to identify the parameters, from which the existing use case based techniques has been characterized. We have also analyzed some of the most commonly used parameters which are presented and used by different literatures. The commonly used parameters are: Level of Testing, Algorithm used, use of tools or supported tools, Level of automation, software domain, complexity of techniques and others. We have presented analysis according to different parameters separately in tabular form as well as combined. A combined analysis is tabulated below in Table 3. Next sections will provide a complete Qualitative analysis of RQ2 by using graphical and tabular form to support the readers and to analyze the techniques in proper form.

Table 3. Analysis of Use Case based Software Testing Techniques w.r.t Different Parameters

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Name of Technique / Author</th>
<th>Algorithm</th>
<th>Level of Testing</th>
<th>Algorithm Supported</th>
<th>Tool Supported</th>
<th>Coverage Criteria</th>
<th>Validation Through</th>
<th>Software Domain</th>
<th>Level of Testing</th>
<th>Technique</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shinpei ogata and Saeko Matsuura, (2010)</td>
<td>Yes</td>
<td>Not Defined</td>
<td>Not Defined</td>
<td>Experiment / Case Study</td>
<td>Business Domain</td>
<td>Black Box/ Specification Based</td>
<td>[37]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Mourad Badri et.al, (2004)</td>
<td>Yes</td>
<td>Not Defined</td>
<td>Yes</td>
<td>Experiment / Case Study</td>
<td>Object Oriented</td>
<td>System Testing</td>
<td>[36]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Matthias et.al, (2003)</td>
<td>Not Defined</td>
<td>Yes</td>
<td>Yes</td>
<td>Experiment / Case Study</td>
<td>Object Oriented</td>
<td>System Testing</td>
<td>[35]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Basanieri et.al, (2000)</td>
<td>Not Defined</td>
<td>Not Defined</td>
<td>Yes</td>
<td>Experiment / Case Study</td>
<td>Object Oriented</td>
<td>Integration &amp; Architecture Testing</td>
<td>[38]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>L Briand and Y Labiche (2002)</td>
<td>Yes</td>
<td>Yes</td>
<td>Not Defined</td>
<td>Experiment / Case Study</td>
<td>Object Oriented</td>
<td>System &amp; Regression Testing</td>
<td>[39]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>JJ Gutiérrez et.al, (2006)</td>
<td>Not Defined</td>
<td>Not Defined</td>
<td>Yes</td>
<td>Experiment / Case Study</td>
<td>Object Oriented</td>
<td>Requirement Testing</td>
<td>[40]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Li, Liping, and Huaikou Miao. (2008)</td>
<td>Yes</td>
<td>Not Defined</td>
<td>Yes</td>
<td>Experiment / Case Study</td>
<td>Object Oriented</td>
<td>System Testing</td>
<td>[41]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>J Heumann. (2001)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Experiment / Case Study</td>
<td>Object Oriented</td>
<td>Requirement Testing</td>
<td>[42]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>L Chen et.al,</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Experiment /</td>
<td>Object Oriented</td>
<td>System Testing</td>
<td>[44]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sr. No</td>
<td>Name of Author, Year</td>
<td>Technique Supported</td>
<td>Algorithm Supported</td>
<td>Tool Supported</td>
<td>Coverage Criteria Supported</td>
<td>Validation Through</td>
<td>Software Domain</td>
<td>Software Testing Technique Supported</td>
<td>Level of Testing Supported</td>
<td>Reference(s)</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>----------------------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>----------------</td>
<td>-----------------------------</td>
<td>------------------</td>
<td>----------------</td>
<td>----------------------------------</td>
<td>-------------------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Nebut et.al, (2006)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Experiment / Case Study</td>
<td>Object Oriented</td>
<td>System &amp; Requirements Testing</td>
<td>[16]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Bill Hasling et.al, (2008)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Experiment / Case Study</td>
<td>Object Oriented</td>
<td>System Testing</td>
<td>[45]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Serguei Roubtsov &amp; Petra Heck, (2006)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Experiment / Case Study</td>
<td>Object Oriented</td>
<td>System &amp; Users Acceptance Testing</td>
<td>[46]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Santosh Kumar et.al, (2010)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Experiment / Case Study</td>
<td>Object Oriented</td>
<td>Integration &amp; System Testing</td>
<td>[47]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Bertolino et.al, (2003)</td>
<td>Not Defined</td>
<td>Yes</td>
<td>Not Defined</td>
<td>Experiment / Case Study</td>
<td>Component Based Software</td>
<td>Integration Testing</td>
<td>[51]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>M. Sarma and R. Mall (2007)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Automated Tool / Case Study</td>
<td>Business Domain</td>
<td>Black Box / System Testing</td>
<td>[52]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Chen et.al, (2007)</td>
<td>Yes</td>
<td>Not Defined</td>
<td>Yes</td>
<td>Not Defined</td>
<td>Business Domain</td>
<td>Requirements Testing</td>
<td>[53]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**4.3.1. Level of Testing:** Level of testing [48] refers to the behavior of system model, which is to be examining under the desired testing technique. An example of levels is Integration, Requirements and Specification etc. We have analyzed each technique in depth to analyze the discussed level of testing in all the techniques and if there is any tool supported by that particular technique we have enlisted them as well. The details of findings are presented in a diagram mentioned below named as Figure 2 and also enlisted in tabular form below in Table 4. The analysis on the basis of number of approaches found in each level as well as the number of approaches supported by a tool also presented below in Figure 3 and Figure 4 respectively.
Figure 2. Analyzing the Findings on basis of Level of Testing Performed by Use Cases Based Techniques

Table 4. Analyzing the Findings on Basis of Level of Testing

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Level of Testing</th>
<th>No. of Approaches</th>
<th>Supporting Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>System Testing</td>
<td>09</td>
<td>06</td>
</tr>
<tr>
<td>2.</td>
<td>Integration Testing</td>
<td>03</td>
<td>02</td>
</tr>
<tr>
<td>3.</td>
<td>Specification Based Testing</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>4.</td>
<td>Regression Testing</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>5.</td>
<td>Requirements based Testing</td>
<td>05</td>
<td>03</td>
</tr>
</tbody>
</table>

Not Defined
Figure 3. Analysis on the Basis of Level of Testing of Total Number of Use Case Based Testing Approaches Found

Figure 4. Analysis on the Basis of Use of Tools of Total Number of Use Case Based Testing Approaches Found
4.3.2. **Algorithm Usage:** Algorithms are used by each technique to traverse the graphs and trees if used by any technique. Most of the cases DFS [49] and BFS [50] algorithms are used by model – based testing techniques.

4.3.3. **Tools Support:** The support of tool is used to automate the testing process. Most of the model – based testing approaches uses model checking tools and algorithms. Some of the techniques are founded from literature, in which researcher proposes a new tool [43] and some of the authors / researchers uses existing tools [47] to generate test cases from use case models.

4.4. **Discussion on RQ2**

As discussed earlier, the findings of the use case based software testing techniques involve many levels of testing *i.e.*, system testing, integration testing, regression testing, requirements based testing, acceptance testing and specification based software testing. The main objective of this study is to find a specific technique which can be applied to test the multiple levels of testing, which is not a complex issue and may reduce the cost and other resources usage. The researchers implemented these techniques successfully and evaluated in some case studies as well as experiments on a high level and complex software development solutions.

To evaluate these issues, our analysis approach is based on maximum possible parameters which can be set by any software tester, researchers and other software engineers from academics. Use of tool and implementation of Algorithm are the main attributes to make the technique simple or complex. Complex algorithms decrease the level of automation of software testing technique, and also requires more support or training to its implementation.

5. **Conclusion and Future Work**

In this article, we have performed a systematic literature review to find the existing approaches to model-based testing through UML Use case diagrams. In our study, we have found different approaches to performing some specific type of testing, which can be performed in any specific or generic phase of software development life cycle.

UML Use case can be used to perform requirements based testing, which ensures that user's requirements are completely fulfilled. And it is performed in a Requirement analysis phase. Similarly, in High-level design or software architecture phase of SDLC, system testing is performed through a use case. And in low-level design, integration testing is performed through Use case. These all techniques and types of test cases full fill the requirements of black box testing in some way. Regression testing, which can be performed in any phase or stage, can also be done through the Use case diagrams. To validate all the requirements and acceptance of software product, acceptance testing is performed. If all the results from previously mentioned testing types are being corrected, then acceptance testing can be performed. For this purpose, Use case can also be helpful for the software engineers and practitioners.

The purpose of this study is to identify and synthesize the current state of the art of existing approaches of Use case based testing techniques. In future, we will work on improvements on different testing techniques performed through requirements specifications.

**References**

[29] S. Kansomkeat, J. Offutt, A. Abdurazik and A. Baldini, “A comparative evaluation of tests generated from different UML diagrams”, In Software Engineering, Artificial Intelligence, Networking, and


Authors

Asad Masood Qazi, he is a young IT Professional with an interest in software design and development, database management systems, OLAP systems, data warehousing, software engineering, software process improvement and software quality assurance. Mr. Asad Masood is also a research student at University Institute of Information Technology – PMAS – Arid Agriculture University Rawalpindi, Pakistan, and also participated in different research oriented projects of different areas related to software engineering. Some of the research publications of Mr. Asad Masood could be found in different International Journals.

Adnan Rauf, he is a Research student at University Institute of Information Technology – PMAS – Arid Agriculture University Rawalpindi, Pakistan, and also participated in different research oriented projects of different areas related to software engineering. The area of interest of Mr. Adnan Rauf in research is Software Testing, Ontologies, Knowledge Management and GSD related issues in software engineering.

Nasir Mehmood Minhas, he is working as Assistant Professor at University Institute of Information Technology – PMAS – Arid Agriculture University Rawalpindi, Pakistan. The major area of research is Software Requirement Engineering, Software Process / Software Process Improvement, Global / Distributed Software Development and Ontologies in software development practices.