Approaches for Test Case Generation from UML Diagrams

TINA SCHWEIGHOFER and MARJAN HERIČKO, University of Maribor

Model based testing (MBT) is an important approach with many advantages that can reduce the cost and increase the effectiveness and quality of a testing procedure. In MBT, test cases can be derived from different models, also from the popularly used UML diagrams. Different UML diagrams include various important pieces of information that can be successfully used in a testing procedure. A lot of papers present approaches for test case generation from different UML diagrams and researchers are trying to find the most optimal one. In this paper, we present the first results of a systematic literature review in the area of test case generation from UML diagrams. Based on research questions, we explored which UML diagrams are most commonly used for test case generation, what approaches are presented in the literature, their pros and cons and connections with different testing levels. We also tried to find approaches that are tailored to test mobile applications. First results show that UML state machine, activity, sequence diagram and of course, a combination of more UML diagrams, are most commonly used for test case generation. Different approaches are used for generation, like graphs, trees, tables, labelled transition systems (LTS), genetics algorithms (GA), finite state machines (FSM) and others. The found approaches have many advantages, but also some disadvantages such as a lack of automatization, problems with complex diagrams and others. A detailed analysis is presented in this article.

General Terms: Test case generation from UML

Additional Key Words and Phrases: test case generation, UML, approaches, MBT, state machine diagram, activity diagram, sequence diagram, SLR

1. INTRODUCTION

Testing is an important part of the software development process in which we want to check if a product satisfies the given requirements. It is a non-trivial process with many important parts. As products become increasingly more complex, the process has become very extensive and time consuming. Consequently, a logical conclusion would be the automatization of the process. Not only for the execution of test cases, but also the preparation process, which includes test case design and generation. The topic of test case generation is becoming more and more popular and because test case design and execution are time and resource consuming, it is understandable that automatic test case generation constitutes an important topic [Samuel et al. 2008].

Test cases can be generated from code, graphs, formal specifications and different models. Testing from models, also known as model based testing (MBT), is currently a popular research topic. MBT is a testing methodology that usually facilitate the automation of a test case generation using either models or properties, as the basis for deriving complete test suites [Francisco and Castro 2012].

One of the most popular models used for test case generation are UML diagrams. The Unified Modeling Language (UML) is used for modeling and presents different views of the system. Some of the diagrams are very popular and often used for modeling; the same can be observed in the process of test case generation. UML models constitute an important source of information for test case design. Therefore, UML based automatic test case generation is an important but also theoretically challenging topic that has been getting a lot of attention from researchers.

The generation of test cases from UML diagrams is also a research topic of our paper. We want to explore which UML diagrams are most commonly used for the test case generation process, how widespread the approaches are and which techniques are used. We used the research method systematic literature review (SLR) and followed good practices and recommendations. Detailed research questions were formed and a detailed analysis of the approaches were made with an emphasis on their pros, cons,

Author's address: T. Schweighofer, Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia; email: tina.schweighofer@um.si; M. Heričko, Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia; email: marjan.hericko@um.si.

In: Z. Budimac, T. Galinac Grbac (eds.): Proceedings of the 3rd Workshop on Software Quality, Analysis, Monitoring, Improvement, and Applications (SQAMIA), Lovran, Croatia, 19.-22.9.2014, published at http://ceur-ws.org

and opportunities for improvement. The content of the work is organized as follows: After the introduction, some theoretic background is presented; including test case generation, MBT and UML, followed by the research part of the work. The research method is described and the results are presented in an organized form. At the end, results are discussed and summarized and the paper is concluded.

2. BACKGROUND

2.1 Test Case Generation

The testing effort is divided into three parts: test case generation, test execution and test evaluation. In comparison with the other two parts, test case generation is the most challenging [Gulia and Chillar 2012; Mingsong et al. 2006]. Test cases that are created manually are usually error prone and time consuming, so the automation of test case specifications is the next logical phase [Schwarzl and Peischl 2010]. Test case generation can save time and effort and at the same time reduce the number of errors and faults [Gulia and Chillar 2012]. It also cuts down on the costs of manual testing and increases the reliability of tests [Shamsoddin-Motlagh 2012]. The reason lies in test cases that can be generated from models in parallel with the implementation of system, which can then be easily updated if the specifications change [Dalal et al. 1999; Wang et al. 2008].

2.2 Model Based Testing

Model based testing (MBT) is a promising approach for software quality control and for reducing the costs of a test process, because test cases can be generated from the software specifications at the same time as development [Cartaxo et al. 2007; Zeng et al. 2009]. MBT can be described with the following action. First, the model is built from software requirements, and the expected inputs and outputs are generated from a formal model. Tests are then run and inputs and outputs are generated. Finally, those outputs are compared to the expected outputs [Cartaxo et al. 2007].

MBT generally creates tests from an abstract model of the software, including formal specifications and semi-formal design descriptions such as UML diagrams [Kansomkeat et al. 2008]. It relies on behaviour models based on input and expected output for test case generation in its implementation [Pretschner et al. 2005]. MBT has evolved out of techniques like finite state machine (FSM), labelled transition systems (LTS), Message Sequence Charts (MSC) and Petri nets to generate test cases for systems [Shirole and Kumar 2013]. One of the most used and popular models for MBT are also UML diagrams [Swain et al. 2010].

2.3 Unified Modeling Language

In our research, we focused on the three UML diagrams that are most commonly used for test case generation: the UML state machine diagram, the UML activity diagram and the UML sequence diagram. UML state machine diagrams are part of behaviour diagrams and are well suited for describing the behaviour of a system [OMG 2011; Schwarzl and Peischl 2010]. It models dynamic behaviour and captures different states that an object can be in and its response to various events that may arise in each of its states [OMG 2011; Samuel et al. 2008]. Because there are formalized aspects of UML, a state diagram can provide a natural basis for test data generation [Shirole et al. 2011]. An activity diagram is a UML diagram that provides a view of the behaviour of a system by describing the sequence of actions in a process [Fan et al. 2009]. An activity diagram is used to depict all possible flows of execution in a use case [Nayak and Samanta 2011]. A sequence of activities in an activity diagram of a use case can be used to generate test cases. It is required to identify all possible begin-to-end paths in an activity diagram in order to cover all the activities and flow of constructs to test a use case satisfactorily [Nayak and Samanta 2011]. The sequence diagram is known as an interaction diagram that represents a scenario as a possible sequence of messages that are exchanged among the object [Khandai et al. 2011].

3. RESEARCH METHOD

Based on a set problem and desired results, we decided to use the research method systematic literature review (SLR). SLR is a means of identifying, evaluating and interpreting all available research relevant to a particular research question or topic area or phenomenon of interest. Individual studies contributing to a SLR are called primary studies and a systematic review is a form of secondary study [Kitchenham and Charters 2007].

3.1 Research Questions

Specifying research questions is the most important part of any systematic review. The review questions drive the entire systematic review methodology [Kitchenham and Charters 2007]. In our research we form 3 mayor research questions that are presented below.

***RQ1**: Are there any systematic literature reviews or mapping studies on the topic of test case generation from UML diagrams?*

> ***RQ1.1**: Are there any systematic literature reviews or mapping studies on the topic of test case generation?*

***RQ2**: Is there any research regarding test case generation or model based testing from UML in the field of mobile applications?*

***RQ3**: Are there any studies regarding test case generation from different UML models?*

> ***RQ3.1**: Which types of UML diagrams are used for test case generation?*
> ***RQ3.2**: What approaches of test case generation are used?*
> ***RQ3.3**: What are the pros and cons of the used approaches?*
> ***RQ3.4**: What are the connections between UML diagram type and testing approach?*

3.2 Data sources and Search Strings

To properly perform a literature review, we chose the appropriate data sources. We decided that we would use different Digital Libraries. Some of the available electronic bases were then searched for primary studies. Table I shows the Digital Libraries that were used.

A general approach to define search strings is to break down research questions into individual terms [Kitchenham and Charters 2007]. In our case, terms were then combined with the logical operator "AND" in order to link together different terms. Different search strings defined from the defined research questions are shown in Table II with links to the appropriate research question.

Table I. Data Sources for SLR

ELECTRONIC BASE	URL
IEEE Xplore	http://ieeexplore.ieee.org
ScienceDirect	http://www.sciencedirect.com
SpringerLink	http://link.springer.com
ACM Digital Library	http://dl.acm.org
Scopus	http://www.scopus.com
ProQuest	http://search.proquest.com
EBSCO DiscoveryService	http://eds.a.ebscohost.com/

Table II. Defined Search Strings

RESEARCH QUESTION	SEARCH STRING
RQ1 and RQ 1.1	"test case" AND "generation" AND "review" AND "systematic" AND "UML" "test case" AND "generation" AND "mapping" AND "systematic" AND "UML" "test case" AND "generation" "test case generation" "test case" AND "generation" AND "review" AND "systematic" "test case" AND "generation" AND "mapping" AND "systematic"
RQ2	"test case" AND "generation" AND "mobile" "model based testing" AND "mobile"
RQ3 and RQ 3.1-3.4	"test case" AND "generation" AND "UML" "model based testing" AND "UML"

3.3 Search Results and Study selection

First, a search was conducted using the search string. A different number of results were found for different search strings. When there were too many results, we limited our search on just the abstracts, title and keywords. The results were reviewed and potentially relevant primary studies were gathered. After this, studies were assessed for their actual relevance with the aim of a study selection to identify those primary studies that provide direct evidence about research questions [Kitchenham and Charters 2007]. We reviewed the selected studies and appropriate ones were selected for further reading. In order to determine whether or not the study was selected, the title and abstract were evaluated regarding inclusion and exclusion criteria. Selection criteria were set in order to reduce the likelihood of bias. They were based on research questions [Kitchenham and Charters 2007]. Exclusion criteria were that the paper was not available in selected electronic databases, the paper is not in English and paper that does not describe the test case generation process. On the other hand, inclusion criteria was that the focus is on generating test cases and that the paper deals with test case generation from the UML model. After initial studies were selected, they were reviewed in detail and the whole article was examined. In the second round, 67 studies were selected for further examination.

3.4 Data Extraction and Data Synthesis

After the study selection, we performed data extraction from the selected primary studies. Data was collected using a form that was designed and reviewed to collect all the information needed to address the review question. The data fields that we collected are presented in Table III.

Data synthesis involves collating and summarising the results of the included primary studies [Kitchenham and Charters 2007]. 67 primary studies were selected for further analysis. We decided that we would continue the analysis separately depending on the type of UML diagram used for test case generation. We decided on the following categories, named after UML diagram types: UML state diagram, UML activity diagram, UML sequence diagram and different used UML diagrams. We are aware that there are many different types of UML diagrams, but those that we selected are used in a significant number of articles. Some others, for example the collaboration or class diagram, were rarely used as only a diagram for test case generation. This is why only the previously mentioned UML diagram types were examined in detail and, of course, papers with a combination of different UML diagrams.

Table IV gives the exact number of selected primary studies divided by types of UML diagrams.

Table III. Data Extraction Form

DATA TYPE	MEANING
Title	Title of the paper.
Authors	Authors of the paper.
Year	Year the paper was published.
UML diagram type	UML diagram used for test case generation.
Testing level	Testing level covered by the approach.
Approach	Describing the proposed approach for test case generation.
Complementary techniques	Used the complementary techniques in the approach.
Practical Example	Simple example, prototype or implementation of approach.
Purpose	Purpose of the approach.
Pros	Approach pros.
Cons	Approach cons.
Future work	Future work described in the article.
Notes	Notes taken by the reviewer.

Table IV. Number of Selected Primary Studies by Types of UML diagram

	STATE MACHINE DIAGRAM	ACTIVITY DIAGRAM	SEQUENCE DIAGRAM	COMBINED DIAGRAMS
Number of studies	22	19	7	19

4. RESULTS

Much of the literature in the field of test case generation is available from different UML diagrams. The literature presents different techniques and approaches, where each has its own advantages and disadvantages. Many examples are found in the literature that describe test case generation from a single UML model and most commonly used are the diagram techniques that we choose and are presented in Table IV. We made an analysis based on data that was extracted and the results show the basis for our answers to specific research questions.

4.1 RQ1

In the first research question, we were looking for systematic studies regarding test case generation from UML diagrams. We found research [Shamsoddin-Motlagh 2012] that addressed automatic test case generation and presented approaches based on UML, graphs, formal methods, web applications and web services. They list some of the techniques but we focused mostly on UML based approaches. Some papers were also found that cover only the area of test case generation from UML diagrams. The most broad one was presented by the authors [Kaur and Vig 2012]. They present a systematic survey of the work done in the field of the automatic generation of test cases. Many techniques proposed for test case generation were found based on different UML techniques. The article [Shirole and Kumar 2013] presents a survey which aims to improve the understanding of UML behavioural based techniques. They present approaches for test case generation based on UML Sequence, State Chart and Activity diagram. We also came across an article [Aggarwal and Sabharwal 2012] that presents only approaches for test case generation from UML State Machine diagrams. Different techniques were presented and divided into groups, based on different methods usage.

4.2 RQ2

The second question was aimed at finding any available research regarding test case generation in the area of mobile applications. We found one piece of research [Chouhan et al. 2012] that directly addresses test case generation from UML activity diagram for mobile applications. The diagram was converted into a table and then into a graph by an algorithm. The test cases are then generated. In our view, the approach does not cover any special properties of mobile applications in the test case generation process. It only addresses the problem of many pages linked with each other and some predefined sequences of occurrences. Another approach is presented by [Cartaxo et al. 2007]. They present a systematic procedure

of functional test case generation by mobile phone applications. They are focused on testing features, which is an increment of functionality and they call mobile applications features. They propose an example of a feature, like Message, that has "send" and "receive" functionalities. The work is part of the research for Motorola mobile phone applications. Their approach is tailored for testing mobile applications or features, whose requirements are specified by sequence diagrams. But, the features (what they call mobile applications) are not the mobile applications that we know in a modern context.

4.3 RQ3

The next RQ is aimed at researching different approaches for test case generation from UML diagrams. We tried to find studies in this area, revealed which UML models are used with which techniques and what are pros and cons of the approaches.

We can surely confirm this, because in the process of searching for appropriate studies we found a lot of results. All of them were not appropriate, so a few of them were eliminated. In the end we chose 67 appropriate studies. The distribution of articles according to different techniques have already been presented in Table IV. An analysis of publications per year by different UML diagrams is presented in Figure 1.

We concluded that the most common diagrams used for test case generation are: the UML state machine diagram, UML activity diagram, UML sequence diagram and a combination of others diagrams like UML class, object and use case diagrams. The same conclusion was also made in the article [Kaur and Vig 2012]. They found that the most widely used ones were a combination of different UML techniques, then UML state, activity, and sequence diagram. UML class, object and use case diagrams are not precise enough for MBT according to [Shirole and Kumar 2013; Utting and Legeard 2007]. Hence, an additional description from the dynamic behaviour model was needed [Shirole and Kumar 2013; Utting and Legeard 2007]. That is why these types of diagrams are often used in test case generation examples as the complementing diagram technique.

We can conclude that regardless of the type of UML diagram, there are some joint techniques used for test case generation, like graphs, diagrams, trees and tables, labelled transition systems, genetic algorithms, OCL, XML and XMI. One widespread technique is also the finite state machine (FSM), but it can be found only in the area of the UML state machine diagram. This is not surprising, because FSM provides basic mathematical concepts for the UML state machine diagram.

Analysed approaches have a number of disadvantages. They are the same ones that plague the UML technique in general. In every analysed group, we found the same problems, like a large number of test cases and test sequences to achieve a good result, while only simple diagrams can be used, where all elements are not supported and specific issues like polymorph, concurrency and infinity loop are not addressed. Also, all the approaches are not automated, some steps have to be done manually and information from some types of diagrams are not enough. Some approaches are expensive and others are in the early stages of maturity and not tested enough. Some are presented only as a theoretical approach with some case studies and examples. We also see that when tools are presented, the test case generation process usually is not well represented.

However, there are also some advantages for the approaches. In general, all of them are trying to solve problems regarding manual testing and long and expensive test case generation procedures. They try to solve problems regarding test case coverage problems and reduce human errors. In addition, some approaches try to improve previously presented approaches. Approaches can process a large amount of data, so that a larger system can also be tested efficiently and controlled.

Despite that, we found approaches that generate test cases for almost all levels and types of testing. There are some findings about which UML diagram is most suitable for each type of testing.

If we start with the UML state machine diagram, we found approaches appropriate for unit testing, system level testing, conformance, and functional testing. According to the results, we can conclude that the majority of approaches are meant for unit level testing. This is also confirmed in different studies [Khandai et al. 2011; Kansomkeat et al. 2008], where they found out that UML state machine diagrams are the most suitable for deriving test cases for unit testing. The second diagram we analysed is UML

activity diagram. Studies revealed that the test cases generated can be used for system level testing and some of them also for functional testing. The last diagram technique is the UML sequence diagram. Test cases can be used for different testing levels and types, like unit, system testing and functional and conformance testing. But, as shown in the research [Kansomkeat et al. 2008] UML Sequence diagrams are very useful for integration level testing.

In approaches that combine different UML diagram techniques, we primarily found approaches for system level testing those that represent implemented tools. But, there are also approaches that generate test cases for unit and integration level testing.

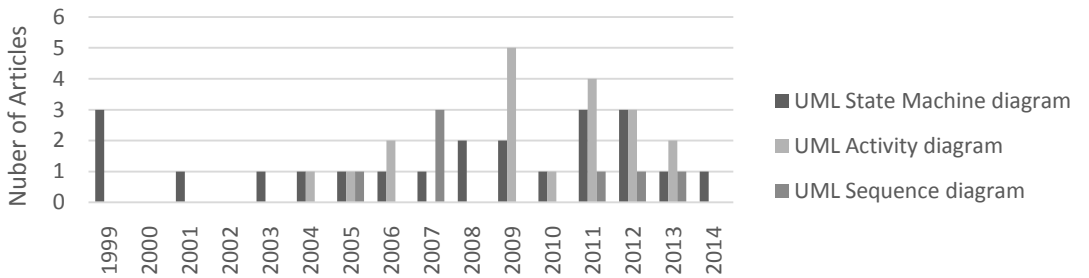

Fig 1. Publications per year divided by different UML diagrams

5. DISCUSSION

Test case generation from different UML diagrams constitutes the main research topic of the presented work. The scope of the research is broadly explored and presented in many articles and studies available from different sources. Our goal was to investigate the field and answer the proposed research questions. Using the research method SLR, we developed a research protocol. After we selected primary studies appropriate for further research, we performed an analysis.

We can conclude that the most commonly used UML diagrams for test case generation are UML state machine, activity and sequence diagrams and the class and object diagram in combination with other UML diagrams. The found approaches generate test cases with the help of some additional methods, most commonly graphs, trees, tables, labelled transition systems (LTS), genetics algorithms (GA) and finite state machines (FSM). We can also conclude that UML State Machine diagrams are most suitable for the unit testing level and UML sequence diagram for integration testing level. However, this is not the only option; we also found approaches that produce test cases for different testing levels and types. On the other hand, the UML activity diagram and approaches with more than one UML diagram are usually generating test cases for system level testing. Especially those approaches that also present an implemented testing tool. We can also conclude that not all UML diagrams are appropriate for all testing levels and types.

Despite the fact that the field of test case generation from UML models is investigated in detail, there are still some possibilities for further work. We would like to continue our work and we plan to proceed and make a more detailed analysis of SLR. We also want to implement the most promising approaches and compare them according to different properties, like effort used for test case generation, the need to supplement the model, the test case coverage of a model, the number of found mistakes, etc. Adjusted and optimized approaches could be used for testing mobile applications while taking into account different mobile characteristics and assessing and evaluating the quality of generated test cases based on different criteria, the number of found mistakes, and coverage, such as bottleneck detection and code duplication detection.

REFERENCES

AGGARWAL, M. AND SABHARWAL, S., 2012. Test Case Generation from UML State Machine Diagram: A Survey. *Computer and Communication Technology (ICCCT), 2012 Third International Conference on*, pp.133–140.

CARTAXO, E.G., NETO, F.G.O. AND MACHADO, P.D.L., 2007. Test case generation by means of UML sequence diagrams and labeled transition systems. *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pp.1292–1297.

CHOUHAN, C., SHRIVASTAVA, V. AND SODHI, P.S., 2012. Test Case Generation based on Activity Diagram for Mobile Application. *International Journal of Computer Applications*, 57(23), pp.4–9.

DALAL, S.R. ET AL., 1999. Model-based testing in practice. *Software Engineering, 1999. Proceedings of the 1999 International Conference on*, pp.285–294.

FAN, X., SHU, J., LIU, L. AND LIANG, Q.J., 2009. Test Case Generation from UML Subactivity and Activity Diagram. *Electronic Commerce and Security, 2009. ISECS '09. Second International Symposium on*, 2, pp.244–248.

FRANCISCO, M.A. AND CASTRO, L.M., 2012. Automatic Generation of Test Models and Properties from UML Models with OCL Constraints. In *Proceedings of the 12th Workshop on OCL and Textual Modelling*. OCL '12. New York, NY, USA: ACM, pp. 49–54.

GULIA, P. AND CHILLAR, R.S., 2012. A New Approach to Generate and Optimize Test Cases for UML State Diagram Using Genetic Algorithm. *SIGSOFT Softw. Eng. Notes*, 37(3), pp.1–5.

KANSOMKEAT, S., OFFUTT, J., ABDURAZIK, A. AND BALDINI, A., 2008. A Comparative Evaluation of Tests Generated from Different UML Diagrams. *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08. Ninth ACIS International Conference on*, pp.867–872.

KAUR, A. AND VIG, V., 2012. Systematic Review of Automatic Test Case Generation by UML Diagrams. *International Journal of Engineering Research & Technology*, 1(6).

KHANDAI, M., ACHARYA, A.A. AND MOHAPATRA, D.P., 2011. A novel approach of test case generation for concurrent systems using UML Sequence Diagram. *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, 1, pp.157–161.

KITCHENHAM, B. AND CHARTERS, S., 2007. *Guidelines for performing Systematic Literature Reviews in Software Engineering*,

MINGSONG, C., XIAOKANG, Q. AND XUANDONG, L., 2006. Automatic test case generation for UML activity diagrams. In *Proceedings of the 2006 international workshop on Automation of software test*. AST '06. New York, NY, USA: ACM, pp. 2–8.

NAYAK, A. AND SAMANTA, D., 2011. Synthesis of test scenarios using UML activity diagrams. *Software & Systems Modeling*, 10(1), pp.63–89.

OMG, 2011. *Unified Modeling Language Version 2.4.1*,

PRETSCHNER, A. ET AL., 2005. One Evaluation of Model-based Testing and Its Automation. In *Proceedings of the 27th International Conference on Software Engineering*. ICSE '05. New York, NY, USA: ACM, pp. 392–401.

SAMUEL, P., MALL, R. AND BOTHRA, A.K., 2008. Automatic test case generation using unified modeling language (UML) state diagrams. *Software, IET*, 2(2), pp.79–93.

SCHWARZL, C. AND PEISCHL, B., 2010. Test Sequence Generation from Communicating UML State Charts: An Industrial Application of Symbolic Transition Systems. *Quality Software (QSIC), 2010 10th International Conference on*, pp.122–131.

SHAMSODDIN-MOTLAGH, E., 2012. A Review of Automatic Test Cases Generation. *International Journal of Computer Applications*, 57(13), pp.25–29.

SHIROLE, M. AND KUMAR, R., 2013. UML Behavioral Model Based Test Case Generation: A Survey. *SIGSOFT Softw. Eng. Notes*, 38(4), pp.1–13.

SHIROLE, M., SUTHAR, A. AND KUMAR, R., 2011. Generation of Improved Test Cases from UML State Diagram Using Genetic Algorithm. In *Proceedings of the 4th India Software Engineering Conference*. ISEC '11. New York, NY, USA: ACM, pp. 125–134.

SWAIN, S.K., PANI, S.K. AND MOHAPATRA, D.P., 2010. Model Based Object-Oriented Software Testing. *Journal of Theoretical & Applied Information Technology*, 14(1/2), p.30.

UTTING, M. AND LEGEARD, B., 2007. *Practical Model-Based Testing: A Tools Approach*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

WANG, X., GUO, L. AND MIAO, H., 2008. An Approach to Transforming UML Model to FSM Model for Automatic Testing. *Computer Science and Software Engineering, 2008 International Conference on*, 2, pp.251–254.

ZENG, F., CHEN, Z., CAO, Q. AND MAO, L., 2009. Research on Method of Object-Oriented Test Cases Generation Based on UML and LTS. *Information Science and Engineering (ICISE), 2009 1st International Conference on*, pp.5055–5058.