JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
31st August 2017, Volume 2, Issue 2, JSEIS, CAOMEI Copyright © 2016-2017
www.jseis.org

A conceptual framework BDDMBT to overcome the quality challenges of mobile network

[1] Neelam Nasir, [2] Muhammad Nabeel
Department of Computer Science and Software Engineering
International Islamic University, Islamabad, Pakistan
Email: [1]neelamnasir2002@yahoo.com, [2]nabeelsr@yahoo.com

ABSTRACT

Increasing complexity and size of mobile networks are challenging factor for testing and requirement process, both phases of software development life cycle have great prolonged impact on cost and quality. In mobile network industry due to rapidly emerging competition the organizations are focusing on gaining quality attributes, not compromising on techniques and tool to ensure quality. To reduce unambiguous requirements may reduce testing interval thus, in order to resolve these issues, we have combined BDD to MBT. In this regard, the name of the suggested framework is Behavior Driven Development Model Based Testing (BDDMBT). An exploratory research as one of the objective of the study has been used to gather preliminary information that will help to define problems and suggest the conceptual framework BDDMBT.

Keywords: behavior driven development; model based testing; mobile network; model based testing;

1. INTRODUCTION

The main focus of software development process is to deliver quality as fast and economical as possible. The mobile network industry is becoming increasing powerful and smarter in mobile functionality that results in increasing complexity and size. Complexity and size of software are challenging to maintain cost and quality. Rapidly emerging competition and as users are growing [1], thus important of quality is also a significant phenomenon. Organizations are not compromising on techniques and tools to ensure quality [2]. Testing is one of the important factors in increasing the quality of a product, but time and cost of testing is major proportion to gain quality.

Reducing cost and maintaining quality are key success of a project. Unambiguous requirement and automatic test case generating are two main factors for reducing testing interval and cost without reducing quality. Requirements errors are additionally expensive to fix in later phases as in requirement analysis and design phases of software development. The preventing and eliminating errors in early stage during the requirement elicitation and specification phases of software development is significant and an efficient way to ensure quality [3]. The only real measure of quality is whether it fulfills user needs in a reliable manner [4]. Identifying the needs of a system has a lot of benefits further than code correctness and establishing communication and cooperation between inter-departmental. Using behavior-driven development (BDD) [5], to understand these needs from the start of software development process through testing external business behaviors, is a great technique to ensure a quality of a project. BDD focuses on obtaining a clear understanding of desired software behavior through discussion with stakeholder [3]. BDD cannot meet desired quality attributes because behavior driven testing have some shortcomings – one of the most explicit is manual test case generation has become too expensive and labour intensive for complex and huge software systems.

For automation of test cases generation Model Based Testing (MBT) is a best testing technique if unambiguous requirements are available for modeling of functional aspects of system under test to simplify the models of complex systems [6]. This is significant to prevent some faults before reaching the test execution phase and effect on decreasing time and cost of testing. Therefore, we proposed a conceptual framework BDDMBT, combine BDD to MBT (Model based testing) for acquiring desired quality attributes. The combined advantages of both techniques are significant to provide better solution through generating test models from test scenarios. The test scenarios connect in test models in order to create a better overview and analysis of the behavior of the System Under Test (SUT) and disclosure of ambiguities in requirement specification and design to overcome cost pressure and time duration of testing to achieve desired quality attributes of more complex and competitive environment of mobile networks.

To achieve our research objective, we have formulized research questions to gather groundwork information from literature that will help to identify problems and proposed solution. These questions are;

- What is BDD?
- What is MBT?
- Is literature available that tell us how we can use BDD and MBT that tell us relationship among them?
- What are Challenges to use or introduce BDDMBT?

This paper is structured as follows: Section 2 provides background about BDD and MBT based on related studies. Section 3 provides elaboration on the identified relationships of BDD and MBT characteristics and presents a proposed model BDDMBT that encapsulates these characteristics. Section 3 represents the case study. Lastly, the Section 4 gives the conclusion and future work.

2. BACKGROUND

This section provides preliminary information related to Behavior Driven Development (BDD) and Model Based Testing (MBT) technique to analyse the features of BDD and MBT and combines to overcome limitation.

a) Behavior driven development

BDD is a disciplined technique for developing software process in such an environment where the product owner, a programmer and tester – collaborate on system behavior [5, 7]. BDD introduced by Dan North [8] is based on synthesis and refinement practice. BDD is in fact an origin from Test Driven Development (TDD) and Acceptance Test Driven Development (ATDD) [3]. BDD is much better as test cases are written as early as in requirement phase and emphasize the system behavior in the test cases. It is actually a shifting from thinking in "test" to "behavior". BDD provides best understanding environment by everyone involved in development including the customers as well. The team collaborates by writing down a common and shared knowledge with well-known format. A high formal humane language is used to create test cases [9]. Gherkin is language for BDD, which uses living specification, defines scenarios format as in Given-When-Then to generate structure about the behaviors [10].

Gherkin provides few keywords to build a Domain Specific Language (DSL) to communicate between developers and business stakeholders. In DSL, scenarios are described in pattern of context-action-outcome. Keywords 'given' is used for context, 'when' for action and 'then' for outcome.

The temple for writing scenarios is as below,

Scenario #: [title of scenario]

Given [set a precondition]

And [set some more preconditions]……

When [event occur]

Then [Outcome]

And [some more outcomes]…..

b) Limitations that discourage the usage of BDD

- BDD provides limited support to the analysis phase, and none to the planning phase.
- Manual design of test cases.
- Exhaustive testing due to the fact of putting high efforts to managing manual test data

c) Model based testing (MBT)

Model-based testing is a software testing technique widely-used for automating the generation and execution of tests, in which the test cases are derived from a model that describes the functional aspects of the SUT [11]. MBT is automation of black box testing, typically involves the following four stages:

1. Designing a testing model for SUT: modeling is used to describe the expected behavior of the system. Models are simple representation of complex systems. For computing, models typically used the state graph or finite state machine. Models are not only used for test generation but also used to ensure bi-directional traceability by providing feedback and error detection between the requirements and the model.
2. Validating the model: detect the errors in the model and to ensure the coverage of the system behaviors.

3. Generate the tests: this step is fully automated to generate test cases from model. But the engineers can control on the number of tests generation, the system parts to be executed and coverage criteria are used.
4. Execute the test: execute the concrete tests with expected output and generate test oracles.

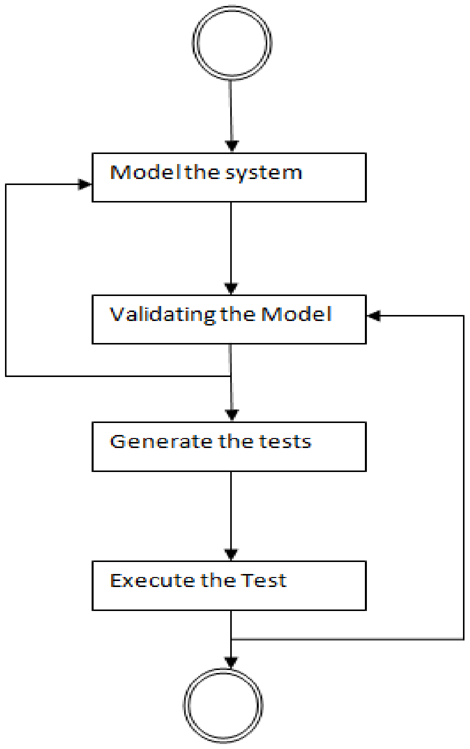

Figure. 1 Steps of model based testing

d) Related work

Manual design of test cases is a costly and time-consuming process. In order to generate automatic test cases, the model checker is used from the formalized requirements [12]. Most of the BDD supporting tools require that tests should be written components that only exist when the system is already implemented and in low-level events. As a result of such low-level of thought, BDD tests can hardly be reused with diverse artifacts and with versions of the system. To resolve this problem the ontology is proposed to raise the abstraction level by the means of a behavior-based development that is aimed at supporting test automation. This approach is used for automating the tests for functional requirements of interactive systems [5].

Large-scale process integration solution testing is cumbersome and complex. To tackle this problem, the researchers employed behavior-driven development. The researchers further used the notational language and Business Process Model to model domain-specific test cases. These test cases can be understood by both business stakeholders and developers. In addition, these test cases can automatically be executed as well [9].

3. BEHAVIOR DRIVEN DEVELOPMENT MODEL BASED TEST (BDDMBT)

In BDDMBT, we combined characteristic of BDD and MBT for unambiguous requirement and generated automatic test cases. As BDD focuses on the requirement phases and provides the system behavior in the test cases which are automatically generated by MBT. BDDMBT encapsulates both of characteristic in such a way that feature files are used to generate system models and then automatic test cases are generated.

a) Stages of BDDMBT

The proposed BDDMBT involves five stages as depicted in Figure 2.

1. Feature file: consists of scenarios which are behavior-driven representations of requirements, written in Given-When-Then format.
2. Designing model: representation of expected behavior of the system to generate tests also used to ensure bi-directional traceability by providing feedback and error detection between the requirements and the model.
3. Validating the model: detects the errors in the model and to ensure the coverage of the system behaviors.
4. Execute the test: execute the concrete tests with expected output and generate test oracles.
5. Generate the tests: this step is fully automated to generate test cases from model. But the engineers can control the number of generated tests, the system parts to be executed and coverage criteria are used.

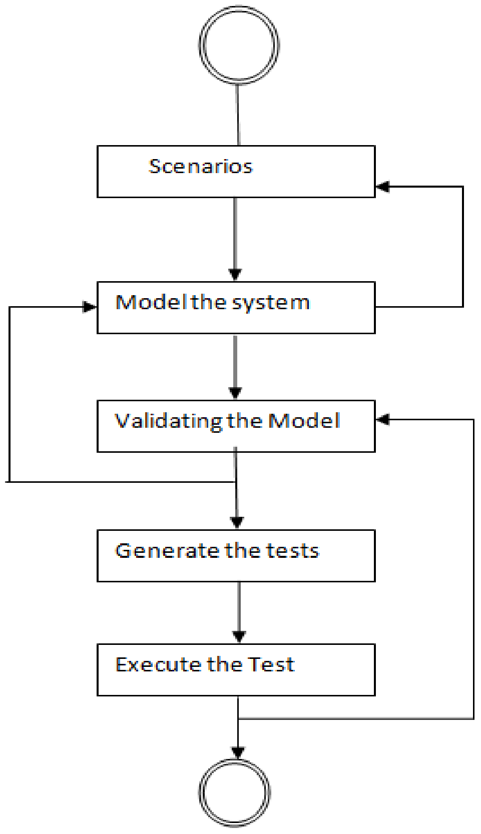

Figure. 2 Steps of the proposed BDDMBT

b) Features of BDDMBT

- Removes ambiguities of requirements.
- Generating test models from scenarios.
- Provide better overview and analysis of the behavior of the system under test.
- Automatic generation of test cases.
- Better test data management.
- Quickly and easily respond to change.

4. CASE STUDY

Our research is based on the case study. We formulated BDDMBT for mobile networks in order to overcome the challenges of mobile networks to compete the market competition and revisited to next generation. Some of the mobile network challenges are shown below:

- Networking technologies
- The network architecture
- Competition from new players
- Quality of services
- Data transfer rate and latency
- High data rate services in any time and any where
- Reliable connection
- Secure communication

a) Systems under test

The SUT is used in Figure 4 which describes architecture of mobile network system at abstract level. The mobile switching center (MSC), base station controller (BSC), and base transceiver system (BTS) are mentioned. In-completed and ambiguous requirements, BDDMBT will help to requirement gathering and illustration process by behavior-driven scenarios and generate automated test cases. As the system is more complex and at large level, therefore the essence of automatic test cases has become more important. It will rather consume more effort and time to generate and control such level of system.

This gives a significant birth to BDDMBT for providing best solution to reduce cost, increase efficiency of the product development in telecom domain.

Figure. 3 System under test

5. CONCLUSION

A mobile phone is no longer just a communication device; it replaced the personal computers by supporting multifunctional application and services, even business critical applications and services are shifting into a mobile device. Emerging importance of mobile services and future demands putting huge pressure on mobile networks. Mobile networks have many challenges which need to overcome. Hence, to reduce vulnerability and ensuring quality we proposed a conceptual framework BDDMBT by combining features of both BDD and MBT. BDDMBT will be better technique by removing ambiguities from requirements and automatic generation of test cases from these unambiguous behavior-driven requirements which will facilitate to overcome the challenges of mobile networks. We noticed some challenges about theoretical framework of BDDMBT; one of them is that the tool and the language are not present. The proposed conceptual framework will be implemented and that theoretical framework will be validated. For theoretical framework, the tools and languages are needed. We will develop a tool which will support the implementation of BDDMBT

ACKNOWLEDGEMENT

We are thankful to International Islamic University Islamabad, Pakistan to provide resources to complete this research.

REFERENCES

1. Amalfitano, D., et al., *MobiGUITAR: Automated model-based testing of mobile apps.* IEEE software, 2015. 32(5): p. 53-59.
2. Bagheri, H., et al., *Software architectural principles in contemporary mobile software: from conception to practice.* Journal of Systems and Software, 2016. 119: p. 31-44.
3. Mahalakshmi, G. and V. Vani, *Test Lifecycle in Behavior Driven Development.*
4. Solis, C. and X. Wang. *A study of the characteristics of behaviour driven development*. in *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*. 2011: IEEE.
5. Carrera, Á., C.A. Iglesias, and M. Garijo, *Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development.* Information Systems Frontiers, 2014. 16(2): p. 169-182.
6. Kramer, A. and B. Legeard, *Introducing MBT in your company.* Model-Based Testing Essentials: Guide to the ISTQB® Certified Model-Based Tester Foundation Level, 2016: p. 179-200.
7. Silva, T.R., J.-L. Hak, and M. Winckler. *A Behavior-Based Ontology for Supporting Automated Assessment of Interactive Systems*. in *Semantic Computing (ICSC), 2017 IEEE 11th International Conference on*. 2017: IEEE.
8. North, D., *Introducing BDD, Better Software Magazine*. 2006.
9. Lübke, D. and T. van Lessen, *Modeling test cases in BPMN for behavior-driven development.* IEEE software, 2016. 33(5): p. 15-21.
10. Almeida, L., E. Cirilo, and E.A. Barbosa. *SS-BDD: Automated Acceptance Testing for Spreadsheets*. in *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing*. 2016: ACM.
11. Stephan Schulz, J.H., Antti Huima. *MODEL-BASED TESTING*. in *2nd ETSI MBT User Conference*. 2012.
12. Rajan, A., *Automated requirements-based test case generation.* ACM SIGSOFT Software Engineering Notes, 2006. 31(6): p. 1-2.

AUTHORS PROFILE