International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Envision Issue Inclined of an Application Exploiting Model-Based Integration and System Test Automation

Kiran V. Bakka[1], Aarti Deshpande[2]

[1]Department of Computer Engineering, G.H.R.C.E.M, Savitribai Phule Pune University, Pune, Maharashtra, India

[2]Professor, Department of Computer Engineering, G.H.R.C.E.M, Savitribai Phule Pune University, Pune, Maharashtra, India

Abstract: ***The world's expanded reliance on programming enabled automated frameworks has elevated real worries about programming and coding dependability furthermore, configuration. New practical approaches and apparatuses for software stability and quality assurance are required. This paper shows an automated test era technique, called New Model-based Integration and System Test Automation (MISTA), for implementation of programming frameworks. Given a Model-Implementation Depiction (MID) specification, MISTA creates test code that can be executed quickly with the procedure under test. The MID specification utilizes a Petri net to catch both control and information related prerequisites for functional testing, access control testing, or deep testing with exhaustive models. Subsequent to producing experiments from the test model as per a given model, MISTA changes over the experiments into executable test code by mapping model-level components into usage level develops. MISTA has executed test generators for different test scope criteria of test models, code generators for different programming and scripting dialects, and test execution environments, for example, Java, C, C++, C#, HTML, Selenium, Webdriver, and Robot Framework. MISTA has been connected to the functional and exhaustive testing of different real-world programming frameworks. Our surveys have shown that MISTA can be profoundly powerful in flaw identification.***

Keywords: Functional testing, Model-based testing, Petri nets, Exhaustive testing, Configuration testing, Regression testing, Software assurance.

1. Introduction

The World Wide Web and the Internet have drawn the general populace into the world of computing. We purchase stock and mutual assets, download music, view movies, get medical guidance, book hotel rooms, sell personal items, schedule airline flights, meet people, do our banking, take college courses, buy groceries—we do just about anything and everything in the virtual world of the Web [7]. Arguably, the Web and the Internet that permits it are the most important developments in the history of computing.

Web-based schemes and applications (WebApps) carry a complex selection of content and functionality to a wide populace of end-consumers [13], [14]. Web corporate is the process used to create high-quality WebApps. Software testing is an essential phase of software development life cycle because of issues generated while developing. Software testing is a standout amongst the most vital and pivotal stage in the product improvement life cycle process, expending a normal of 40% to 70% of programming improvement process [2]. Programming testing is a technique, which is utilized for evaluating the usefulness of a product program.

Today numerous product applications are composed as automated application that keeps running in an Internet Program. The financial importance of online application builds the significance of controlling and enhancing its quality. ISO 9000 gives higher customer appreciation of quality control [6]. The quality certification of a framework relies on automation testing that declines the test cost and increments work productivity. Automation empowers more test cycles because of repeatable tests and more continual test runs. It likewise facilitates fast, efficient verification of prerequisite changes and bug fixes, and minimizes human misses.

Model Based Testing (MBT) focuses on just a required segment of framework. Experiments are not yet executable with the SUT because of two causes. To start with, tests created from a model are regularly insufficient on the grounds that the real parameters are not decided. For instance, when a test model is spoken to by a state machine or grouping chart with imperatives (e.g., prerequisites), it is difficult to consequently focus the genuine parameters of test arrangements so all compels along every test successions are fulfilled [8]. Second, tests created from a model are not specifically executable in light of the fact that displaying and programming use distinctive dialects. Mechanized execution of these tests regularly requires usage particular test drivers or connectors. Model-based Integration and System Test Automation (MISTA) catch both information and control flows of test necessities. It converts model-level tests into executable code.

The remaining paper is organized as follows. The section 2 includes the survey of different methodologies that were been developed using different tactics has been discussed briefly. The section 3 includes MISTA's architecture. The section 4 introduces the overview of proposed system. The section 5 gives the comparative analysis of test generators for savvy testing. The section 6 concludes the paper in brief.

2. Related work

This work is motivated by the test automation problems found from our prior research on exhaustive testing. These papers are related to Web applications, MISTA, Model-based testing tools, testing with Petri nets, Modelling and testing of access control policies, Exhaustive testing, Configuration testing, Regression testing.

A. Web applications

Various web automation testing tools which will help us to understand the automation testing as well as the tools available for automation testing [2]. To select the best tool for a task various issues like ease of integration should be considered and evaluated against the cost and performance. Automation Software Testing saves time and cost. The main issue about it is tool compatibility with the design and implementation of an application.

B. MISTA

MISTA presents an automated test generation technique, called Model-based Integration and System Test Automation (MISTA), for integrated functional and security testing of software systems [1]. The specification uses a high-level Petri net for functional testing, access control testing, or penetration testing. It is highly effective in fault detection and supports not only various programming and scripting languages, but also a number of test execution frameworks. The main issue about this system is that automation of test sequences deals with simple test models.

C. Model-based testing

A model-based way to deal with programmed era of executable access resistor tests utilizing predicate/move nets [11]. Part consent test models are constructed by coordinating ultimate access resistor rules with utilitarian test models or conventions of the related exercises (the framework capacities) [3]. It gives arranged procedures to building part authorization test models and backings the era of test code in an assortment of dialects. The main issue about it is RBAC strategy can be executed wrongly due to several causes.

D. Petri nets

A Petri net (also known as a place/transition net or P/T net) is one of several mathem-etical modelling languages for the description of distributed systems [4]. The methodology consists of four testing strategies: transition-oriented testing, state-oriented testing, data flow-oriented testing, and specification-oriented testing [5].

E. Modelling and testing of access control policies

It focused on the testing of role-permission assignments and user-role assignments in RBAC, where users, roles, and permission rules are predefined [12]. It also automatically generates executable access control tests from the test models. The empirical studies using three Java programs have demonstrated that the approach is highly effective in detecting access control defects [3].

F. Exhaustive testing

It presents certain logistical problems. For even small programs, the number of possible logical paths can be very large [7]. For instance, consider the 100 line suite in the language C. After some basic data declaration, the program contains two nested loops that execute from 1 to 20 times each, dependent on situations specified at idea. Inside the inner loop, four if-then-else hypotheses are required. There are around 1014 possible tracks that may be accomplished in this suite.

G. Configuration testing

Automation of you web application testing allows your development team to make changes and refactor code with more confident, they can rapidly check the application's usefulness after each change [7]. On the other hand, really building programmed tests for web requests can be testing on the grounds that the client interface of your application may change consistently [9], in light of incompatibilities in the middle of programs and on the grounds that you more often than not have to bolster different server or customer stages.

H. Regression testing

It is conducted when system requirements or implementation are changed. If test cases are not completely generated, tester needs to determine whether they have become invalid and whether they have to be changed [5]. Using ISTA, however, only needs to change the specification for test generation. Consequently testing your web request is a decent approach to guarantee that new forms of your request don't present bugs and relapses [6].

3. Existing System

Dianxiang Xu [1] proposed a system that computerized Test Generation Method for Software Excellence Assertion. This system is shown in fig.

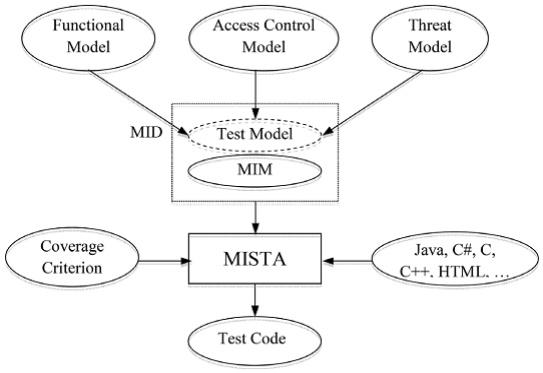

Figure 1: System Architecture of MISTA

The important phases of this system are:

- The input to MISTA is called a Model-Implementation Description (MID).
- The test model represented by a PrT net can be a functional model, an access control model, a threat model, cost effective testing model or symbolic methods model.
- MIM maps the elements of the test model to the target implementation-level constructs.
- MISTA generates executable test code.

The main drawback of this existing system is that automation of test sequences deals with simple test models. Thus, a new system is needed.

4. Proposed Work

The major intend of this paper is to automated test era strategy for applications that give software quality confirmation and it will be very successful for fault detection. We have proposed a strategy New Model-based Integration and System Test Automation (MISTA) which will coordinate functional testing of programming frameworks. Given a Model-Implementation Description (MID) specification that contains test model which can be a functional model, an entrance control model or exhaustive model and Model-Implementation Mapping which will be in responsibility of mapping components of test model to target execution level develops. MISTA creates executable test code in target language(for example, Java, C#, C, C++, HTML and VB) and number of test execution systems, (for example, JUnit, Robotium, Selenium, WebDriver, JSON-RPC, Robot Framework) as indicated by a scope basis of the test model, (for example, reachability scope, state scope, transition scope, profundity scope, and objective scope).

The proposed plan produces tests for reachability scope with robustness tests (power tests). The most critical component of this plan is that both substantial ways and invalid ways are tried. This strategy perform near investigation of test generators on the premise of different parameters like fault identification ability, time execution and adaptability of test generators results into savvy testing and create a dynamic reachability graph which manages more mind boggling test models. This system is shown in fig.

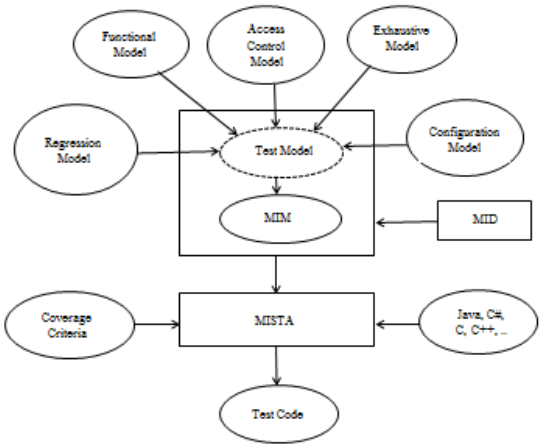

Figure 2: System Architecture of New MISTA

5. Comparative Analysis of Test Generators

Analysis standards are a widespread resource in the direction of measure the flaw recognition ability of test suites and to bullock test suite generation [10]. Their definitions are, however, often unclear and informal. It is very hard to even find a definition of what scope actually are. Following are the various scopes.

TS = Test Sequence, V = Vertex, TG = Test Goal, T = Transition, T1/T2 = Transitions, Var = Variable, C = Configuration, AC = Atomic Condition, S = Scenario.

- State scope if state is visited by at least one test sequence.

V ϵ TS

- Transition scope if transition is traversed by at least one test sequence.

TG = V (TS)
T ϵ TS

- Transition-Pair scope if pair of adjacent transition is traversed by at least one test sequence.

TG = T (TS)
T1/T2 ϵ TS

- Reachability scope if each edge is visited by at least one test sequence.

Var ϵ TS
T $\rightarrow$ Var

- Path scope if possible path is traversed by at least one test sequence.

TG = C (TS)
List T
get Path (C, T)

- Profundity scope if all sequences whose lengths are less than or equal to a given depth are covered.

T ϵ TS
AC ϵ T

- Scenario scope is sequence of transitions in application.

TG = TS (T)

- Objective scope if each reachable goal by at least one test sequence

TG = TS (S)

If the test generator will satisfy the above condition of scope then it is belonging to that scope. From above scopes following comparisons is made from which time and cost analysis will be found.

Table: Comparative analysis of test generators for savvy testing

Tool	Formulism	State Scope	Transition Scope	Transition -Pair Scope	Reachability Scope	Path Scope	Profundity Scope	Scenario Scope	Objective Scope	Time Required	Cost
GOTCHA-TCBean	EFSM	Yes	Yes	No	No	No	No	No	No	More	More
Graph Walker	EFSM	Yes	Yes	No	No	No	Yes	No	No	More	More
TestCast	EFSM	Yes	Yes	Yes	No	No	No	Yes	No	More	More
TestOptimal	FSM	Yes	Yes	No	No	No	No	Yes	No	More	More
AGEDIS	UML, Collaboration, State diagram	Yes	Yes	No	No	No	No	No	No	More	More
ParTeG	UML, State diagram	Yes	Yes	No	No	No	No	No	No	More	More
Conformiq Qtronic	UML	Yes	Yes	Yes	Yes	Yes	No	No	No	Less	More
Test Designer	UML, State, Object diagram	Yes	Yes	Yes	No	No	No	No	Yes	More	More
Spec Explorer	State diagram	Yes	No	No	No	No	No	No	No	More	More
MISTA	Petri nets, State diagram	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	More	More
New MISTA	Time Petri nets	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Less	Less

6. Conclusion

The current paper reports our a method for automation of test activities exploiting reachability graph with more complex test models and envision fault inclined to an application exploiting New Model-based Integration and System Test Automation (MISTA). It additionally bolsters various programming dialects (e.g., Java, C#, C++, VB) and test execution systems (e.g., JUnit, Selenium, Webdriver and Robot Framework). Because of the system's extensible structural planning, it is anything but difficult to present another test generator, target dialect, or test execution environment. The approaches we propose address the impacts on testing coverage and productivity and reduce cost and time of testing.

7. Acknowledgment

We would like to thanks all the authors of different research credentialsdiscussedin writing this paper. It was very knowledge achievement and cooperative for the advanceexploration to be done in future.

References

[1] Dianxiang Xu, Senior Member, IEEE, WeifengXu, Senior Member, IEEE, Michael Kent, Lijo Thomas, and Linzhang Wang," An Automated Test Generation Technique for Software Quality Assurance", IEEE TRANSACTIONS ON RELIABILITY, VOL. 64, NO. 1, MARCH 2015.

[2] Monika Sharma, RigzinAngmo," Web based Automation Testing and Tools", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (1), 2014.

[3] Dianxiang Xu, Senior Member, IEEE, Michael Kent, Lijo Thomas, TejeddineMouelhi, and Yves Le Traon, —Automated Model-Based Testing of Role-Based Access Control Using Predicate/Transition Nets", IEEE TRANSACTIONS ON COMPUTERS, VOL. 64, NO. 9, SEPTEMBER 2015.

[4] Zhu and X. He, —Amethodology for testing high-level Petri nets", Inf. Softw. Technol., vol. 44, pp. 473–489, 2002.

[5] Dianxiang Xu ,—A Tool for Automated Test Code Generation from High-Level Petri Nets', PETRI NETS 2011, LNCS 6709, pp. 308–317, 2011.© Springer-Verlag Berlin Heidelberg 2011.

[6] [Book]. Available: Ron Patton, –Software Testing".

[7] [Book]. Available: Roger S. Pressman, –Software Engineering".

[8] Xu, D., Xu, W., Wong, W.E., –Automated Test Code Generation from Class State Models", International J. of Software Engineering and Knowledge Engineering 19(4), 599–623, 2009.

[9] Desel, A. Oberweis, T. Zimmer, and G. Zimmermann, –Validation of information system models: Petri nets and test case generation," Proc. SMC'97, pp. 3401–3406, 1997.

[10] L. Gallagher and J. Offutt, –Test sequence generation for integrationtesting of component software," Comput. J., Advance Access, Nov.2007.

[11] Pretschner, Y. L. Traon, and T. Mouelhi, –Model-based tests for access control policies," in Proc. 1st Int. Conf. Software Testing Verification and Validation (ICST'08), Lillehamer, Norway, Apr. 2008.

[12] H. Hu and G. Ahn, –Enabling verification and conformance testing for access control model," in Proc. 13th ACM Symp. Access Control Models and Technologies, 2008, pp. 195–204.

[13] [Online].Available:http://www.magentocommerce.com

[14] [Online]. Available: http://www.zen-cart.com