FUNCTIONAL TEST CASE GENERATION AND REDUNDANCY REMOVAL BASED ON MODEL DRIVEN TESTING USING UML ACTIVITY DIAGRAM

Runal G.
Department of Information Technology,
Bharati Vidyapeeth Deemed to be University, Pune
Pune-411043, Maharashtra. India.

Prof. Pramod Jadhav
Department of Information Technology,
Bharati Vidyapeeth Deemed to be University, Pune
Pune-411043, Maharashtra. India.

Dr. S. D. Joshi
Department of Information Technology,
Bharati Vidyapeeth Deemed to be University, Pune
Pune-411043, Maharashtra. India

Prof. Pruthviraj R. Pawar
Department of Computer Engineering,
Bharati Vidyapeeth College of Engineering, Navi Mumbai
Navi Mumbai-400614, Maharashtra. India.

ABSTRACT
Software testing is that the primary activity to provide reliable software system. Irresponsible of a software system is incredibly abundant passionate about the means of testing performed. Software system testing that is sometimes last activity of the software system development cycle is performed below the pressure. Quality and irresponsible of software system area unit abundant passionate about take a look at ways that area unit dead by take a look at cases. Generation of optimized take a look at ways may be a difficult a part of the software system testing method. During this paper, a very important effort is formed to propose a brand new technique to get the optimized take a look at ways from UML sequence diagram. A tailored algorithmic rule known as as Firefly algorithmic rule is employed to induce the vital ways. Firefly algorithmic rule is met heuristic and impressed from flashing behavior of fireflies. A case study of Patient registration system area unit is employed on justify the planned approach. Info Flow Metric and their cyclamate complexity area unit used for
prioritization of take a look at ways. Results indicated that optimized ways from sequence diagram haven't any redundancy and made the higher results.

This system explains a brand new property- and model-based testing technique exploitation UML/OCL models, directed by property diagrams and an instrument for helping the temporal properties rationalization. The patterns square measure suggested within the TOCL language, correlate edition of Dwyer’s property diagrams to OCL. The patterns square measure accustomed distribute temporal needs while not having to be told a fancy physical logic like LTL or CTL.

This analysis aims to come up with action at law mechanically from UML diagram since model primarily based testing that conducted on early section of software system development method show higher potency. UML diagrams utilized in this analysis square measure activity, sequence diagram and SYTG because the combination graph. These 3 are established because the most compatible diagram to come up with action at law from previous analysis. Technique projected during this paper is Depth initial Search formula that’s changed to come up with expected take a look at cases. Gift a way which will be accustomed (i) mechanically verification sure whenever attainable (ii) tighten a group of bounds projected by the user and (iii) guide the user within the bound choice method. This approach could increase the usability of UML/OCL finite verification tools and improve the potency of the verification method.

**Key words:** UML, Sequence Diagram, Behavioral model Test generation, depth first search algorithm, software testing, test cases generator, Code Smell, Refactoring, Test case redundancy


1. INTRODUCTION

Software testing is a crucial method that often won’t to validate the standard of the software system. The proper testing will increase software package quality. With the rising demand for reliable software system, software system testing will add up to five hundredth of the whole software system value. Software system testing has not solely evolved for look errors or bugs within the software system however it becomes a discipline for evaluating the standard software system [1]. In line with IEEE testing is, “The method of elbow grease or evaluating system or system elements by manual or machine-driven suggests that to verify that it satisfies fixed requirements” [2]. It may be performed manually or mechanically. Machine-driven software system testing is found to be higher than manual testing. Software system testing method wants additional effort with an individual’s interface. During this analysis paper author principally generating prioritization and optimization based mostly take a look at method from UML sequence diagram victimization Firefly algorithmic rule. Software system testing usually used 2 ways that area unit recording machine testing and white box testing. White box testing is understood as structural testing, recording machine testing focuses solely the output of the software system testing is understood as useful testing those victimization useful criteria [3, 4].

Model-based checking (MBT) [5] consists in employing a model to get check cases and work out the test finding, in conditions of projected behavior of the system underneath check (SUT). Models are designed supported the informal needs of the method and used by model
coverage principle to work out check cases. Additionally, the models create it doable to work out the check oracle, mainly the projected results of the check. When a concretization step, the theoretical checks will then be dead on the SUT as well as also the test finding is often mechanically allowed. MBT is so a suitable manner to automate checks creation and, to some level, checks implementation numerous methods for MBT exist [6], supported totally different forms victimization, as an example, automata (mealy machines, IOLTS, IOSTS), or pre-post conditions symbols (B, VDM, JML, UML/OCL), etc. related to them, check choice principle create it doable to check cases that assurance a known level of assurance that the system has been suitably handled.

During this system aims to boost associate degree accessible MBT method, supported UML/OCL models, that method originally targets practical testing. This 1st method has been progressed and with success transmitted into the business because the sensible take a look acing Certify it test creator. This instrument works by mechanically employing a structural take a look at choice principle on the model, particularly the division coverage of the OCL specification [7] of operations limited in a very UML category diagram. though this method is kind of efficient in apply, it gets from its judgment, as well as therefore, particular behaviors of the system, that need a additional intensive take a look at effort, don’t seem to be abundant focused by this testing approach. To beat this downside, effective take a look at choice criteria area unit proposed. These consist in scenario-based testing methods that goal to use additional widely particular components of the thought-about system. Such take a look at situations area unit articulated in a very committed matter language that explains sequences of steps (usually operation calls) which will be completed, at the side of potential middle states entered throughout the development of the situation. Not withstanding, the planning of the take a look at situations remains a manual job that we tend to aim to change. Throughout earlier procedures within the utilize of situations, we’ve detected that situations usually create from a manual explanation of a known assets that exercises the effective of the system [8].

Modified DFS algorithmic rule is planned during this analysis as Associate in nursing improvement of analysis from Tripathy et al.4. In our experiment, current DFS algorithmic rule that applied for action at law generation method generated some redundant node. Thus, a modification is required to induce optimum check cases result. Comparison check cases result for each said algorithms is provided within the fifth section. The second section can discuss regarding state of the art of this field. Later section describes our planned approach in generating action at law mechanically. Conclusions area unit given within the last section. This paper aims to produce higher analysis and comparison of action at law generated from totally different UML diagrams with new planned method; that is changed deep initial Search (DFS) algorithmic rule.

Refactoring is a crucial technique of eliminating the weaknesses of a package by applying modifications to the ASCII text file while not ever-changing the outward behavior of the system [9]. In raising the standard of any code, it’s essential that the standard of the code is maintained [10]. There area unit differing types of code smell and their individual refactoring approaches as named by [11]. This paper focuses on the lazy category code smell that is as a results of presence of a category that’s doing very little to face as a category. A lazy category will merely be outlined as a category that’s too little to exist. This suggests that the price of its existence is quite its price. A lazy category is usually as results of refactoring of an outsized or God category. A lazy category is refactories victimization the inline technique of refactoring.

Runal G., Pramod Jadhav, Dr. S. D. Joshi, Pruthviraj R. Pawar
2. LITERATURE SURVEY

In this project, Authors [1] Aritra Bandyopadhyay, Sudipto Ghosh, Vikas Panthi, Durga Prasad Mohapatra published a unique testing approach that mixes data from UML sequence models and state machine models. Current approaches that bank entirely on sequence models don’t contemplate the results of the message path underneath check on the states of the taking part objects. We have a tendency to extend their Variable Assignment Graph (VAG) based mostly approach to incorporate data from state machine models. The extended VAG (EVAG) produces multiple execution methods representing the results of the messages on the states of their target objects. We are going to additionally compare the price and effectiveness of our approach with state machine based mostly approaches.

Author proposes a method for check Sequence Generation exploitation UML Model Sequence Diagram. UML models provides a ton of knowledge that ought to not be unheeded in testing. During this paper main options extract from Sequence Diagram at the moment we are able to write the Java ASCII text file for that options consistent with Model Junit Library. Model JUnit is associate degree extended library of JUnit Library. By exploitation that ASCII text file we are able to generate action Automatic and check Coverage. This paper describes a scientific action Generation Technique performed on model primarily based testing (MBT) approaches by exploitation Sequence Diagram.

In this project, Authors [2] Md Azaharuddin Ali et.al, S. Shanmuga Priya et.al, proposed a method for generating the check cases mechanically. This projected methodology decreases time and will increase the reliableness of the software package testing part. The main criterion of software package checking is to supply test cases. This methodology contains reworking the state diagram (UML) into finite state machine (DFA / NDFA) wherever each node represents state and transactions area unit diagrammatical by arrow connecting the states. The projected methodology achieves enough check coverage while not increasing the quantity of check cases. It additionally attains a lot of vital coverage like transition coverage, transition try coverage, and provides state coverage.

They proposes a piece that represents a model based mostly take a look acting techniques from that the test methods area unit automatic generated and achieved earlier or through the method of development and then, once the code of application is offered, the take a look at cases may well be dead that aids in fixing the errors at initial part. Unified Modeling Language (UML) Sequence Diagram is taken into account for planning and a case study of Medical Consultation System is taken for the projected work. Testing are going to be disbursed on collected demand, planning and committal to writing. Though, if testing is followed within the initial innovated SDLC several errors would be removed and can be prohibited while not commercial enterprise to succeeding part. Therefore, testing should not be isolated to one part alone in SDLC.

In this project, Authors [3] Ching-Seh Wu and Chi-Hsin Huang, They propose a technique of Model-Based Testing (MBT) to reinforce testing of interactions among the online services. The technique combines Extended Finite State Machine (EFSM) and UML sequence diagram to get a take a look at model, referred to as EFSM-SeTM, they need additionally outlined numerous coverage criteria to get valid take a look at methods from EFSM-SeTM model for an improved take a look at coverage of all attainable situations. This approach focuses on making a take a look at model from the UML sequence diagram and EFSM. Derived the take a look at ways from the take a look at model for composite net service testing. To develop the take a look at model, EFSM-SeTM is generated from sequence diagram and EFSMs for composite net service testing. The EFSM-SeTM represents states of objects and interaction info within the take a look at model.
Refactoring lowers the price of enhancements

When a package is winning, there's continuously a desire to stay enhancing it, to repair issues and add new options. After all, it's known as a package for a reason! However the character of a code-base makes an enormous distinction on however straightforward it's to form these changes. Usually enhancements are applied on prime of every different during a manner that creates it more and tougher to form changes. Over time new work slows to a crawl. To combat this variation, it is important to refactor code in order that additional enhancements do not cause needless quality.

Refactoring is a part of day-to-day programming

Refactoring is not a special task that might show up in a very project arrange. Done well, it is a regular a part of programming activity. After I got to add a brand new feature to a codebase, I inspect the present code and take into account whether or not it's structured in such some way to create the new modification simple. If it is not, then I refactor the present code to create this new addition simple. By refactoring initial during this manner, I typically notice it's quicker than if I hadn't distributed the refactoring initial.

In this project, Authors [4] Ke Z, Bo J, Chan WK, Papadakis M, Malevris N. and Zhang C, Groce A, Alipour MA, editors proposes the two well-known approaches presently being looked into square measure test suit prioritization and test suit step-down. In these 2 approaches, the check cases square measure generated which incorporates the redundant check cases. The two approaches square measure well established within the space of regression testing. There’s less attention given to redundancy shunning in an exceedingly recently developed application. Hence, there's a requirement for a way that averts redundant check cases from being generated in an exceedingly new system. Asaithambi9 steered refactoring as a attainable answer that may be explored to cut back the generation of redundant check cases. Automatic unit tests ought to be came upon before refactoring to confirm routines still behave obviously. Unit tests will bring stability to even giant refactors once performed with one atomic commit. a typical strategy to permit safe and atomic refactors spanning multiple comes is to store all comes in an exceedingly single repository, referred to as monorepo. With unit testing in situ, refactoring is then associate degree repetitious cycle of creating atiny low program transformation, testing it to confirm correctness, and creating another little transformation. If at any purpose a check fails, the last hard cash is undone and continual in an exceedingly totally different manner. Through several little steps the program moves from wherever it absolutely was to wherever you wish it to be. For this terribly repetitious method to be sensible, the tests should run terribly quickly, or the technologist would get to pay an oversized fraction of their time watching for the tests to end. Proponents of maximum programming associate degree alternative agile code development describe this activity as an integral a part of the code development cycle.

In this project, Authors [5]. Lashari SA, Ibrahim R, Senan N., Ahmed M, Ibrahim R, Ibrahim N, Fowler M. Proposes Refactoring the ASCII text file will eliminate code smells that would generate redundant take a look at cases. Refactoring is a very important methodology of eliminating the weaknesses of a package by applying modifications to the ASCII text file while not dynamical the outward behavior of the system. In up the standard of any software package, it's essential that the standard of the software package is maintained. There area unit differing kinds of code smell and their various refactoring approaches as named by This paper focuses on the lazy category code smell that is as a results of presence of a category that's doing very little to square as a category. A lazy category will merely be outlined as a category that's too little to exist. This suggests that the price of its existence is over its price. A lazy category is typically as results of refactoring of an outsized or God category. A lazy category is refactories victimization the inline methodology of refactoring.
This study still, applies the inline category refactoring technique to refactor the mechanical man ASCII text file to eliminate lazy category code smells that would probably cause generating redundant take a look at cases. This is often achieved by planning the detection and refactoring rules for lazy category smell. The code smell with its detection and refactoring technique is formally specified. Manually performed refactoring is time overwhelming and takes numerous efforts. Automating the method is indispensable. Hence, a tool named Direct Attention Thinking Tools (DATT) was developed to implement the detection and refactoring rules. DATT is evaluated victimization allot application by generating branch coverage and cyclamate quality for the first and refactories ASCII text file victimization herbaceous plant for mechanical man and also the results obtained were compared. Take a look at cases area unit generated before and once refactoring to validate DATT. Therefore, this paper is organized as follows: ensuing section discusses connected work on action minimization and refactoring, framework for this study is then presented. Implementation of the framework is mentioned next, followed by the results and analysis. Finally, the conclusion of the study is mentioned supported the findings.

3. CONCLUSION
This paper presents firefly formula based mostly check ways generation from UML sequence diagram. UML sequence diagram are often terribly useful in cluster level testing of the code. Check ways area unit typically utilized by code engineers to ascertain the practicality of the code. There is also an infinite variety of check ways for even tiny size code. Testing all ways in not possible as testing is sometimes performed in pressure. We’ve conferred a model-based checking method supported test assets. These final area units intended in very devoted forms that captures the dynamics of the system. Every asset is converted into an automaton, that new coverage principle are introduced, so as maybe the property. Additionally, we tend to suggest to improve the automation thus on show exact transitions that area unit directly associated with fault traces that don’t seem to be received by the property. This method prepares it attainable to establish a view of the testing to confirm that the assets are properly enforced. This type of method has been converted into an industry-strength tool and is currently projected as connect to the Certify It check creator. The benefits of this method area unit two fold. Today code testing may be a necessary method to confirm the created code is corresponding with business method needs. During this paper we tend to conferred changed DFS formula to generated automatic check cases mistreatment of UML activity diagram and sequence diagram. Our study shows DFS formula has to be changed as in our projected approach to get taken check cases. From the experiment result, the check cases generated supported activity diagram, sequence diagram, and SYTG (combination graph) area unit provided.

REFERENCES


