ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

Functional Testing Technique Using ODC to Identify And Predict Faults

K.Sivaprakash[1], Prof. Preethi harris[2]

PG Scholar (ME), Department of Software Engineering, Sri Ramakrishna Engineering College, Coimbatore, India[1]

Associate Professor, Department of Information Technology, Sri Ramakrishna Engineering College, Coimbatore, India[2]

Abstract: The type of faults in software testing techniques may differ and are more prone to detect. Based upon the features of the application under test, their performance varies. The current methodology like Novel Fault Classification Scheme, used to identify and classify faults based on least square software vector machine in software application. The Novel Classification Scheme based on least square SVM may requires more time and also with less accurateness. The Genetic algorithms are used to generate the test cases and fitness of each test case has been determined individually. The Orthogonal Defect Classification (ODC) mechanism identifies the faults and classifies based on the fault types for the given set of generated test cases through the genetic algorithm. The testing techniques such as functional testing, statistical testing, robustness testing and stress testing are utilized to classify the faults based on their types. Besides, the difficulty of switching among the techniques has been resolved using the Markov Decision Process (MDP) that maximizes the number of faults. The classification of the faults may be varied based upon the testing technique that has been utilized. The performance of the ODC fault classification mechanism and classification system is analyzed through the percentage of faults identified and classified based on the ODC mechanism.

Key words: Software testing, Fault classification, Software metrics.

I. INTRODUCTION

Software quality relates two different and distinct entities defined such as software functional quality and software structural quality concerned in a business context.

- Software functional quality reflects how well it complies with the given functionality, based on requirements or specifications. The attributes can also be described as the fitness for purpose of a piece of software or how it compares to competitors in the marketplace as a worthwhile product.
- Software structural quality relates the non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, the degree to which the software was produced correctly.

II. FAULT IDENTIFICATION AND CLASSIFICATION

During software testing, engineers have to cope with several heterogeneous software faults, which have to be detected and fixed before releasing the product. How many faults are in the software, and of what type, depends on several factors, such as the techniques adopted along the development cycle, the developers' skills, and the size and complexity of the system under test. In modern complex software systems, the case of applying just one testing technique is not sufficient to meet the desired quality with acceptable resources.

Each technique pursues a specific goal, resulting more or less prone towards some type of fault, more or less suitable for a specific kind of system, or for a particular kind of organization

Effective techniques are utilized for the specific soft-ware that is to be tested, and what techniques are more prone to detect a faults based on specific types. It has been reported to the tester's, resulting each time in different and undesirable

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

outcomes. Such practice often leads to immature testing processes: decisions are made based on subjective abilities, and the knowledge that may derive from past experiences in the organization remains implicit in the people's mind, unstructured, and, ultimately, not fully exploited.

III. THE ORTHOGONAL DEFECT CLASSIFICATION

Orthogonal Defect Classification (ODC) essentially means that we categorize a defect into classes that collectively point to the part of the process which needs the attention in cases such as characterizing a point in a Cartesian system of orthogonal axes by its (x, y, z) coordinates. In the software development process although activities are broadly divided into design, code, and test. The case of the process stages in several instances may overlap while different releases may be developed simultaneously. Process stages can be performed by different people and sometimes different organizations.

Therefore, the classification scheme which is widely applicable must have consistency between the stages. Ideally, it should be quite independent of the specifics of a product or organization. If the classification is consistent across phases and independent of the product, must be fairly process invariant and can eventually yield relationships and models that are very useful.

- orthogonally
- consistency across phases
- uniformity across products

The choices of defect types have evolved over time from the original five types which were refined by working with the IBM Mid-Hudson Valley Programming Lab to eight. The idea is to capture distinct activities in fixing a defect for given the programming paradigm and is limited in degrees of freedom. There are only so many distinct things are possible when fixing a defect. Adding a new capability to software (function) is quite different from a small change in a few lines of code say to correct values of variables (assignment). If the choices are orthogonal then it also leaves little confusion. The increase from five to eight occurred in dimensions relating to moving from a proof of concept to a production environment.

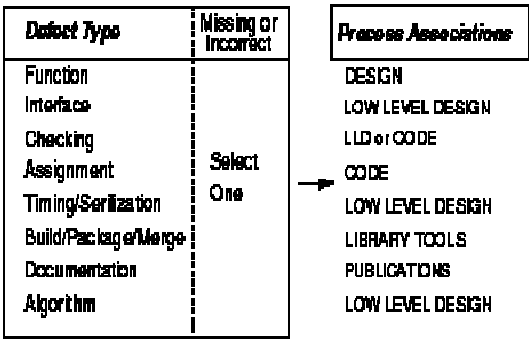

Figure 1 Defect Type and Process Associations

IV. THE TECHNIQUE

The system starts with the generation of the test cases along with the fault classification mechanism ODC, followed by the performance evaluations that are based upon the type of testing technique that has been utilized.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

4.1 Test case generation using genetic algorithm

The first step is to constructs test data and runs one test case. The algorithm takes two parameters to construct the test data and test case generation: a set M of java target methods and a genetic algorithm chromosome c appropriate to M. It controls the aspects of chromosome in the algorithm's behavior. M is the set of "target methods." it defines the types of interest corresponding to M; it returns values of target methods in M for all types of receivers, parameters, and also for all primitive types that are the types of parameters to constructors of other types of interest.

A call description is an object representing one method call. The method call is used to constructed and run test cases under randomized unit testing in genetic algorithm. A test case is defined as the sequence of call descriptions, the test case generation takes the possible data as a solution space to search, and apply to GA approach to find for a good solution. The test case generation collects the number of lines covered by the test case. It finds the fitness function only on coverage and the chromosome would benefit for coverage of larger number of method calls and test cases in every new method call is the potential of coverage of more test code.

A set M of target methods and a chromosome c as inputs to construct and run test case. It begins by initializing and constructing value pools and runs a test case, and returns the test case. It uses method called try-Run-Method; it takes a method as input and then calls the method, returns a call description.

Step 1. Initialize the population and enter Step 2.
Step 2. Ranking the individuals using any ranking method and enter Step 3.
Step 3.Nowthe genetic algorithm in conjunction with the classification method is used to select the smallest subset of data from the above selected M values that gives maximum accuracy.
Step 3. Recombine individuals generating new ones and enter Step 4.
Step 4. Mutate the new individuals and enter step 5.
Step 5. If the stopping criterion is satisfied, STOP the process; otherwise, replace old individuals with the new ones restructure the population tree and return to Step 2.
Step 6. Finally presents a fitness function Fitness (x) = A(x) + P/N (x) to maximize the accuracy where for chromosome x.

4.2 ODC fault types classification

Fault types play a relevant role in this study, since they are of primary importance in both steps of the method. A software fault is a development fault that may cause a deviation from the expected (i.e., correct) program behavior. To evaluate techniques, a classification scheme is required to fault types. In this paper we adopt the well-known Orthogonal Defect Classification (ODC) to distinguish fault types.

- Assignment: values assigned incorrectly or not assigned.
- Checking: missing or incorrect validation of data or incorrect loop or conditional statements.
- Interface: errors in interacting components, modules, call statements, parameters list.
- Algorithm: It is an incorrect or missing implementation that can be fixed by re-implementing an algorithm or data structures.
- Function: affects amount of the code size and refers to capability that is either implemented incorrectly or not.

The classification comes from a preliminary collection of faults observed in the MVS (Multiple Virtual Storage) system, schematized in a level particularly close to the programming one. Since its introduction, the ODC has been adopted in several large projects for process tracking and improvement. Faults, in this scheme, are classified into non-overlapping (orthogonal) classes, in which the choice is uniquely identified according to the semantic of their fix, i.e., a fault is characterized by the change in the code that is necessary to fix it. It was conceived to keep the classification process simple, with little confusion, and few classes that are distinct and mutually exclusive.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

4.3 Performance evaluation

Then applied the second step of the procedure by characterizing the following testing techniques: Functional Testing, Statistical Testing, Robustness Testing and Stress Testing. In general, these techniques are conceived to pursue different goals, and their application aims at improving some specific aspect of the software being developed. In the context of this work, the main goal is to characterize their behavior with regard to the type of faults they are more prone to detect, and to figure out how their overall performance varies depending on different conditions. It is worth noting that the chosen techniques are intended for the system testing stage, hence after unit and integration testing.

To have a set of techniques able to cover different types of faults, we selected two techniques, namely, Functional and Statistical testing, for verifying if the system does what is required to do and two techniques, namely robustness and stress, for verifying that the system does not do what is not expected For system test stage, these are the most used ones.

Functional testing In Functional testing, engineers derive test cases from specifications and test the system against what it is required to do. It is often referred to as "specification-based testing" or "black-box testing".

Statistical testing is to improve reliability of the software, intended as the probability that the application does not fail during its mission time. Similarly to Functional testing, statistical testing does not require any knowledge of the code. The term Statistical Testing is used differently in the literature, depending on how the distribution is chosen, e.g., to satisfy an adequate criterion expressed in terms of functional or structural properties or selecting test cases by sampling from a distribution over the input domain reflecting the expected operational profile.

Robustness testing aims at evaluating the software application's reaction to exceptional and unforeseen inputs describe one of the most successful robustness testing techniques, and the corresponding tool: Ballista. This technique adopts a data-type based fault model to generate invalid inputs, and generates test cases by combining invalid parameter values of invoked function's data type.

Stress testing is a technique to evaluate the application's ability to react to heavy loads. Unlike robustness testing, stress testing does not use exceptional inputs; it uses normal input values, with excessive load. The goal is to observe the performance of the system under excessive loads, and to see if there are bugs in the management of these situations. It is particularly useful for long-running, mission and business critical applications, where the performance degradation.

4.4 Markov Decision Process (MDP)

Besides, the strategies implemented in the ODC classification technique have been suggested as a difficulty of switching among the techniques. For example, during the session of the testing, the initial approach would tend to have highest number of faults, and then the index techniques can be computed in a better way. So as an effective methodology, more complex strategies are adopted as stochastic models. MDP promotes the best set of actions that promises to yield maximum amount of the faults that could be discovered in the software application.

VI.CONCLUSION

The variations of the fault percentile have been depicted through the graphical representation based on their respective technique of the testing that has been utilized.

This technique allows mixing techniques so as to maximize the number of faults revealed for the software under test among those estimated to be present. Thus, the method provides engineers within an organization with a practical way for planning testing activities and efforts. As for fault types, the method refers to the well-known ODC classification; indeed, this work is the first one, to the best our knowledge, that relates testing techniques to ODC fault types and these to software metrics. As for testing techniques, we considered techniques of testing like functional, statistical, robustness and stress.

The method consists of two complementary steps. The first one is to construct, for the given application, a set of simple regression models to predict the number of faults it contains for the ODC fault categories of interest. This step requires an initial estimate of the faults number, which the testing engineer in an organization can derive in several ways, such as from previous versions of the software, or from other existing systems in the same product line, or in

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

similar product lines. For the purpose of showing how to construct such models, in this paper we used a set of applications from the technical literature, whose ODC fault types are known. The second step consists of a procedure to characterize testing techniques of interest in terms of ODC fault categories they are more prone to detect. It provides the basis for planning testing activities and efforts.

REFERENCES

[1] Domenico Cotroneo,Roberto Pietrantuono, Stefano Russo, (2013) "Functional Testing techniques using on ODC to identify and predict faults ",ELSEVIER Transactions on Software Testing, Vol 86, pp.1613-1637.
[2] Avritzer.A, Weyuker.E.J, (1996) "Metrics to assess the likelihood of project success based on architecture reviews", Empirical Software Engineering Journal,Vol4 ,pp. 197–213.
[3] Cai Kai, GuBo, Hu Hai, and Li Young,(2007) "Adaptive software testing with fixed memory feedback", Journal of Systems and Software,Vol80 ,pp.1328–1348.
[4] Catal.C, Diri.B.M, (2009)"A systematic review of software fault prediction studies. Expert Systems with Applications",Vol 36.
[5] Carreira.J, Madeira.H, Silva.J.G, (1995) "Xception: software fault injection and monitoring in processor functional units", IEEE Transactions on Software Engineering, Vol 24.
[6] Chillarege.R, Bhandari.I.S, Chaar.J.K, Halliday.M.J, Moebus.D.S, Ray.B.K, and Wong.M.Y, (1992) "Orthogonal defect classification – a concept for in-process measurements", IEEE Transactions on Software Engineering, Vol 18,pp.943–956.
[7] Christmansson.J,Chillarege.R,"Generation of an error set that emulates software faults. (1996) " In: Proceedings of the 26th IEEE Fault Tolerant Computing Sym-posium, pp. 304–313.
[8] Cotroneo.D, Natella.R, Pecchia.A, and Russo.S, (2009) "An approach for assessing logs by software fault injection".In: Supplemental Proceedings IEEE International Conference on Dependable Systems and Networks, pp. A15–A20.
[9] Dalal.S, Hamada.M, Matthews.P, and Patton.G, (1999)" Using defect patterns to uncover opportunities for improvement". In: Proceedings of the Intl. Conference Applications of Software Measurement.
[10] Do.H,Elbaum.S, and Rothermel.G, (2005) "Supporting controlled experimentation with testing techniques: an infrastructure and its potential impact", Empirical Software Engineering,Vol10 ,pp.405–435.
[11] Duraes.J.A,Madeira.H, (2006) "Emulation of software faults: a field data study and a practical approach", IEEE Transactions on Software Engineering,Vol 32, pp.849–867.
[12] Frankl.P.G, Hamlet.R.G, Littlewood.B, and Strigini.L,(1998) "Evaluating testing methods by delivered reliability", IEEE Transactions Software Engineering,Vol24 ,pp.586–601.
[13] Grottke.M, Li.L, Vaidyanathan.K, and Trivedi.K.S. (2006) " Analysis of software aging in a web server", IEEE Transactions on Reliability,Vol 55.
[14] Kallepalli.C, Tian.J,(2001) "Measuring and modeling usage and reliability for statistical Web testing",IEEE Transactions on Software Engineering,Vol 27, pp.1023– 1036.