Critical Evaluation of GUI Software Exception Testing

M.Rajalakshmi[#1], Dr. Viji Vinod [#2]

#1 Research Scholar
Department of Computer Science & Engineering
Dr.MGR Educational and Research Institute University
Chennai – 95, Tamil Nadu, India

#2 Professor & Head of the Department,
Department of Computer Applications,
Dr.MGR Educational and Research Institute University
Chennai – 95, Tamil Nadu, India

ABSTRACT

Demand on complex systems increased more rapidly. The size of complexity of computer systems has grown during the past decades in a very inspiring manner. Lots of work has been done on software exception testing and some of major critical evaluation have appeared in the literature are discussed in this paper. This paper also summaries the related developments of the same work. The main objective of this paper is to find the critical sections of software exception and evaluate those failure modes with effective analysis and level of evaluation. However, there is no single method that is handling to all the situations. So this paper also provides various ways to improve the evaluation to handle the critical sections with comparative study.

Key words: GUI software; exception testing; SFMEA; Evaluation Methods; Object- FMA

Corresponding Author: M.Rajalakshmi

INTRODUCTION

Graphical User Interface (GUI) is a program interface that takes advantage of the Computer's graphics capabilities to make the program easier to use. Graphical User Interface (GUI) provides user an immense way to interact with the software [13]. GUI testing is a process to test application's user interface and to detect if application is functionally correct. GUI testing involves carrying set of tasks and comparing the result of same with the expected output and ability to repeat same set of tasks multiple times with different data input and same level of accuracy. GUI Testing includes how the application handles keyboard and mouse events, how different GUI components and functions reacts to user input and whether or not it performs in the desired manner. In a GUI test suite, the combination of many test cases for the GUI application, it is often the case that certain actions will be repeated in succession in order to properly test the GUI. Implementing GUI software testing for your application early in the software development cycle speeds up development improves quality and reduces risks towards

the end of the cycle.

GUI Testing can be performed both manually with a human tester or could be performed automatically with use of a software program. Software testing is not only a common way to ensure the software quality [2], but also is a significant approach to control the software quality during the software development life cycle. In general sense, software testing contains the following two main tasks, one is to ensure that the software does the required work, another one is to ensure that software does not do the unexpected work that is usually validated by the exception testing. The validation results of the exception testing largely determine the software quality. The more sufficient exception testing is, the residual defects will be less and the software quality will be better. However, currently, the exception testing lacks of a mature approach for generating the testing cases and commonly utilizes the error-guessing approach. Although the error-guessing approach might be useful, it is rather arbitrary, time consuming, expensive and does not cover the frequent accidents sufficiently [3]. An object-based approach for the failure mode analysis (Object-FMA) also using in this paper.

The Object-FMA approach provides a systematic method for the failure mode analysis, and makes the failure mode analysis more comprehensive. This paper utilizes this Object-FMA approach to analyze the failure modes of the common functions in Windows, and generate the database of the failure modes of the functions for guiding to design test cases. The exception test cases generated by the Object-FMA approach not only are more sufficient than the ones generated by the error-guessing approach, but also detect more exceptions. This proposed approach can avoid to overreliance on the experience of testers during designing the exception testing cases. Moreover, this approach can ensure the quality of the exception testing cases from the methodological viewpoint. Thus, the feasibility and validity of this proposed Object-FMA approach are validated. Exception testing is defined as a complete and honest effort to BREAK the system. With exception testing, the team tries to dream up every possible way that users will run the system and make sure that: 1: The system does NOT crash.2: The system is able to stop bad data. 3: The system reports the root cause of the problem in an understandable way and suggests possible ways of fixing it.

Rest of the paper is organized as follows: section 2 of the paper describes related developments. Section 3 describes implementation of critical evaluation of GUI software exception testing. Finally, the paper concludes with future work direction in section 4.

RELATED DEVELOPMENTS

Isabella and Emi Retna proposes a technique that can be used for GUI Testing has become very important as it provides more sophisticated way to interact with the software. The complexity of testing GUI increased over time. The testing needs to be performed in a way that it provides effectiveness, efficiency, increased fault detection rate and good path coverage. To cover all use cases and to provide testing for all possible (success/failure) scenarios the length of the test [1].Penelope A. Brooks and Atif M Memon proposes a technique that can be used for regression testing of GUI applications, i.e., information gathered from usage of the current version of a GUI application is used to determine whether the new application perform well.

Alexander K. Ames and Haward Jie Thus these critical paths become important to choosing which unit tests to implement. Performing unit testing based on our data reduces the time to implement these tests. This is an improvement over to implementing the tests alone, which takes longer and does not always test the critical paths [8].

Xun Yuan, Member, IEEE, Myra B. Cohen proposes new criteria range in both efficiency (measured by the size of the test suite) and effectiveness (the ability of the test suites to detect faults). Our study shows that by increasing the event combinations tested and by controlling the relative positions of events defined by the new criteria, we can detect a large number of faults that were undetectable by earlier techniques [13]. Jing Lai, Hong Zhang, Baiqiao Huang were introduces the SFMEA (Software Failure Mode Effect Analysis) to generate the exception testing cases for the GUI software by analyzing the failure modes of the controls and the control sets of the GUI software and then translating those failure modes directly into the exception test cases. In order to make the failure mode analysis sufficient, they first proposed an object-based approach for the failure mode analysis (i.e. Object-FMA) [14].

Gurram.Saritha dharma. Lakshmi Padmaja presents the implementation of graphical user interface and comparative study is more efficient to existing work, systems explore the impact of test criteria proposed analysis can detect a large number of faults that were undetectable [7]. Abdul Rauf, Arfan Jaffar and Arshad Ali Shahid attempted to exploit the event driven nature of GUI. Based on this nature, presented a GUI testing and coverage analysis technique centered on genetic algorithms [9]. Basili,V.R. Selby,R.W proposed an experimentation methodology to compare three state-of-the-practice software testing techniques: a) code reading by stepwise abstraction, b) functional testing using equivalence partitioning and boundary value analysis, and c) structural testing using 100 percent statement coverage criteria [17].

Andrea Adamoli, Dmitrijs Zaparanuks, Milan Jovic, Matthias Hauswirth were proposed a Performance testing imposes additional requirements upon GUI test automation tools set of case studies that the concluded about perceptible performance drawn from manual tests still hold when using automated tests driven by Pounder. Besides the significance of our findings to GUI performance testing, the results are also relevant to capture and replay-based functional GUI test automation approaches [18].

CRITICAL EVALUATION OF GUI SOFTWARE EXCEPTION TESTING

Features of GUI Software Testing

Graphical user interface (GUI) testing is the process of testing software's graphical user interface to safeguard it meets its written specifications and to detect if application is working functionally correct. GUI testing involves performing some tasks and comparing the result with the expected output. This is performed using test cases. GUI Testing can be performed either manually by humans or automatically by automated methods.

The GUI software technology provides the intuitive interfaces between the users and the software. Therefore, it is widely used in commonly used software [4]. It even occupies a large

part of the codes in the software, and is very significant to the software engineering. As a result, this paper selects the approach for generating the GUI exception test cases as the research object. In order to solve this problem, the SFMEA[5][6] (i.e. Software Failure Mode Effect Analysis's introduced to analyze the failure modes of the functions and interactions between the functions satisfying the special constraint relationships in the GUI software, and translate these failure modes directly into the exception test cases.

The process of testing a GUI application calls for a colossal effort, owing on account of the complexity entailed in such applications. Subsequently, organizations were spurred to initiate the automation of GUI testing, thereby proposing various techniques to achieve this end. A GUI model event-flow graph, an innovative technique being utilized in the field of automated GUI testing, represents, likewise control flow graph, all promising progressions of events that can be executed on GUI. Graphical user interface ripping and the construction of process flow graph and event interaction graphs include all the elements and events that can be found in a graphical user interface once the module is built the process generates automatically all the possible functional test cases.

Proposed Approach

This paper proposed a critical evaluation of GUI software exception testing that is to find out the Goal of testing and critical sections in all software exception situations and evaluate depends upon the test results and expected results using various methods. First it takes place rules and assumptions of failure mode, identification of failure mode, and then failure effects to be analyzed after describe different methods for failure detection and track and rank the failure modes finally, evaluate the critical sections

System Implementation

Goal of Testing

Process of creating a program consists of the following phases 1. defining a problem; 2. designing a program; 3. building a program; 4. analyzing performances of a program, and 5. final arranging of a product. According to this classification, software testing is a component of the third phase, and means checking if a program for specified inputs gives correctly and expected results. Software testing (Figure 1) is an important component of software quality assurance, and many software organizations are spending up to 40% of their resources on testing. For life-critical software (e.g., flight control) testing can be highly expensive.

Testing is an activity performed for evaluating software quality and for improving it. Hence, the goal of testing is systematical detection of different classes of errors (**error** can be defined as a human action that produces an incorrect result [15]. in a minimum amount of time and with a minimum amount of effort. We distinguish

- Good test cases - have a good chance of finding an yet undiscovered error; and
- Successful test cases - uncovers a new error.

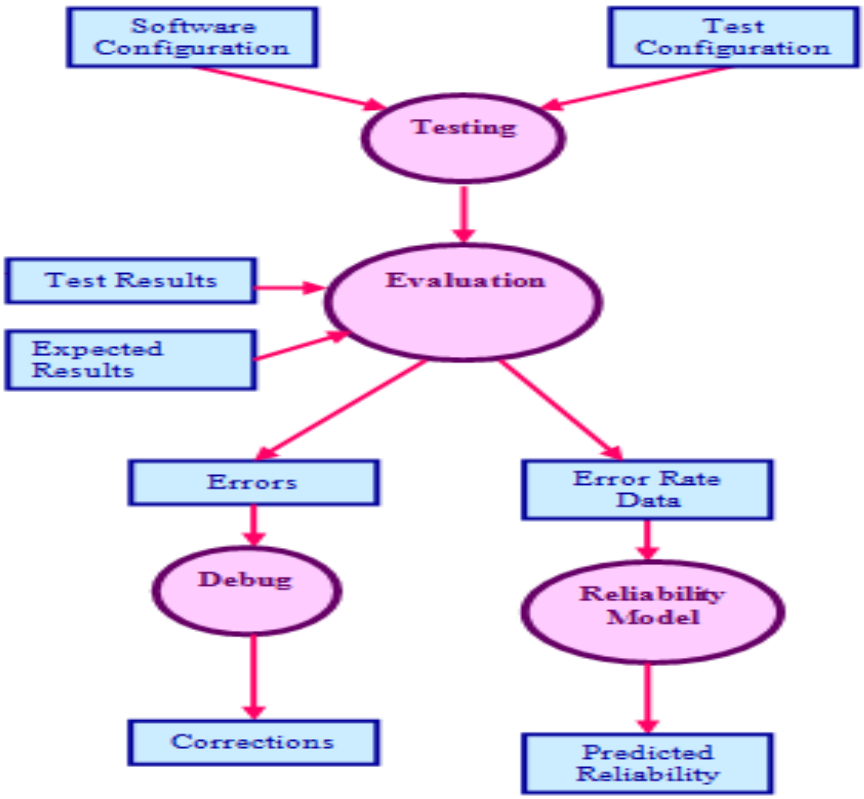

Fig1: Test Information flow

Anyway, a good test case is one which:

- Has a high probability of finding an

Error; Is not redundant;

- Should be "best of breed";
- Should not be too simple or too Complex [16].

Testing is which finds the defects/ bugs and produces the quality assurance product. For each product in the role of testing it finds functionality to meets the user requirements before it goes to live. Graphical user interface is frontend tool which raises the so many defects in any domain to avoid such bugs.

Rules and Assumptions

Before detailed analysis, ground rules and assumptions are usually defined and established to, for example:

- Standardized mission profile with specific fixed duration mission phases
- Sources for failure rate and failure mode data
- Fault detection coverage that system built-in test will realize
- Whether the analysis will be functional or piece part
- Criteria to be considered (mission abort, safety, maintenance, etc.)
- System for uniquely identifying parts or functions
- Severity category definitions.

3 Identification of Failure mode

For each function covered by the analysis, a complete list of failure modes is developed. Thus failure modes include:

- Untimely operation
- Failure to operate when required
- Loss of output
- Intermittent output
- Erroneous output (given the current condition)
- Invalid output (for any condition)

For piece part FMECA, failure mode data may be obtained from databases. These databases provide not only the failure modes, but also the failure mode ratios (table 1). For example: Each function or piece part is then listed in matrix form with one row for each failure mode. Because Failure mode analysis usually involves very large data sets, a unique identifier must be assigned to each item (function or piece part), and to each failure mode of each item.

Table 1

Device Failure Modes and Failure Mode Ratios (FMD–91)

Device Type	**Failure Mode**
Relay	Fails to trip
	Spurious trip
	Short
Resistor, Composition	Parameter change
	Open
	Short

4 Analysis of Failure Effects

Failure effects are determined and entered for each row. And considering the criteria identified in the ground rules. Effects are separately described for the local, next higher, and end (system) levels. System level effects may include:

- System failure
- Degraded operation

- System status failure
- No immediate effect

The failure effect categories used at various hierarchical levels are tailored by the annalist using engineering judgment.

5 Software Evaluation Criteria

The Software Sustainability Institute provides a software evaluation service based on two complementary approaches developed over many years in the research software arena. The service can help you to improve your software. It can assess the general usability, and can identify technical or development issues, as well as any barriers to sustainability. The two approaches makes more sense than the other. One is criteria-*based* approach is a quantitative assessment of the software in terms of sustainability, maintainability, and usability. This can inform high-level decisions on specific areas for software improvement.

This approach forms the basis of our online sustainability online sustainability evaluation, a web-based assessment you can use straight out of the box. Second *tutorial-based* approach provides a pragmatic evaluation of usability of the software in the form of a reproducible record of experiences. This gives a developer a practical insight into how the software is approached and any potential technical barriers that prevent adoption. Evaluations of software products must be objective - based upon observation, not opinion.

Table 2: Defines the definition of evaluation process.

	Process for developers	**Process for Acquirers**	**Process for Evaluators**
Analysis	definition of quality requirements and analysis of their feasibility	establishing purpose and scope of evaluation	describing the objectives of the evaluation
Specification	quantification of quality requirements	defining the external metrics and corresponding measurements to be performed	defining the scope of the evaluation and the measurements
Design	planning of evaluation during development	planning, scheduling and documentation of evaluation	documenting the procedures to be used by the evaluator
Execution	monitoring of quality and control during development	evaluation shall be performed, documented and analyzed	obtaining results from performing actions to measure and verify the software product

Some criteria's, presented shortly as *Scope:* What items are included in the software?

Breadth: Are all aspects of the software documentation covered? *Depth:* To what level of detail of information provided does the software go? *Time:* Is the information in the resource limited to certain time periods? *Format:* Are certain kinds of Internet resources (for example FTP) excluded? *Content:* Is the resource an integral resource updated by the information original source? Some Specific aspects related to the content include the accuracy, authority, currency and uniqueness of a resource. *Accuracy:* Is the information in the resource accurate? The information is placed to advertise, or support a particular point of view. *Authority:* Does the resource have some reputable organization or expert behind it? Are sources of information stated? *Uniqueness:* Is the information in this resource available in other forms (for example other sites, print, CD-ROM)? Does it complement another resource, for instance by providing updates to a print source? *Links made to other resources:* Are the links made in such a way that it is clear that an external site is being referred to. *Quality of writing:* Is the text well written? The quality of writing is important for the content to be communicated clearly. *Graphic and multimedia design:* Is the resource interesting to look at? If audio, video, virtual reality modeling, etc. are used, are they appropriate to the purpose of the source? *Purpose:* What is the purpose of the resource? Is this clearly stated? Does the resource fulfill the stated purpose? *Audience:* Who are the intended users of this resource? *Workability:* Is the resource convenient and effective to use? This is the area where criteria for Workflow software resources differ most from print sources. Some Aspects of workability include: *User friendliness*: Is help information available? Have user interface issues been addressed, such as menu design, readability of screens, etc. Required computing environment: Can the resource be accessed with standard equipment and software? *Searching*: How effectively can information be retrieved from the resource? *Browsability and organization*: Is the resource organized in a logical manner to facilitate the location of resources? Is the organizational scheme appropriate, for example chronological for historical source, or geographical for a regional resource? *Interactivity:* Where interactive features (forms, cgi scripts) are provided, do these work? Connectivity: Can the resource be accessed with standard equipment and software, or are there special software, password, or network requirements? Can the resource be accessed reliably? *Cost:* Currently the costs of Internet information resources become important. Costs can be divided into: costs of connecting to the resource and (b) costs associated with the use of the intellectual property contained in the resource. In terms of (a), users paying traffic charges are already having to consider the costs of connection, and may want to include this in criteria for selection. Software evaluation is multi-criteria decision making problem that refers to making preference decisions over the available alternatives. We found that AHP (Analytical Hierarchical Process) has been widely used for evaluation of the software packages. AHP was developed by Saaty [19] and has been identified as an important approach to multi-criteria decision making problems of choice and prioritization. Another technique used for evaluation of software package is the weighted scoring method. In this method weights and rating scales are assigned to each criterion. The weight reflects the relative importance of each of the criteria while the rating scale indicates how easily each package is able to meet the specific criterion. The rating scales are then multiplied by weight factor of each criterion. Using this scheme a score is calculated for every criterion for each tool. These scores are then totaled to produce a score for each criteria category. Finally, the categorical scores are combined to calculate an overall tool score. A fuzzy based approach for software evaluation has been used as performance rating and weights cannot be given precisely.

In such cases the fuzzy set theory is used to model the uncertainty of human judgments and such problem is known as fuzzy multiple criteria decision making (FMCDM).software

critical revolve around two important concepts.
1. Critical software must be developed and tested with constant consideration of the "system". 2. A standard, planned, and documented process carried out by responsible and knowledgeable humans is a requirement for success.

6 Concepts of Evaluation Assurance Level

The intent of the higher levels is to provide higher confidence that the system's principal security features are reliably implemented. The EAL level does not measure the security of the system itself, it simply states at what level the system was tested. Although every product and system must fulfill the same assurance requirements to achieve a particular level, they do not have to fulfill the same functional requirements. The functional features for each certified product are established in the Security Target document tailored for that product's evaluation. An evaluation level defines the depth or thoroughness of the evaluation in terms of evaluation techniques to be applied and evaluation results to be achieved, both with reference to evaluation objectives. Example of evaluation objectives is safety conditions, security constraints, economic risk, availability conditions, and application constraints. As a consequence evaluation at different levels gives different confidence in the quality of the software product and independent level of software critical. The evaluation may be augmented to include assurance requirements beyond the minimum required for a particular EAL. Officially this is indicated by following the EAL number with the word augmented and usually with a list of codes to indicate the additional requirements.

CONCLUSION AND FUTURE WORK

Software is an immovable mechanism that consist of computer programs, procedures, rules, data related documentation. The increase in the number of software failures badly affected the performance of transposition, telecommunication, and military, industrial process etc. which made software critical evaluation more & more important. This study provides testing goals, identification and analyzes of the modes of failure .Further, it provides the criteria for the software evaluation and evaluation level. Overall, we conclude that software exception testing using analytical effects is critical and it improves the GUI software testing. Our future goals are for functions and interactions between the functions to be evaluated using various evaluation methods for GUI software exception testing.

REFERENCES

[1] Isabella and Emi Retna Study Paper on Test Case Generation for GUI Based Testing International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012

[2] GJB-Z 141-2004[M].

[3] Daboczi T, Kollar I., Simon G, Megyeri T. How to Test Graphical User Interfaces[J]. IEEE Instrumentation and Measurement Magazine,2003, Vol(6) No.3_27-33.

[4] A.M.Memon.A Comprehensive Framework For Testing Graphical User Interfaces[D]. Department of Computer Science, University of Pittsburgh, 2001.

[5] Peter L. Goddard. Raytheon. Troy. Software FMEA

Techniques[C].Proceedings Annual Reliability and Maintainability Symposium 2000:118-123.

[6] Nathaniel Ozarin,Michael Siracusa.A Process for Failure Modes and Effects Analysis of Computer Software[C].Annual Reliability and Maintainability Symposium,2003:365-370.

[7] Gurram.Saritha [1]dhyaram. Lakshmi Padmaja Software Testing Approach – Survey on GraphicalUser Interface International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume1 Issue 2 Nov 2012 Page No. 85-93.

[8] Alexander K. Ames and Haward Jie Critical Paths for GUI Regression Testing Univ. of California, Santa Cruz.

[9] Abdul Rauf, Arfan Jaffar and Arshad Ali Shahid Fully Automated GUI Testing and Coverage Analysis using genetic algorithms International Journal of Innovative Computing, Information and Control Volume 7, Number 6, June 2011.

[10] Huang Baiqiao, Zhang Hong, Lu Minyan,etc. Test Data Selecting Strategy for GUI Software Test[J]. Journal of Computer Research and Development, 010_47(Suppl.),101-107.

[11] Wei Dong-mei,Hong Mei,Li Bo. A Regression Test Optimization Based on "Good" GUI Test uite[J],Computer Technology and Development.2008,Vol(18),No.7:1-4.

[12] Zhang Bo-feng,Zou Chang-qing _ Generation of interaction diagram for GUI software testing and its implementation _ Application Research of Computers, 2007,Vol(24),No.11:222-235.

[13] Xun Yuan, Myra B. Cohen, Atif M. Memon, (2010) "GUI Interaction Testing: Incorporating Event Context", IEEE Transactions on Software Engineering, vol. 99.

[14] Jing Lai, Hong Zhang, Baiqiao Huang The Object-FMA Based Test Case generationApproach for GUI Software Exception Testing 2011 IEEE.

[15]http://www.testingstandards.co.uk/living_glossary.htm# Testing, February 08, 2009.

[16] Stacey, D. A., "Software Testing Techniques".

[17] Basili,V.R. Selby,R.W. Comparing the Effectiveness of Software Testing Strategies 2006 IEEE.

[18] Andrea Adamoli, Dmitrijs Zaparanuks, Milan Jovic, Matthias Hauswirth Automated GUI erformance testing Software Quality Journal December 2011, Volume 19, Issue 4, pp 801-839.

[19] T.L. Saaty, The Analytic Hierarchy Process, McGraw-Hill, New York, 1980.

[20] http://www.software.ac.uk/software-evaluation-guide.