Critical Components Identification for Effective Regression Testing

M. Ramalakshmi Praba, D. Jeya Mala

Assistant Professor – Dept.of MCA, KLN College of Information Technology, Madurai
Associate Professor, Dept.of Computer Applications, Thiagarajar College of Engineering, Madurai.

Abstract—Regression testing is to check program correctness after it was changed. But during regression testing, due to the stopping criteria followed by industries, some of the critical components and their dependent components might have been missed. This leads to catastrophic failure in terms of cost, time and human life. To address this most important and critical problem this paper proposes a novel method to identify the critical components and prioritize them for testing based on their dependency and complexity metrics before the software is delivered to the customer side.

Keywords: Software Testing, Regression Testing, Component based Testing, Critical component, Metrics.

1. Introduction

Testing is the one of the ways of assuring the quality of the product. According to 40-20-40 rule, software development consumes 40% of total time for project analysis and design, 20% for programming and rest of 40% for testing [17]. Hence better testing methodology should be followed by the industries for producing better product.

Component based system development is desired by the industries because of its flexibility, reusability, extensibility etc.**,** Even though the industries followed better testing methodology and produce quality product, the customer may return back the product to the industry for feature enhancement or modification of the existing functionality or for defect fixing. After changing the product, based on the customer's requirements, the product has to be tested. This type of testing which is known as regression testing, consumes significant portion of development and maintenance costs [19]. Regression testing is an important but expensive way to build confidence that software changes introduce no new faults as software evolves [20]. In reality, the industries skip testing some components during regression testing, in order to manage the release schedule and cost. Now, the problem occurs if some of these skipped components are critical components which have their impact or side effect on other components. One solution is to test potentially risky components or critical components rigorously during regression testing prior to other components in the system.

This paper proposes a novel method to identify the critical components being tested rigorously using known metrics and measures. Also, the proposed regression testing method identifies the dependent components of each changed component. Then prioritization takes place during regression testing, which will reduce the threats related to the critical components.

2. Related Work

Jerry GAO [10, 11], proposed a model to measure the maturity levels of a component testing process.

According to McGregor [12]. All the components were classified according to three risk categories and components falling in one category were tested at the same coverage level. But exact quantification of the risks associated with each component is not possible using this technique and it fails to give an account of number of most critical components that need to be tested.

Jeya Mala et.al. [13] Proposed a technique for optimizing the test cases to improve the efficiency of the testing process using various coverage metrics.

Srivastava [22] suggested prioritizing test cases according to the criterion of increased APFD and proposed a new algorithm which could be able to calculate the average number of faults found per minute by a test case and using this value to sort the test cases in decreasing order.

Rothermel et al [23], have described several techniques for test case prioritization and empirically examined their relative abilities to improve how quickly faults can be detected by those suites. The objective is to detect faults as early as possible so that the debugger will not sit idle.

Mao and Lu [20] proposed a testing method; Component developers should calculate the change information from labeled method call graph and provide it to component users via XML files. Component users use this change information and their instrumentation records together to pick out test cases for next-round testing.

Malishevsky et al [21] proposed cost models that would help them assess the cost-benefits of techniques. The cost-benefits models were used for regression test selection, test suite reduction, and test case prioritization.

Jeya Mala et.al.[24,25] Proposed the metrics for critical component identification.

The dependency based test prioritization improves the early fault detection when compared to traditional test prioritization as well as total number of fault detection. The experiments result suggested that quality of a system can be improved in terms of effectiveness using test prioritization.

3. Problem Formulation

A component based system consists of 'n' number of components and, most of the components are dependent on each other. During regression testing, the verification and validation of a component based system is a tricky task, because testing all the components with all possible inputs is a challenging one. The main challenge is to identify and test the components that are critical for the overall working of the system. Also, the testers should know about the information of the modified component to identify those components which are dependent on the modified component. Hence, the research problem here is to find out the dependent components of each of the modified components and locating potentially risky or highly critical components among the dependent components and finally prioritize them during regression testing.

In this research work, the component based system (CBS) is represented by means of a specific graphical representation called as Component Execution Sequence Graph (CESG). This graph is a network representation of the CBS and it consists of nodes to represent the components and edges. Figure1 is a typical Component Execution Sequence Graph G which contains five nodes, N (G) = {A, B, C, D, E} With Edges, E (G) = {i, j, k, l, m}

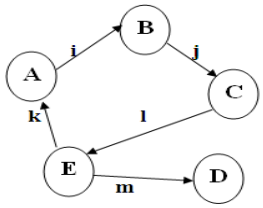

Figure 1. Component Execution Sequence Graph

A. Critical Value Calculation

The critical value for each component is calculated as the summation of a specific class of metrics. The selection of such metrics focuses on identifying the critical components. They are classified as external metrics and internal metrics. The external metrics shows the dependence value of the modified component quantitatively and are derived from the dependence attributes of the components such as

1) Fanin, 2) Fanout and 2) Coupling between the Objects.

The internal metrics shows the potential complexity value of each component. The internal metrics are

1) Weighted Methods per Class (WMC), 2) Lack of Cohesion of Methods (LCOM) ,
3) Number of static methods (NSM), 4) Depth in Tree(DIT), 5) Number of static Attributes (NSA),
6) Number of Children (NSC), and 7) Method lines of code (MLOC).

Metrics and their definitions are shown in Table I.

4. Proposed Approach for Effective Regression Testing

A. Proposed Framework

The proposed framework is shown in Figure2. In this framework, the given software under test (SUT) is analyzed and the components are extracted from it. For each component, the proposed component prioritize module calculates the external metric values with respect to the modified component. Based on these values, the dependent component list for each modified component is prepared.

Then the Internal metric value for each component in the dependent component list is measured. After that the total critical value for each component is calculated as the sum of internal metric values and external metric values. The prioritizer module then prioritizes the components based on their criticality value and the final list will be generated for effective testing. These component lists along with their test cases are kept in the regression test database (RTDB). This module also provides the provision for visual representation of critical components as Component Execution Sequence Graph (CESG). From the visual representation, the tester can easily identify the dependent components. So he can easily choose the suitable test cases for rigorous testing.

5. Experimental Setup and Result Analysis

For identifying the critical component list, the class files are necessary for each component. To calculate the various metrics, the Java Byte code Analysis is applied. The class files for Software under Test (SUT) are generated by using Java compiler. This compiled format is not in the human readable format. Hence, from the class file, the Oolong file was created, in this research work. Oolong is an assembly language for the Java Virtual Machine (JVM), it is nearly equivalent to the class file format but in the human readable form. For each component, the Oolong instructions are analyzed and then the proposed component prioritizer module calculates the External metric value and generates the dependent component List. The Internal metric values and the external metric value for each dependent component are measure to identify the critical components and they are prioritized based on that value.

A range of case studies are taken from the online project libraries such as (1000projects.org, www.itprojectsforyou.com, www.javaworld.com) for effective regression testing. These case studies are varied in its number of classes and Lines of codes. Each case study is analyzed and the proposed metrics were measured. The Experiment result shows that, time taken for proposed metric calculation is very tiny, when compare with overall time taken for testing all the components.

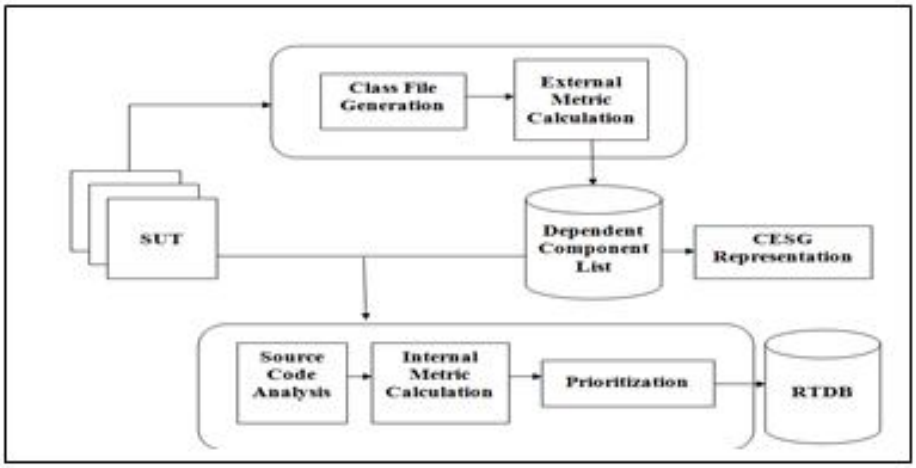

Figure2. Framework for Critical Component Prioritization

Table I: Metrics and its definition

Metrics	Description	Definition
Fanin (M1)	The number of other classes that reference a class.	Fanin = number of other classes that reference a class
Fanout (M2)	The number of other classes referenced by a class	Fanout = number of other classes referenced by a class
CBO (M3)	Coupling between the objects	$\text{Count cp} = \sum_{i=0}^{n} Ii + \sum_{j=0}^{m} MINVj$ Where $M_{INV} = \sum_{i=0}^{m} M_i \times [1 + Arg_m \times 0.046]$ $\text{Cob} = \frac{k}{\text{Count cp}}$ Where k = 1 and is a proportionality constant which may be adjusted as experimental verification [18] M_i Method I, Arg_m is the Argument of method.
WMC (M4)	Weighted Methods per Class	Sum of the McCabe Cyclomatic Complexity for all methods in a class
LCOM (M5)	Lack of Cohesion of Methods.A measure for the Cohesiveness of a class.	m= number of procedures (methods) in class a= number of variables (attributes) in class mA= number of methods that access a variable (attribute) LCOM2 = 1 - sum(mA)/(m*a)
NSM (M6)	Number of static methods	NSM= Number of static methods
DIT (M7).	Depth in Tree. Distance from class Object in the inheritance hierarchy.	DIT = maximum inheritance path from the class to the root class
NSA(M8)	Number of static Attributes	NSA= Number of static attributes
NSC(M9)	Total number of direct subclasses of a class.	NSC = number of immediate sub-classes of a class
MLOC (M10)	Method lines of code MLOC	MLOC = number of non-blank and non-comment lines inside method

A. Case Study

For the first case study, 'Vehicle Management System' is taken. It is application software. It consists of thirty components and 5511 lines of codes.

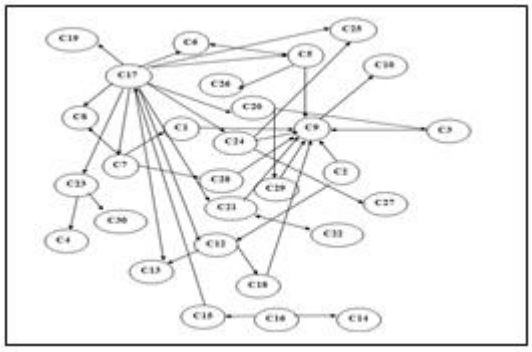

Figure 3. Component Execution Graph of the Vehicle Management

To calculate the proposed metrics, initially all the components are identified in the 'Vehicle Management System'. For all the changed components, external metrics are assessed. Then for each dependent component the internal metrics are calculated. Each component is assigned a weight as the sum of external and internal proposed metrics called as criticality value. The TABLE II shows the Vehicle Management System project's components and their corresponding criticality value. Using this value, the priority value is assigned to each component. Then each component is tested based on this priority value which thus helps in rigorous testing of components without missing any of the critical components. The CESG for the Vehicle Management System shown in Figure 3.

Table II: Metric values for vehicle management System

Component Name	Fanin	Fanout	CBO	NSM	NSF	NSC	MLOC	DIT	LCOM	WMC	Total	Priority
AddEntry	1	1	0.58	0	2	0	139	5	1.19	9	158.761	14
AddNewEntry	1	1	1.53	0	2	0	293	5	1.00	27	332.532	3
AddPassenger	1	1	0.58	0	0	0	194	5	0.98	14	216.555	11
AddRoute	0	1	0.00	0	2	0	223	5	1.05	26	258.045	8
Booking	3	1	0.67	0	0	0	368	5	0.89	33	411.551	1
Booking_report	0	2	0.00	0	1	0	79	5	1.15	6	94.154	22
Buses	3	1	1.43	1	2	0	171	5	0.91	12	197.343	13
Bus_Details	0	2	0.00	0	4	0	84	5	1.14	6	102.143	20
DateChooser	0	1	0.00	0	10	0	153	6	0.85	48	218.850	10
Employee	3	2	0.96	1	5	0	127	5	0.00	7	150.956	15
employee_report	0	2	0.00	0	1	0	85	5	1.00	8	102.000	21
LoginScreen	1	1	0.33	0	0	0	95	6	0.75	8	112.083	19
Main	2	0	2.00	1	0	0	4	1	0.00	3	13.000	26
MDIWindow	12	1	11.47	0	1	0	242	6	0.85	18	292.324	5
NewEntry	1	1	0.55	0	2	0	338	5	0.82	27	375.366	2
NewUser	0	1	0.00	0	0	0	111	5	0.00	11	128.000	18
Passengers	2	1	0.96	1	5	0	109	5	0.00	8	131.956	17
Payment	2	2	1.24	0	0	0	218	5	0.97	16	245.205	9
Route	2	1	0.48	1	5	0	114	5	0.00	10	138.478	16

Schedule	3	1	1.04	0	2	0	263	5	0.91	21	296.950	4
Scheduling_report	0	2	0.00	0	1	0	77	5	1.14	6	92.143	23
Show_Booked	0	1	0.00	1	5	0	66	5	0.00	6	84.000	25
Show_schedules	0	1	0.00	1	5	0	73	5	0.00	6	91.000	24
UpdateEntry	1	1	0.28	0	1	0	193	5	0.55	9	210.826	12
UpdatePass	1	1	0.28	0	0	0	250	5	0.88	14	272.156	7
UpdateRoute	0	1	0.00	0	1	0	242	5	0.70	25	274.700	6

In the case study, 'Vehicle Management System', the components Schedule, Booking, Employee, passenger, payment are taken for modification by means of the defect injection in the components as per the Offutt's [9] mutant guidelines method. The components which are dependent on the modified component are identified using the external metric value associated with the modified component. Then the internal metric value for each component in critical component list is calculated. Based on this value the components are prioritized. The priority value is called as the critical value and the dependent components are listed as critical component test based on their critical value.

a. Comparison with Existing Approaches

To analyze the efficiency of the proposed approach the existing two basic regression testing methods such as Full Regression testing and Unit Regression testing are applied. In the Full Regression testing method all the components in the software, are tested. In the unit regression testing method only the modified component is tested.

During the application of each of the method, the time taken to reveal the defect is calculated. TABLE III shows time taken by Basic Regression testing methods and the proposed regression testing method. It is depicted in Figure 4. The following inferences have been made from the critical values.

As full Regression testing method tests all the components in the software, it takes long time to complete the testing. Unit Regression testing method takes very little amount of time because it focuses only on the modified component. In the proposed regression testing technique based on critical component identification, the focus is not only on the modified component but also on the dependent component. During the dependent components testing, the critical components are identified and they are tested with higher priority than the other. And comparatively it takes more time than unit regression testing, and less time than Full Regression testing. Even though the time complexity shown in TABLE III indicates the Unit Regression testing takes less time, it is not a reliable one as the dependent components of the modified components or the components which are being dependent by the modified components will not be covered by it.

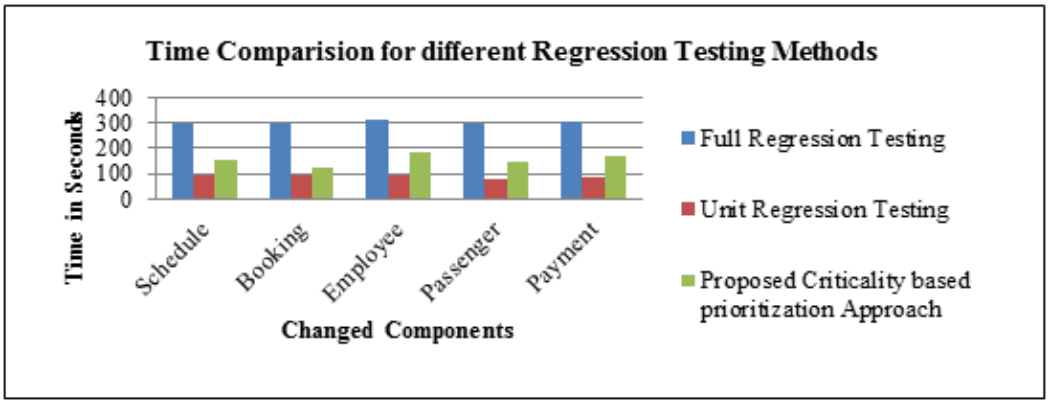

Figure 4. Time Comparison for different Regression Testing Methods

Table III: Time Taken by various Regression testing and percentage of error free

S.No	Defect No.	Defect Injected Component	Time Taken by various Regression Testing(in Sec) and % of Error free in terms of Requirement satisfaction in the total system					
			Full Regression Testing		Unit Regression Testing		Proposed Criticality based prioritization Approach	
			Time Taken (in Sec)	% of Requirement Satisfaction	Time Taken (in Sec)	% of Requirement Satisfaction	Time Taken (in Sec)	% of Requirement Satisfaction
1	Defect#1	Schedule	300.23	100%	92.1	93%	155.63	100%
2	Defect#2	Booking	298.26	100%	90.3	90%	123.6	100%
3	Defect#3	Employee	315.71	100%	91.2	75%	137.89	99%
4	Defect#4	Passenger	299.65	100%	81.18	78%	145.21	100%
5	Defect#5	Payment	302.68	100%	85.3	80%	150.34	100%

This may yield negative results during its execution. Hence, based on the analysis the proposed regression testing has been identified as a better method to yield reliable results for retesting. The above three Regression testing methods are applied in ten different projects. For each projects, three components are modified. For each component testing, the time taken for the Full Regression testing, Unit Regression testing methods and Proposed Regression testing methods is noted. In all the case studies takes less time for proposed regression testing method when compared with time taken for full regression testing method.

Conclusion and Future Work

In the proposed method, initially component's dependency is measured and critical components are identified. Then its criticality value is calculated for each dependent component and components are prioritized based on the critical value. Efficiency of the above method is confirmed by ten projects. The future work plans to provide some more dependency factors in the analysis of large systems and provide the visualization tool that helps the testers.

Acknowledgment

This work is the part of UGC research project supported by University Grants Commission, New Delhi, India.

References

[1] Thomas Zimmermann,, Nachiappan Nagappan,, Kim Herzig , Rahul Premraj and Laurie Williams "An Empirical Study on the Relation between Dependency Neighborhoods and Failures", In the proceedings of 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation

[2] Renee C Brcyce, Sreedevi Sampath, Atif M Memon "Developing a Single Model and Test Prioritization Strategies for Event Driven Software" IEEE Transaction on software Engineering, Vol. X, no X, January 2010.

[3] P. K. Suri, Sandeep Kumar "Simulator for Identifying Critical Components for Testing in a component Based Software System", IJCSNS International Journal of Computer Science and Netwok Security Vol. 10, no 6,June 2010.

[4] Katerina Goseva - Popstojanova "Guest Editors' Introduction to the Special Section on Evalution and Improvement of Software Dependability" IEEE Transaction on software Engineering, Vol.36, no 3,May/June 2010.

[5] Mariani, L., et al, "Compatibility and Regression Testing of COTS- Component Based Software," In the proceedings 0f 29th IEEE conference on Software Engineering, 2007, pp. 85-95.
[6] Xiaofang Zhang; Changhai Nie; Baowen Xu; Bo Qu "Test Case Prioritization Based on Varying Testing Requirement Priorities and Test Case Costs" In the proceedings of Seventh IEEE Conference on Quality Software (QSIC 2007)
[7] Jasmine K. Sl Dr. R. Vasantha "Identification of software performance bottleneck components in Reuse based software products with Application of Acquaintanceship Graphs. In proceedings of the IEEE conference on Software Engineering Advances (ICSEA 20007)
[8] www.Projectparadise.com
[9] Jingyu Hu, Nan Li and Jeff Offutt "An Analysis of OO Mutation Operators" In the proceedings of 24th Annual International Computer Software and Application Conference, Taipei, Taiwan, Oct 2000.
[10] Zheng Li, Mark Harman, and Robert M. Hierons "Search Algorithms for Regression Test Case Prioritization" IEEE Transaction on Software Engineering, April 2007.
[11] Gao, J., "Testing Coverage Analysis for Software Component Validation," In the proceedings of 29th Annual International Computer Software and Applications Conference, Edinburgh, Scotland, July 26-28, 2005.
[12] McGregor, J.D., "Component Testing," Journal of Object Oriented programming, Vol. 10, No. 1, 1997. pp. 6-9.
[13] D. Jeyamala, V. Mohan, M. Kamalapriya, "Automated Software Test Optimization Framework - an Artificial Bee Colony Optimization based Approach", International Journal - IET - Software ,Vol.4, No.5, pp.334-348, 2010
[14] OO Design Quality Metrics An Analysis of Dependencies By Robert Martin October 28,1994
[15] Programming for the Java™ Virtual Machine By Joshua Engel
[16] Ilene Burnstein, "Practical Software Testing", Springer International Edition, Chennai, 2003.
[17] Roger S. Pressman, "Software engineering - A practitioner 's Approach ",McGraw-Hill International Edition, 6th edition, 2005.
[18] A. Mitchell and J.F. Power. "Run-time cohesion metrics: An empirical investigation." In International Conference on Software Engineering Research and Practice, pages 532-537, Las Vegas, Nevada, USA, June 21-24 2004.
[19] "Reduce, Reuse, Recycle, Recover: Techniques for Improved Regression Testing" Mary Jean Harrold College of Computing Georgia Institute of Technology Atlanta, GA 30332-0280 harrold@cc.gatech.edu
[20] "Configuration aware prioritization techniques in regression testing" Xiao Qu Dept. of Comput. Sci. & Eng., Univ. of Nebraska, Lincoln, NE. IEEE Confreence (2009)
[21] "Regression testing for component-based software systems by enhancing change information "Chengying Mao Coll. of Comput. Sci. & Technol., Huazhong Univ. of Sci. & Technol., China Yansheng Lu APSEC '05. 12th - Asia-Pacific Software Engineering Conference, 2005.
[22] P. R. Srivastava, "Test Case Prioritization," Journal of Theoritical And Applied Information Technology. JATIT 2008.
[23] Rothermel, R.H. Untch, C.Chu ,M.J.Harrol, " Test Case Prioritization: An Emperical Study," In Proceedings of the24thIEEE International Conference Software Maintenance (ICSM) Oxford, U.K, September 1999.
[24] M.Ramalakshmi Praba, Dr. D.Jeyamala, "Critical Component Analyzer - A Novel Test Prioritization Framework for Component Based Real Time Systems" ,MySec 2011 - organized by IEEE-Malaysia and Institute of Technology, Malaysia,IEEExplore.
[25] D.Jeyamala,Critical Components Identification and Verification for effective software Test Prioritization,International Conference on Advanced Computing 2011 Organized by Anna University, Chennai,IEEExplore.