A Hierarchical System Test Case Prioritization Technique based on Requirements

13th Annual International Software Testing Conference in India, 2013
04 – 05, December 2013, Bangalore, India

Harish Kumar
Department Of Computer Engineering
YMCA University of Science & Technology, Faridabad, India
Email:htanwar@gmail.com

Ved Pal
Department Of Computer Engineering
YMCA University of Science & Technology, Faridabad, India
Email:ved_ymca@yahoo.co.in

Naresh Chauhan
Department Of Computer Engineering
YMCA University of Science & Technology, Faridabad, India
Email:nareshchauhan19@gmail.com
A Hierarchical System Test Case Prioritization Technique based on Requirements

Abstract - System Testing encompasses a large number of test cases, which may not be able to get executed due to constrained time, budget and limitation of the resources. Therefore, the test cases must be prioritized in some order such that the critical and most required functionality can be tested early. In this paper, a hierarchical approach for system test case prioritization based on requirements has been proposed that maps requirements on the system test cases. This approach analyzes and assigns value to each requirement based on a comprehensive set of twelve factors thereby prioritizing the requirements. Further, the prioritized requirement is mapped on the highly relevant module and then prioritized set of test cases. To analyze the effectiveness of this approach, a case study of income tax calculator software [10] has been taken. The existing as well as the proposed approach were applied and analyzed on this software. The results show the efficacy of the proposed approach in terms of fault detection and severity early.

Keywords: System Testing, Test case prioritization, fault severity.

1. Introduction

Test case prioritization techniques organize the test cases in a test suite by ordering in such a manner that the most critical test cases are executed first thereby increasing the effectiveness of testing. The prioritization techniques [4] provide a way to find out more bugs under resource constrained environment and thus improve the reliability of the system quickly. Moreover, as faults are revealed earlier, software engineers have more time to fix the bugs and adjust the project schedule. Many prioritization techniques have been proposed for prioritizing the system test cases on the basis of requirements. However, the requirements only in consideration cannot include critical test cases. The implementation complexity and test case complexity may also affect the test case prioritization. Though Hema Srikanth [1] has included the developer perceived complexity for implementation factor but it is only a scaling assigned by developer explicitly. There may be lot of complexities and issues in design and code of the mapped requirements. All these factors should also be considered while prioritizing the test cases. The researchers have also considered, fault proneness of requirements only in connection with customer-reported failures. But, there is need to consider fault-proneness for every requirement with every affected factor. Moreover, the fault proneness associated with mapped code should also participate in prioritizing the test cases.

In this paper, a hierarchical test case prioritization is proposed wherein the prioritization process is performed at three levels given below:

1. The requirements are first prioritized on the basis of twelve factors by assigning a priority weightage to each requirement.
2. The highest priority requirements are then mapped to their corresponding modules to get prioritized modules.
3. The test cases based on to the highest prioritized module are then put in order for execution.

2. Related Work

Hema Srikanth et al. [5] considered four factors for analyzing and measuring the criticality of requirements. These factors are Customer-Assigned priority of requirements, Requirement Volatility, Developer-perceived implementation complexity. Based on these four factor values, a Prioritization factor value (PFV) is computed. PFV is then used to produce a prioritized list of system test cases.

R. Kavitha & N.Suresh Kumar [7] proposed a method to prioritize the regression test cases considering the following factors: (1) Customer assigned priority of requirements, (2) Developer-perceived code implementation complexity, (3) Changes in requirements, (4) Fault impact of requirements, (5)Completeness, (6) traceability (7) Execution time. Based on these factors, a weightage was assigned to each test case in the software thereby prioritizing the test cases.
Patric Berander and Anneliese Anfrews [2] considered an approach that provides means to find an optimal subset of requirement resulting in trade of desired project scope against sometime conflicting constraint such as schedule, budget, resources, time to market and quality. They also considered requirement prioritization as the basis of the product strategy.

Maya Daneva and Andera Herrman [1] proposed a conceptual model of requirements prioritization based on benefit and cost prediction.

Siripong Roongruangsuwan and Jirapun Daengdej [9] proposed a new classification of test case prioritization techniques considering a new test case prioritization method along with practical weight factors like test case complexity, dependency and test impact etc.

A critical review of the work done by the researchers in the direction of system test case prioritization indicates that the following factors have not been considered that may affect the system test case execution:

- **Developer assigned priority**: The developer may assign the priority to every requirement on the basis of its importance.
- **Show Stopper Requirements**: These are the critical requirements in the absence of which the software may not work. The developer may therefore assign the priority to these types of requirements.
- **Frequency of Requirements**: It is the frequency of a requirement how much it is being used in the software.
- **Expected fault**: The developer may analyze the causes which may make the software error prone.
- **Implementation Complexity**: It is the criteria how much each requirement is difficult to implement considering technology dependency, interdependency of the requirements, complexity of requirement itself, etc.
- **Cyclomatic Complexity**: It is the logical complexity [3] of a program. The module with higher complexity may lead to complex test cases.
- **Non DC path**: In data flow graph [3] of a program, the non-dc paths are the problematic areas with respect to the use of a variable. Therefore, this factor may also be considered for module prioritization.

### 3. Proposed Work

The proposed approach starts with analyzing and assigning value to each requirement based on a comprehensive set of twelve factors thereby prioritizing the requirements. After getting the ordered list of requirements, a mapping between the highest priority requirement and its corresponding modules is performed. The modules are then prioritized based on cyclomatic complexity and non dc path. The weighted prioritized module is then selected for testing. It may be possible again that there are several test cases corresponding to this selected module. For this purpose, the third level of prioritization is applied by prioritizing these several test cases based on four factors. In this way a hierarchical system test case prioritization technique is proposed and discussed in subsequent sections.

**Abbreviations and Acronyms**

- **RPFV** – Requirement prioritization factor value
- **MPFV** – Module prioritization factor value
- **TCPV** – Test Case prioritization factor value

### A. Process of prioritization of system test cases

In the proposed prioritization process almost every stakeholder viz. the customer, developer, tester, and business analyst participate. The prioritization process includes the following steps (See Figure. 1).
Figure 1. Process for System test case prioritization based on requirement

Steps:

1. Customer, developer, analyst and tester assign values to the requirement factors.
2. Apply the process of the requirements prioritization (RP).
3. On the basis of the prioritized requirements a mapping between the requirements and their corresponding modules is performed.
4. Apply the process of prioritization of the modules (MP).
5. Tester assigns the value to each factor of test case of the prioritize module.
6. Apply the process of test case Prioritization (TP).
7. The resulting test suite contains the prioritized test cases.

B.Prioritization of requirements

There are twelve factors on the basis of which process of prioritization of requirements is performed. These factors are assigned a value on a scaling between 1 to 10. For each requirement, a Requirement Prioritization factor value (RPFV) is calculated as given below.

$$RPFV_i = \sum_{j=1}^{n} (rfvalue_{ij} \times rfweight_j)$$
In the above formula $RPFV_i$ represents the prioritization factor value for requirement $i$, which is the summation of the product of factor value and the assigned factor weight for each of the factor. $rfvalue_{ij}$ represent the value for factor $j$ for requirement $i$, and $rfweight_j$ represents the factor weight for $j$th factor for a particular product.

Table 1. Requirements Prioritization

<table>
<thead>
<tr>
<th>FACTORS</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>rfweight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer assigned</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>0.02</td>
</tr>
<tr>
<td>Developer assigned</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>0.08</td>
</tr>
<tr>
<td>Requirement Volatility</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>Fault Proneness</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.15</td>
</tr>
<tr>
<td>Expected fault</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>0.10</td>
</tr>
<tr>
<td>Implementation Complexity</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>0.10</td>
</tr>
<tr>
<td>Execution frequency</td>
<td>5</td>
<td>10</td>
<td>9</td>
<td>6</td>
<td>0.05</td>
</tr>
<tr>
<td>Traceability</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>Show stopper requirement</td>
<td>0</td>
<td>9</td>
<td>6</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Penalty</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>0.05</td>
</tr>
<tr>
<td>Cost</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>0.05</td>
</tr>
<tr>
<td>Time</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>0.05</td>
</tr>
<tr>
<td><strong>RPFV</strong></td>
<td><strong>2.25</strong></td>
<td><strong>4.77</strong></td>
<td><strong>4.15</strong></td>
<td><strong>2.47</strong></td>
<td><strong>1.0</strong></td>
</tr>
</tbody>
</table>

Table 1 shows the prioritization of four sample requirements on the basis of the RPFV for each requirement. By using Table 1 prioritization of the test cases for the four requirements are R2, R3, R4, and R1. The weights assigned to each factor are shown in Table 1 are project specific provided by development team and customer.

C. Prioritization of Module (MP)

In the Prioritization of module mapping between the prioritized requirement and its corresponding modules is performed. If there is more than one module then the modules are prioritized. The criteria for module prioritization are based on its cyclomatic complexity and non dc path [3]. The higher the cyclomatic complexity and non dc path of the module, the higher is the priority of that module. For each module a module prioritization value (MPV) is calculated as given below.

$$MPV= \text{Cyclomatic complexity} + \text{Number of Non dc path}$$

Table 2. Modules Prioritization

<table>
<thead>
<tr>
<th>Factors</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclomatic Complexity</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Non-dc Path</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td><strong>MPV</strong></td>
<td><strong>15</strong></td>
<td><strong>9</strong></td>
<td><strong>10</strong></td>
<td><strong>8</strong></td>
</tr>
</tbody>
</table>

Table 2 shows the prioritization of four sample module on the basis of MPV for each module. The order of modules on the basis of MPV is M1, M3, M2 and M4.
D. Test case prioritization process (TP)

The test case prioritization process is used to prioritize and schedule the test cases corresponding to prioritized modules. After getting the prioritized modules mapping between the prioritized modules and their test cases is performed. The several test cases corresponding to a module are prioritized by assigning a test case prioritization value (TCPV) based on four factors. TCPV is calculated as given below:

\[ TCPV_i = \sum_{j=1}^{n} (tfvalue_{ij} \times tfweight_j) \]

In the above formula TCPV\(_i\) represents the prioritization factor value for test case \(i\), which is the summation of the product of factor value and the assigned factor weight for each of the factor. \(tfvalue_{ij}\) represent the value for factor \(j\) for test case \(i\), and \(tfweight_j\) represents the factor weight for \(j\)th factor for a particular product.

Table 3. Test Cases Prioritization

<table>
<thead>
<tr>
<th>Factors</th>
<th>TC1</th>
<th>TC2</th>
<th>TC3</th>
<th>TC4</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Impact</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>.25</td>
</tr>
<tr>
<td>Test Case Complexity</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>9</td>
<td>.25</td>
</tr>
<tr>
<td>Requirement Coverage</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>.25</td>
</tr>
<tr>
<td>Dependency</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>.25</td>
</tr>
<tr>
<td>TCPV</td>
<td>6.25</td>
<td>5.50</td>
<td>5.25</td>
<td>7.50</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 3 shows the prioritization of four sample test cases on the basis of TCPV for each test case. The order of test cases on the basis of TCPV is TC4, TC1, TC2 and TC3.

4. Effectiveness of Proposed approach

To analyze the effectiveness of the proposed prioritization technique it was applied to income tax calculator software [10]. The software consists of 1160 lines of code and has five modules. This software is based on five requirements: accept Personal detail (APD), accept income detail (AID) accept tax deduction (ATD) accept Savings and Donation details (ASD) and Generate tax detail (GTD). All types of bugs like critical, major, medium and minor bugs were introduced intentionally so that testing can be performed on the software using this new approach. Table 4 shows the total faults severity of each requirement. The faults severity [8] [10] is calculated as given below:

Fault severity = 4\* no. of critical bugs + 3\* no of major bugs + 2\* no of medium bugs + 1\* no of minor bugs

Table 4. Number and type of faults detected by all requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Critical</th>
<th>Medium</th>
<th>Major</th>
<th>Minor</th>
<th>Fault Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTD</td>
<td>1</td>
<td>10</td>
<td>5</td>
<td>6</td>
<td>39</td>
</tr>
<tr>
<td>ATD</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>AID</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>ASD</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>APD</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>11</td>
</tr>
</tbody>
</table>
The proposed work was further analyzed on the basis of APFD metric [6]. The APFD graph as shown in Figure 2 indicates that APFD value obtained for proposed approach is more than the previous methods, thereby showing the efficacy of the prioritized method.

![Graph showing comparison between Random, PORT, and Proposed approach](image)

**Figure 2.** Comparison between Random, PORT, and Proposed approach

### 5. Conclusion

A hierarchical system test case prioritization technique has been presented in this research paper. The proposed technique maps the requirement to its corresponding design modules and further mapped to the corresponding test cases. This approach can be used to improve the rate of severe fault detection for system testing. An experimental study of income tax calculator software is presented for comparing the effectiveness of proposed approach with previous approach (PORT) and with random prioritization approach. The experimental results show that proposed new prioritization technique is promising in terms of ordering requirements so that faults are detected earlier in the testing phase.
References

Appendix:
Testing: The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software items.
Error: The difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.
Fault: An incorrect step, process, or data definition in a computer program.
System Testing: System testing is the process of attempting to demonstrate that a program or system does not meet its original requirements and objectives, as stated in the requirement specification.
Test Case Prioritization: It is the process of prioritizing the test cases based on certain criteria. The purpose of prioritization is to reduce the set of test cases based on some rational, non-arbitrary criteria, while aiming to select the most appropriate tests.
Cyclomatic Complexity: It is the number which tells us about the complexity of the design of a module. It should be less than 10, otherwise the module should be redesigned.
DC-path: A dc-path with respect to a variable v is a path between the definition node and the usage node such that no other node in the path is a defining node of variable v.
Authors

1. HARISH KUMAR
Harish Kumar is pursuing his Ph.D. in Computer Engineering from YMCA University of Science & Technology. He has completed his M.Tech (CE) and B.Tech. (CE) from M.D.U. Rohtak in the year 2006 and 2004. He has 8 years of experience in teaching. Presently he is working as an Assistant Professor in department of Computer Engineering in YMCA University of Science & Technology, Faridabad, Haryana, India. His research areas include Software Testing, Project Management, Software Engineering & Computer Programming.

2. Ved Pal
Ved Pal is pursuing his Ph.D. in Computer Engineering from YMCA University of science & Technology, M. Tech (CE) From YMCA University of science & Technology, Faridabad in year 2012, MCA from MD University Rohtak in year 2008. He has four Year experience in teaching. Presently he is working as an assistant professor in department of computer engineering in YMCA University of science & Technology, Faridabad, Haryana, India. His Research Areas include Software engineering, Software Testing and object Technology.

3 Dr. Naresh Chauhan
Naresh Chauhan received his Ph.D in Computer Engineering in 2008 from M.D. University, M. Tech (IT) from GGSIT, Delhi in year 2004, B.Tech (CE) from NIT Kurukshetra in 1992. He has 20 years of experience in teaching as well as in industries like Bharat Electronics and Motorola India Pvt. Ltd. Presently he is working as a Professor and Chairman in the department of Computer Engineering, YMCA University of science & Technology, Faridabad, Haryana India. His Research areas include Internet Technologies, Software Engineering, Object-Oriented Technologies, Operating Systems, Software Testing, Software Quality Management and Real Time Systems.