Chapter 1

Introduction

1.1 Background

Software has found an enormous dissemination in the past years. At the beginning stages of software development process, requirements and specifications for the software product are prescribed in order to explain the intended usage for the product. As the system grows, it should confine to these requirements. The most important requirement of a software product is that it should meet the requirements of the customer. In today’s changing business environment, time-to-market is a key factor to achieve a successful product. As mentioned by Karlsson and Ryan in [Kar97], for a product to be most successful, its quality must be maximized while minimizing the cost and keeping delivery time short. Software quality concerns are quite broad, including correctness, robustness, readability and changeability. Quality can be measured by the customer satisfaction with the resulting system based on the requirements that are incorporated successfully in the system. Due to the importance of the quality of the software product, more than fifty percent of the cost of software development is devoted to testing. Software testing is the most important method of assuring the quality of software. Software testing is an expensive [Cra02] and iterative process as presented in Figure 1.1. There are two major types of testing: black box testing and white box testing. Black box testing ignores the internal mechanism of the system and focuses on the outputs generated for specified input conditions [Iee90]. White box testing takes into account the internal mechanism of the system. System testing is a type of black box testing conducted in the last phase before the product is delivered to the customer.
In this research, new approaches are presented to improve user-perceived software quality.

Throughout the software life cycle, test cases should be written to adequately test the software. Test cases can be defined as sets of input parameters for which the software will be tested. The testing process using test cases is shown in Figure 1.2.

In the testing process test cases are selected and executed and the results are compared with the estimated result. Even with good planning it is impossible to execute all planned test cases, within the time and budget for the total test cases. As quoted in one of the examples in [Elb00], an industrial collaborator stated that one of his products of approximately 20,000 lines of code, required seven weeks to run the entire test cases. During regression testing the entire test cases have to be rerun, whenever a change is made to even a small portion of the code. Due to
the substantial amount of overhead incurred, developers are continually seeking methods to reduce the cost and time of regression testing. In so far different methods have been developed to reduce the cost and time of regression testing. The most common of these techniques include (i) test case selection and (ii) test case prioritization.

1.1.1 Test case selection

Regression test selection techniques attempt to reduce the cost of regression testing by selecting and running only a subset of the test cases in a sensible way [Rot96]. While this approach may lessen the cost of performing regression testing, it is quite difficult to find a balance between the time required to select and run test cases and the fault detection ability of the remaining test cases. As stated in [Tod01], test case selection techniques are not effective, implying that they could fail to select a test case that would have revealed a fault. Although the test case selection technique proposed by Rothermel and Harrold [Rot97] is safe, the magnitude of work required to prove that the subset of test cases exposes the same number of faults as the full test suite is difficult in some instances. Though some test suite minimization and selection techniques lower the costs as reported in [Rot98], the fault detection has to be compromised. To improve the fault detection capability Dennis Jeffery and Neelam Gupta [Den07] selectively retained the test cases during test suite reduction.

1.1.2 Test case prioritization (TCP)

To optimize the time and cost spent on testing, prioritization of the test cases for execution in a test suite can be beneficial. Prioritization of the execution order of test cases in a test suite, can improve test efficiency. Prioritization techniques are often preferred to selection techniques because they do not eliminate any test cases from the original test suite. Test case prioritization
techniques, sort the test cases to increase their effectiveness at achieving some performance goal. For example Rothermel et al. [Rot 99] sorted the test cases such that the test cases with the highest levels of code coverage are run first. Usually test cases are arranged to improve the rate of fault detection. The rate of fault detection is a measure of how quickly a test suite detects faults during the testing process. Focusing on a different prioritization approach, Elbaum et al. has performed several studies in regression test suite prioritization based on code coverage [Elb00]. In these studies, an empirical investigation was carried out using several structurally-based criteria to prioritize tests. One goal was to determine if a prioritized test suite would demonstrate a higher rate of fault detection, and if so, which criterion would best prioritize the test suite. The study discovered that all structurally-based criteria examined, even the simplest approaches, can improve the rate of fault detection in comparison to the use of no prioritization technique. It is also important to note that the increase in the rate of fault detection is not necessarily significant. If all of the test cases within a test suite can be executed in a short time, the increase caused by the prioritization may not be worth the effort. However, if a test suite takes days or weeks to run, meeting testing goals earlier can yield meaningful benefits. There can be number of possible goals of test case prioritization. For example testers may wish to increase the rate of fault detection, decrease the cost of fault detection, find the critical faults and cover maximum coding under testing, at the earliest. But these objectives focus on one major perspective that the testers want to increase the confidence in the reliability of the system at a faster rate. So testers are interested in finding the maximum number of faults as well as the most critical faults at the earliest.

The primary motivation of the research work presented in this thesis is to prioritize the test cases based on requirement factors, requirement coverage, code
coverage with the goals to improve the customer satisfaction, the rate of severe fault detection and to reduce cost and time of testing. This is because it has been observed from the literature that most of the prioritization techniques found in the literature have considered only coverage information. Even though very little literature is available on test case prioritization, effort has been put forward to propose a set of prioritization techniques in this research.

In order to measure the performance of the prioritization technique, it is necessary to assess effectiveness of the ordering of the test suite. The effectiveness of the prioritization techniques are measured by validation metrics. The most common of these validation metrics include (i) Average Percentage of Fault Detected (APFD) and (ii) modified APFD (APFD²).

1.1.3 APFD metric

To quantify the goal of increasing a subset of the test suite's rate of fault detection, a metric called APFD developed by Elbaum et al. that measures the rate of fault detection per percentage of test suite execution is used. The APFD is calculated by taking the weighted average of the number of faults detected during the run of the test suite. APFD can be calculated as follows using a notation introduced by Kapfhammer [Gre04]:

\[
APFD(T, P) = 1 - \frac{\sum_{i=1}^{g} \text{reveal}(i, T)}{ng} + \frac{1}{2n}
\]  

(1.1)
Table 1.1 Faults detected by regression test suite $T = \{ T_1, \ldots, T_7 \}$

<table>
<thead>
<tr>
<th>Faults</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
<th>$T_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_2$</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_3$</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_4$</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_5$</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For example, if it is assumed that the test suite $T$ consist of seven test cases $\{T_1, T_2, T_3, T_4, T_5, T_6, T_7\}$ and then it is assured that the tests detect the five faults $f_1, f_2, f_3, f_4, f_5$ in $P$ as shown in Table 1.1. Consider the two prioritized test suites $TS_1$ with test sequence $\{T_1, T_2, T_3, T_4, T_5, T_6, T_7\}$ and $TS_2$ with test sequence $\{T_1, T_3, T_2, T_4, T_6, T_7, T_3\}$. Incorporating the data from Table 1.1 into the APFD equation yields

$$APFD(TS_1, P) = 1 - \frac{3 + 1 + 2 + 6 + 2}{7 \times 5} + \frac{1}{2 \times 7}$$

$$= 1 - 0.4 + 0.07$$

$$= 0.67,$$

and

$$APFD(TS_2, P) = 1 - \frac{1 + 7 + 3 + 4 + 3}{7 \times 5} + \frac{1}{2 \times 7}$$

$$= 1 - 0.51 + 0.07$$

$$= 0.56.$$
Thus, according to the APFD metric, $TS_j$ has a better rate of fault detection than $TS_2$ and is therefore more desirable. Calculation of APFD is possible only when prior knowledge of faults is available. Hence the APFD calculations are exclusively used only for evaluation.

1.1.4 Modified APFD metric

$APFD^C$ is a “cost conscious” model of APFD, developed by Elbaum et al. The modified version incorporates test case execution time to provide a more accurate measure of the percentage of faults discovered in a certain period of time. The metric rewards test case orders, proportionally to their rate of “units-of-fault-severity-detected-per-unit-test-cost” [Elb01]. $APFD^C$ is calculated as follows:

Let $T$ be a test suite containing $n$ test cases with costs $t_1, t_2, \ldots, t_n$. Let $F$ be a set of $m$ faults revealed by $T$, and let $f_1, f_2, \ldots, f_m$ be the severities of those faults. Let $\text{reveal}(i, T)$ be the first test case in an ordering $T'$ of $T$ that reveals fault $i$. The cost-cognizant average percentage of faults detected during the execution of $T'$ is given by the equation:

$$APFD^C(T, P) = \frac{\sum_{i=1}^{m} (f_i \left( \sum_{j=\text{reveal}(i, T)}^{n} t_j - \frac{1}{2} t_{\text{reveal}(i, T)} \right))}{\sum_{i=1}^{n} t_i \times \sum_{i=1}^{m} i = 1 \text{mf}_i}$$

(1.2)

This equation remains applicable when either test case costs or fault severities are identical. Further, when both test case costs and fault severities are identical, the formula reduces to the formula for APFD. Rather than comparing just the percentage of total faults a test case finds, in relation to the percentage of detected faults, $APFD^C$ also considers the percentage of total test case cost incurred. Application of the equation to each of the new test suites would identify the most successful regression test prioritization orderings.
Along with the primary motivation presented in section 1.1.2, other motivations of the research work presented in this thesis is to find new metrics to compute the rate of severe fault detection, to compute the average percentage of test cases executed to detect the induced faults and to compute the average percentage of requirements satisfied by the test cases. It is so because it has been observed from the literature that in the existing validation metrics requirements are not quantified and severity levels of faults are not assessed, and also equal priority is given to all faults.

1.2 Research objectives

The objective of this research is two fold, namely, to propose test case prioritization techniques for new and regression testing and to propose validation metrics to measure the effectiveness of the proposed prioritization techniques. In order to achieve these objectives a set of goals are realized viz. (i) to improve the rate of severe fault detection, (ii) to improve customer satisfaction, (iii) to improve the requirement coverage, and (iv) to minimize the time and cost of testing.

In this research five different prioritization techniques and three different validation metrics are proposed. The proposed prioritization techniques and validation metrics are as follows.

Proposed Test Case Prioritization (TCP) techniques
1. TCP with factor values
2. TCP with factor values and factor weights
3. TCP with requirement weights and test case costs
4. TCP with fuzzy approach
5. Regression test suite prioritization using Genetic Algorithms

Proposed Validation metrics
1. Average percentage of Severe Fault Detection (ASFD),
2. Average Test Effort Index (ATEI),
3. Requirement Satisfied by Test case cost \((\text{ReqSatTcc})\)

An overview of the research works carried out in this thesis work is presented in the following sections.

1.3 TCP with factor values and validation metrics ASFD and ATEI

In software testing, making changes to the system during the last phase of the development cycle could be very expensive [Hou89]. In order to optimize the time and cost spent on testing, the current code coverage TCP techniques are extended and a new system level black box TCP technique is proposed in chapter 2 with the goals to improve the rate of severe fault detection and the customer satisfaction. The new prioritization technique is based on the proposed factors viz. customer assigned priority of requirements, developer-perceived code implementation complexity, changes in requirements, fault impact of requirements, completeness and traceability, that influence the requirements. These factors are identified by conducting post mortem analysis.

This proposed prioritization technique prioritizes the test cases based on the weights of the test cases. In order to compute the weights of the test cases, the weights of the requirements are computed using the requirement factor values. For each requirement, these factors are assigned values in a 10 point scale. With these factor values, weight of each requirement is computed. These requirement weights are used to compute test case weights, by mapping the test cases to the requirements. Based on the computed weights, the test cases are prioritized. In order to measure the performance of the proposed prioritization technique, it is necessary to assess effectiveness of the ordering of the test suite. To measure the
effectiveness of the prioritized test cases, two different metrics that incorporates severity of faults are proposed in chapter 2 as follows.

- Average percentage of Severe Fault Detection (ASFD)
- Average Test Effort Index (ATEI)

The proposed ASFD metric is based on the analysis of the faults detected for a product under testing. For analysis purpose, faults are classified into five different severity levels based on the impact of the failure. Faults of various severity levels are assigned various severity values. The assignment of severity values is carried out to enable the computation of Total Severity of Faults Detected (TSFD) for a product, which in turn serves to compute the Average Severity of Faults Detected (ASFD) for each requirement. The computation of ASFD needs knowledge of the mapping of the faults to the requirements. This mapping is obtained using the proposed end-to-end traceability. ASFD is used to analyze the effectiveness of the prioritization technique and it is intended to experimentally test whether the requirement with a higher computed requirement weight, actually has higher ASFD, when the product is system tested.

To measure the effectiveness of the proposed prioritization technique another new metric, Average Test Effort Index (ATEI) is proposed in chapter 2. This proposed metric, analyzes the average number of test cases executed to detect the induced faults. The lower ATEI value indicates the better prioritization.

The effectiveness of the proposed prioritization technique and the validation metrics are experimented with, five student projects and two industrial projects, using the proposed metrics ASFD and ATEI. The results obtained prove that the rate of severe fault detection and the user satisfaction show remarkable improvisation compared to the existing random ordering and coverage based prioritization. Also two industrial case studies are analyzed by conducting a post hoc analysis of the test efforts.
1.4 TCP with factor values and factor weights

To improve the test effort, a new TCP technique with two new factors viz. usability and application flow, identified through another postmortem analysis is proposed in chapter 3 of this thesis. Along with factor values, factor weights are also incorporated in this new prioritization technique with the goals to improve the rate of severe fault detection and the customer satisfaction.

To prioritize the test cases, values are assigned to the proposed factors in a 10 point scale. Unlike the prioritization technique, without factor weight proposed in chapter 2, first, the weight for each factor is computed using the factor values. With these factor weights and factor values, weight of each requirement is computed. These requirement weights are used to compute test case weights, by mapping the test cases to the requirements. Based on the computed weights the test cases are prioritized.

The measurement of effectiveness of the proposed TCP technique is executed through the same set of validation metrics ASFD and ATEI proposed in chapter 2. The proposed TCP technique is validated by conducting experiments on two industrial case studies. The results obtained indicate, that the proposed prioritization technique improves the rate of severe fault detection compared to the existing random ordering and the TCP technique with factor value, proposed in chapter 2.
1.5 TCP with requirement weights and test case costs

To improve requirement coverage of test cases and to reduce the cost of testing, a new prioritization technique is proposed based on the requirement weights and test case cost and the same is presented in chapter 4 of this thesis. In this technique, to prioritize the test cases, the same set of factors identified in chapter 3 are considered and values are assigned to these factors in a 10 point scale. These factor values are used to compute the weight of each factor with the following condition.

$$\text{Sum of all factor weight} = 1$$

With factor values and weights, the weight of each requirement is computed. For each test case, based on the execution time and requirement coverage information from the previous tests, the cost is computed. Based on the obtained values of requirement weight and the test case costs, the following prioritization algorithms are proposed.

i. Additional requirement coverage based test case prioritization (Addtl_TCP)

ii. Total requirement coverage based test case prioritization (Total_TCP)

To measure the effectiveness of the prioritized test cases, a metric similar to APFD (explained in section 1.1.4), called Requirement Satisfied by Test case cost (ReqSatTcc) is proposed in chapter 4. The proposed metric evaluates the rate of requirement weights satisfied per unit cost. An increased ReqSatTcc, improves testing quality and customer satisfaction. It provides earlier feedback on the system under test, supporting faster strategic decisions about release schedules. Moreover, if the testing period is minimized, it can increase the possibility of executing those test cases that cover the higher testing requirement priorities during the available testing time.
The effectiveness of the proposed TCP technique and the validation metric are experimented with an industrial case study. Also an experiment is conducted on two industrial projects to measure the effectiveness of the proposed two prioritized orders \textit{Addtl\_TCP} and \textit{Total\_TCP}. The results obtained indicate that the proposed technique \textit{Addtl\_TCP} improves the requirement coverage of test cases in reduced cost compared to the existing random ordering. Also the proposed TCP technique improves the test effort compared to TCP with factor values, proposed in chapter 2 and TCP with factor values and factor weights, proposed in chapter 3.

1.6 TCP with fuzzy approach

In chapter 5 of the thesis, a new fuzzy logic approach is presented to prioritize the test case that can improve the rate of severe fault detection. In this technique, to prioritize the test cases, the same set of factors identified in TCP with factor values and weights are considered and values are assigned to these factors in a one point scale. These factor values are represented in terms of fuzzy input membership functions: Low, Medium and High. Based on the total number of factors and input membership functions, the total number of fuzzy rules is computed and these rules are generated with their corresponding proposed output values. Based on a set of rules, the weight of each input membership function corresponding to each factor is computed and is used to compute the rule number to be fired, to obtain the requirement weight. These requirement weights are utilized to compute test case weights by mapping the test cases to the requirements. Based on the computed weights the test cases are prioritized.

The proposed prioritization technique is validated with the validation metrics ASFD and ATEI. The effectiveness of the proposed prioritization
technique is measured by conducting experiments with student projects and industrial projects. The results obtained, prove that the rate of severe fault detection and the user satisfaction are improved compared to the existing random ordering. Also this fuzzy based TCP technique improves the test effort compared to TCP with factor values, proposed in chapter 2, TCP with factor values and factor weights, proposed in chapter 3 and TCP with requirement weights and test case costs, proposed in chapter 4.

1.7 Regression test suite prioritization using Genetic Algorithms

Finally, another soft computing technique, Genetic Algorithms (GA) is used to prioritize the test cases based on the coverage information and maximum time allotted to the test cases and the same is presented in chapter 6. It aims to improve the code coverage and reduce the cost and time of testing.

In this technique, for each test case, the maximum time allotted for execution, the potential fault information and the coverage information are considered for test case prioritization. Initially, the execution time (without loading time) of each test case is recorded. Code coverage information is obtained using the Emma coverage tool. With this coverage information and execution time required by a test case, fitness value is computed. Initially two best test suites are selected to be the elements in next generation. Through the roulette wheel selection method, the pairs of test suites are identified. These selected pairs of test suites are merged with cross over method, to create two new test suites. Each test suite in the pair may be mutated based on mutation method. New test case may be added or deleted from the test suite pair. These processes are repeated until all the test cases in a test suite are considered to obtain the prioritized order.
The effectiveness of the proposed prioritization technique is measured with the existing validation metric called Average Percentage of Faults Detected (APFD) explained in section 1.1.3. To validate the proposed Genetic algorithm based prioritization technique, experiments are conducted on two programs viz. JDepend and Progress Report, to analyze the effectiveness of the proposed GA based prioritization technique, with block coverage and method coverage. The results obtained prove that the proposed prioritization technique improves the average percentage of faults detected both in block coverage and method coverage. Also, experiments are conducted on two industrial projects to measure the test effort of the prioritized test cases. The genetic algorithm based TCP technique improves the test effort compared to random order and to all the other proposed TCP techniques in this research. A comparison of all the five prioritization techniques proposed in this thesis, based on the ATEI metric is presented in chapter 6.

Henceforth this dissertation is structured as follows. The new TCP technique with factor values is described in chapter 2. Validation metrics based on the rate of severe fault detection and average test effort index is also presented in chapter 2. In chapter 3, TCP technique with factor values and factor weights are presented. The TCP technique with requirement weights and test case costs is presented in chapter 4. Also in chapter 4 a new validation metric, to measure the rate of requirement coverage is presented. Soft computing techniques such as fuzzy logic and genetic algorithm based TCP techniques are presented in chapter 5 and chapter 6 respectively. A comparison of average test effort of all the five proposed techniques is also discussed in chapter 6. Conclusion and future directions are presented in chapter 7.