Chapter 3

Requirement Based System Test Case Prioritization of New and Regression Test Cases

3.1 Introduction

In this chapter a new prioritization technique has been proposed with two new prioritization factors for regression testing and with computed weights to the proposed six factors. This proposed technique prioritizes the test cases based on their weights. The procedure for computing the weights of the test cases is three fold. Firstly, to calculate the weights of the proposed factors, the factors that influence the requirements are identified. The post mortem analysis carried out (explained in section 3.3) to identify the proposed factors, reveals that the factors Usability and Application Flow have higher influence on regression test cases than Completeness and Traceability. Subsequently, the impacts of these factors on the requirements are quantified by assigning values in a ten point scale. With these factor values, factor weights are computed proportional to the mean value of each factor for all the project requirements. Intuitively, factors which have higher weights tend to be more important for the proposed test case prioritization technique.

Secondly, to compute the weight of the requirements, the factor values and the factor weights are used. Finally, the test cases are mapped towards corresponding requirements, by establishing a knowledge based mapping between them. The proposed prioritization technique has been validated and experiments have been conducted to study the effectiveness of the proposed prioritization technique on two industrial case studies and on two industrial

projects. The results obtained indicate that the proposed prioritization technique improves the rate of severe fault detection.

The related work found in the literature for test case prioritization is presented in the following section.

3.2 Related work

Rothermel et al. [Rot01] present a number of techniques that use test execution information to prioritize test cases for regression testing. These techniques are classified broadly into three categories: (i) order test cases based on total coverage of code components; (ii) order test cases based on coverage of code components previously uncovered, and (iii) order test cases based on estimated ability to reveal faults. The elaborate experiments that accompany this work several that each of the prioritization techniques improved the rate of fault detection.

In subsequent work, Elbaum et al. [Elb02] address three important questions: (i) are prioritization techniques more effective when made specific to modified versions? (ii) does granularity (e.g., statement vs function level) of coverage matter? and (iii) do inclusion of measures of likelihood of faults provide any improvements? New techniques are presented and detailed experiments suggest that version-specific prioritization improves test case ordering for the specified version; granularity and likelihood of faults do not significantly improve prioritization ordering.

Srivastava and Thiagarajan [Sri02] present Echelon, a test- prioritization system that runs under a Windows environment. Test cases are prioritized based on the changes made to a program. Echelon uses a binary matching system that can accurately compute the differences at the basic block level between two versions of the program in binary form. Test cases are ordered to maximize

coverage of affected regions using a fast, simple and intuitive heuristic. Echelon has been found to be quite effective on large binaries.

Harrold et al. [Har93] present a technique for selecting a representative set of test cases from a test-suite, which provides similar coverage. In other words, the technique is used to eliminate redundant and obsolete test cases. In [Rot98], Rothermel et al. show through experiments that a potential drawback of redundant test elimination techniques is that in minimizing a test suite, the ability of that test suite to reveal faults in the software may be reduced.

Jeffrey and Gupta [Jef05] present another technique to minimize this draw-back. Apart from the commonly shared goal of improving the process of regression testing, the redundant test-case elimination problem is addressed by Jeffry et al. In [Dic01], Dickinson et al. find failures by cluster analysis of execution profiles.

Bryce and Colbourn [Bry05a] present a case for test case prioritization for interaction testing, a mechanism for testing systems deployed on a variety of hardware and software configurations. They adapt a greedy method to generate a set of tests to identify all pairwise interactions when the ordered set of test cases is run to completion or else the more important test cases are run, if the testing is terminated without completion. Bryce et al. perform a detailed study of greedy algorithms for the construction of software interaction test suites in [Bry05b].

There has been substantial work in the area of impact analysis [Api05, Law03, Ors03, Ram06]. In many of these approaches, functions that follow a modified function in some execution path are added to the impact set. One of the reasons for using dynamic impact analysis is to reduce the parts of program that need to be retested while performing regression testing. For example, in [Ren04], Ren et al. detect a set of changes responsible for a modified test's behavior and the set of tests that are affected by a modification are identified. The information obtained through impact analysis can help in designing new methods for

prioritization. As long as these new methods can be specified in terms of a relation between test cases, a test case dissimilarity graph can be built. Many interesting techniques have been devised for bug detection in software systems [God07, Csa06, Liv05]. For example, in [God05], Godefroid et al. present a technique to automatically generate test cases so that the coverage of the program is increased.

3.3 Factor identification process

To determine the two new factors involved in the proposed prioritization technique, we conduct a postmortem analysis with the same set of projects used in section 2.3.1 of chapter2. Test cases of Project-A are executed for its customer usability and application flow. Test cases of Project-B are executed for its completeness and traceability. The faults are classified into four degrees i.e. Severity1: Testing must be stopped until the defect is fixed, Severity2: Tester can work around failure but the faults have to be fixed before the product can be deployed, Severity3: Faults can be fixed in later version, and Severity4: Faults cannot be fixed at all. A total of 97 faults (37 faults of severity1, 25 faults of severity2, 20 faults of severity3, and 15 faults of severity4) in Project-A and 33 faults (15 faults of severity1, 10 faults of severity2, 8 faults of severity3)in Project-B are identified in the system test.

In project-A, 38% of faults are of severity1, 25% of faults are of severity2 and in project-B, 45% of faults are of severity1, 30% of faults are of severity2. Then the whole system is divided into four different modules: the first module comprises of codes for ease-of-use, second module comprises of codes for application flow, third module comprises of codes for completeness and fourth module comprises of the codes for traceability. The faults are then mapped to these four different modules and percentage of faults of severity1 and severity2 are obtained. In Project-A, 65% of severe faults are found in module1and

module2 and 35% of severe faults are found in module3 and module4. In Project B, 69% of severe faults are found in module1 and module2 and 31% of severe faults are found in module3 and module4. For each of the projects-A and B, module versus percentage of faults of severity1 and severity2 are obtained and the same is presented in Figure 3.1

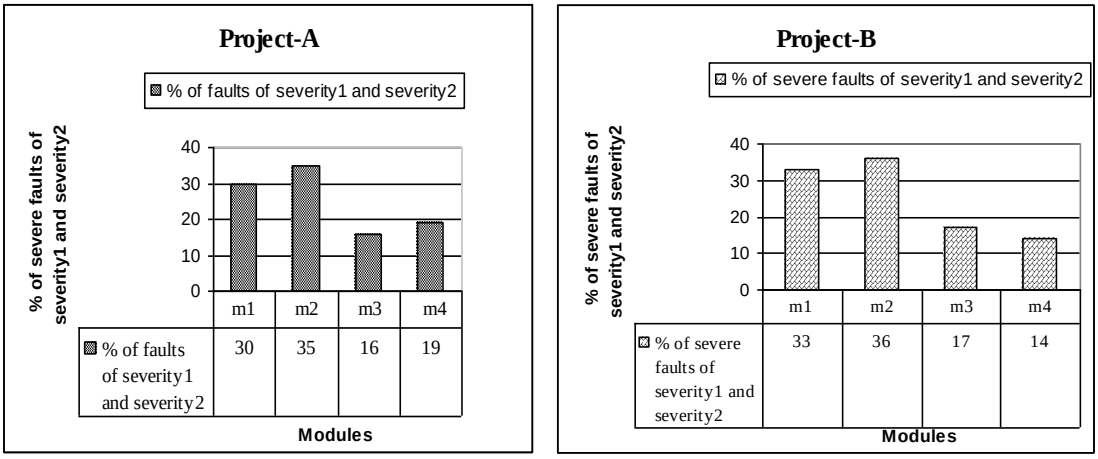


Figure 3.1 Fault - factor-module

The outcome of this post mortem analysis indicates that the influence of the factors usability and application flow on regression test cases is significantly higher than the factors viz. completeness and traceability. This result motivated us to consider usability and application flow. Hence it is proposed that the factors completeness and traceability is replaced by usability and application flow to propose the following new set of factors: (1) customer assigned priority of requirements (2) developer-perceived code implementation complexity (3) changes in requirements (4) fault impact (5) usability and (6) application flow.

3.4 Proposed prioritization technique

In this section, the rationale behind the selection of the proposed factors, computation of factor weight, the proposed prioritization technique and the steps involved in the proposed prioritization technique are presented.

3.4.1 Reason for the selection of factors

Having presented the rationale behind the selection of factors (1-4) in the previous section, we now present the factors 5 and 6 viz. usability and application flow in this subsection.

i) Usability (US): It assures that an application is easy to understand. The implementation of system specification may negate some aspects of ease-of-use design which in turn may reduce the quality of the software. Each user interface requirement is analyzed for its usability and value ranges from 0 to 10 is assigned by the customer, when that requirement is considered for reuse.
Reasoning: Customer satisfaction such as the rapidity with which the software responds to the user request can be improved by considering usability of the requirement.

ii) Application flow (AF): It assures that the program performs the specified function in the manner indicated and that the functional data can be accumulated properly from run to run. Each requirement is analyzed for its flow and a value ranging from 0 to 10 is assigned by the tester when that requirement is considered for reuse.
Reasoning: The quality of the software can be improved by considering the application flow of the requirement.

3.4.2 Computation of factor weight

In this proposed technique, a weight is assigned for each of the proposed six factors for the project using the factor values. This factor weight is unique for a particular project and allows the user to customize the factors' importance in each project. First, the factor weights are assigned based on the discussion between customer and the development team. For more effectiveness, factor

weights are assigned proportional to the mean value of each factor for all the project requirements as shown in Equation (3.1).

$$\text{Weight of a factor} = \frac{\textit{Mean Value of That Factor}}{\textit{Sum of Mean Value of All Six Factors}} \tag{3.1}$$

3.4.3 Proposed technique

In this proposed work, the values for all the six factors are assigned for each requirement during analysis phase. These values tend to evolve continuously during the software. Based on the mean value of the factors, weights are assigned to all the six factors for a project using equation (3.1). With factor values and factor weights of a project, the Requirement Factor Value for requirement i, RFV_i is proposed to be computed by the following relation.

$$RFV_i = \sum_{j=1}^{6} (\text{FactorValue}_{ij} \times \text{Factor Weight}_j) \tag{3.2}$$

Here RFV_i signifies the Requirement Factor Value for requirement i, which is the summation of the product of factor value and the assigned factor weight for each of the factors. The proposed RFV for requirements is represented in a value matrix and the same is given in equation (3.3).

$$\begin{pmatrix} RFV_i \\ . \\ . \\ . \\ . \\ RFV_n \end{pmatrix} = \begin{pmatrix} R_1^{cp} & R_1^{IC} & R_1^{RC} & R_1^{FI} & R_1^{US} & R_1^{AF} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ R_n^{cp} & R_n^{IC} & R_n^{RC} & R_n^{FI} & R_n^{US} & R_n^{AF} \end{pmatrix} \begin{pmatrix} W_{CP} \\ W_{IC} \\ W_{RC} \\ W_{FI} \\ W_{US} \\ W_{AF} \end{pmatrix} \tag{3.3}$$

The RFV is used in the computation of Test Case Weight (TCW). Since traceability is an important characteristic, several tools like traceability tree and RebaTe are used in testing software. In this proposed work test cases are mapped to its associated requirements by the testers using an end–to-end Requirement

Traceability tool, TBreq, which is unique in providing end-to-end traceability reports in a single view.

Having computed the requirement weights and the traceability mapping between the requirements and test cases, the TCW is computed, as a product of the following two elements:

(i) The average RFV of the requirements the test case maps

(ii) The Requirements-coverage a test case provides.

The second element, requirements-coverage is the fraction of the total project requirements exercised by a test case. With the total of n requirements, if a test case *t* maps to *i* number of requirements then, TCW_t is proposed to be computed by the equation (2.5) presented in chapter 2. TCWs are thus computed for all the test cases and these test cases are sorted for execution based on the descending order of TCW, such that the test case with the highest TCW runs first.

3.4.4 Proposed prioritization algorithm

The requirements and their corresponding proposed factor values and test cases serve as inputs to the proposed prioritization technique. The sorted test cases in descending order of its weights, is produced as output. The steps involved in the proposed prioritization technique are presented here under:

Algorithm:

Input: *The requirements and their corresponding proposed factor values and test cases.*

Output: *The sorted test cases in descending order of their weights.*

Begin

1. *Select the factors to be considered based on the prioritization goal.*
2. *Get total number of requirements and total number of test cases planned for the project that is to be tested.*

3. *Get the factor values for all requirements from the person involved in the software development.*
4. *Obtain each factor weight, by computing the fraction of mean value of each factor amongst the total mean value of all factors.*
5. *Compute each Requirement Weight, by computing the sum of product of factor value and factor weight of each requirement.*
6. *Using an end-to-end traceability approach, analyze and map the test cases to the respective requirements.*
7. *Obtain each Test Case Weight, by computing the fraction of requirement weight each of the test cases map amongst total requirement weight of the project.*
8. *Sort the test cases in descending order of their weights and the test cases with a higher weight are run before others.*

End

3.5 Experiments and results

The proposed prioritization technique is validated using two different validation techniques proposed and presented in section 2.4 of chapter 2. First validation technique is based on the analysis of the faults detected for a product and the second validation technique is based on the analysis of the number of test cases executed to detect the faults. Experiments have been conducted in two categories to measure the effectiveness of the proposed prioritization technique. Category I include research methods applied on two industrial case studies and Category II includes two industrial projects. The testing and results of projects in these two categories are presented in the following subsections.

3.5.1 Category I

In this category of experiment, the effectiveness of the proposed prioritization technique is validated based on the rate of fault detection. Two

industrial case studies are analyzed by conducting a post hoc analysis of the test efforts at Cosmosoft Technologies limited, software technology parties of India, a leading developer of ERP projects and Tech Zone, a leading developer of PHP projects in India. The first case study (Cosmosoft Technologies) involves the analysis of student self service portal written in JAVA language, that comprises 91,733 changed lines of code, built upon approximately a 350,000 lines of code base; 25 modified requirements; and 100 system level test cases. The second case study (Tech Zone) involves in analysis of job searching web site, written in PHP, comprising of 200,000 lines of code; 20 new requirements and 100 system level test cases.

The following steps are carried out for the two industrial case studies.

1. Computation of the factor weight is done using equation (3.1).
2. Computation of RFV for the requirements is carried out by applying equation (3.2).
3. Computation of TCW for the test cases is done by applying equation (2.5).
4. Computation of TSFD for the project which is a summation of the severities for all project failures is done by applying equation (2.6).
5. Prioritization of the test cases is carried out according to the proposed prioritization technique.
6. Generation of 20 random ordering of test cases is carried out.

For both the case studies, the engineering teams provide the research team with project requirements, test cases, test failures and mapping of failures to test cases, and test cases to requirements for conducting this analysis. Also the engineering teams provide the factor values for the requirements. RC is assigned

based on two factors: 1) New requirements that are added to the release and 2) existing requirements that are modified for the release.

Table 3.1 Cosmosoft-Student self service portal -Factor values

Req-ID	**CP**	**IC**	**RC**	**FI**	**US**	**AF**
R01 -3.1.1	9	8	5	2	7	6
R02 -3.1.2	8	6	4	1	5	5
R03 -3.2.1	9	7	3	2	6	7
R04 -3.2.2	6	7	3	1	5	5
R05 -3.2.3	9	7	5	2	6	5
R06 -3.2.4	5	4	5	1	9	9
R07 -3.2.5	6	9	2	3	7	7
R08 -3.2.6	7	5	3	2	5	5
R09 -3.2.7	6	9	4	1	4	4
R10 -3.2.8	8	8	5	2	5	5
R11 -3.2.9	4	3	5	2	3	3
R12 -3.2.10	8	7	2	1	5	4
R13 -3.2.11	6	7	2	1	5	4
R14 -3.2.12	5	5	3	2	6	6
R15 -3.2.13	9	3	5	2	7	4
R16 -3.2.16	4	7	3	1	5	6
R17 -3.2.20	6	5	2	2	6	6
R18 -3.3.1	7	3	1	1	7	6
R19 -3.3.2	5	5	5	2	4	7
R20 -3.3.3	3	7	4	3	5	6
R21 -3.3.4	6	5	5	2	6	6
R22 -3.4.1	9	9	5	2	8	8
R23 -3.4.2	8	7	4	1	7	7
R24 -3.4.3	6	5	3	2	8	6
R25 -3.5	5	9	2	1	9	9

Table 3.2 TechZone - PHP website -Factor values

Req-ID	**CP**	**IC**	**RC**	**FI**	**US**	**AF**
R001	10	10	3	0	7	9
R002	10	10	3	0	6	7
R003	10	7	3	0	8	6
R004	7	7	7	0	7	7
R005	10	7	3	0	8	7
R006	7	7	3	0	9	9
R007	10	7	3	0	7	7
R008	10	3	3	0	5	5
R009	10	7	7	0	6	6
R010	7	3	3	0	7	6
R011	10	10	3	0	6	6
R012	10	3	3	0	5	4
R013	7	7	10	0	5	4
R014	10	10	10	0	6	6
R015	10	7	10	0	7	4
R016	10	10	3	0	5	6
R017	3	7	3	0	3	3
R018	9	3	3	0	6	6
R019	5	5	5	0	4	7
R020	10	7	3	0	4	5

CP is assigned, based on the needs of customers. IC is assigned by the engineering team, based on the implementation difficulties of the requirements. FI is assigned based on the failures achieved from previous release. FP is applicable to Cosmosoft-SSSP-case study and not applicable for TechZone-PHP-case study as the project is going through initial release and has no prior field data. US and AF are assigned by the engineering team based on the user friendliness of the product. Factor values for both the case studies are presented in Table 3.1 and Table 3.2. With these factor values, Factor weight is computed

using equation (3.1). For each requirement RFV is computed using equation (3.2). Test cases are mapped to its associated requirements by the testers. The screen shot for mapping of test cases to the requirements is presented in Figure 3.2. Mapping of test cases for both the case studies is presented in Table 3.3 and Table 3.4.

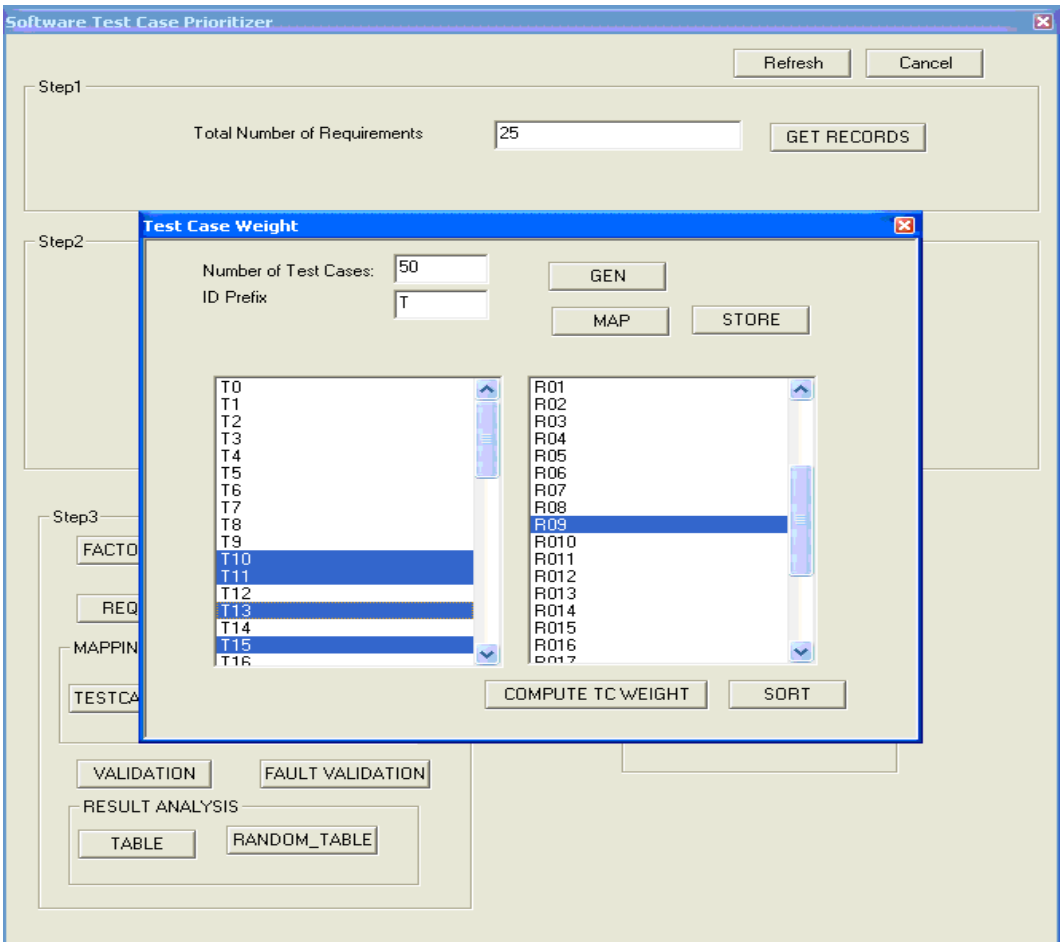


Figure 3.2 Mapping between Test Case and Requirement

Table 3.3 Cosmosoft-SSSP: Mapping Test case to requirements

Test Case ID	Req Id	Req Id	Req Id	Req Id	Req Id
T001-3.1	R01 -3.1.1				

T002-3.1	R01 -3.1.1				
T003-3.1	R01 -3.1.1				
T004-3.1	R01 -3.1.1				
T005-3.1	R01 -3.1.1	R02 -3.1.2			
T006-3.1	R01 -3.1.1	R02 -3.1.2			
T007-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T008-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T009-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T010-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T011-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T012-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T013-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T014-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T015-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T016-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T017-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T018-3.2	R03 -3.2.1	R04 -3.2.2	R05 -3.2.3	R06 -3.2.4	
T019-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T020-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T021-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T022-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T023-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T024-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T025-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T026-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T027-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T028-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T029-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T030-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T031-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T032-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T033-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T034-3.2	R07 -3.2.5	R08 -3.2.6	R09 -3.2.7	R10 -3.2.8	R11 -3.2.9
T035-3.2	R12 -3.2.10				
T036-3.2	R12 -3.2.10				
T037-3.2	R12 -3.2.10				

T038-3.2	R12 -3.2.10				
T039-3.2	R12 -3.2.10				
T040-3.2	R12 -3.2.10				
T041-3.2	R12 -3.2.10				
T042-3.2	R13 -3.2.11	R14 -3.2.12			
T043-3.2	R13 -3.2.11	R14 -3.2.12			
T044-3.2	R13 -3.2.11	R14 -3.2.12			
T045-3.2	R15 -3.2.13	R16 -3.2.16			
T046-3.2	R15 -3.2.13	R16 -3.2.16			
T047-3.2	R15 -3.2.13	R16 -3.2.16			
T048-3.2	R15 -3.2.13	R16 -3.2.16			
T049-3.2	R15 -3.2.13	R16 -3.2.16			
T050-3.2	R15 -3.2.13	R16 -3.2.16			
T051-3.2	R17 -3.2.20				
T052-3.2	R17 -3.2.20				
T053-3.3	R18 -3.3.1	R19 -3.3.2			
T054-3.3	R18 -3.3.1	R19 -3.3.2			
T055-3.3	R18 -3.3.1	R19 -3.3.2			
T056-3.3	R18 -3.3.1	R19 -3.3.2			
T057-3.3	R18 -3.3.1	R19 -3.3.2			
T058-3.3	R18 -3.3.1	R19 -3.3.2			
T059-3.3	R18 -3.3.1	R19 -3.3.2			
T060-3.3	R18 -3.3.1	R19 -3.3.2			
T061-3.3	R18 -3.3.1	R19 -3.3.2			
T062-3.3	R18 -3.3.1	R19 -3.3.2			
T063-3.3	R20 -3.3.3				
T064-3.3	R20 -3.3.3				
T065-3.3	R20 -3.3.3				
T066-3.3	R20 -3.3.3				
T067-3.3	R20 -3.3.3				
T068-3.3	R20 -3.3.3				
T069-3.3	R20 -3.3.3				
T070-3.3	R21 -3.3.4				
T071-3.3	R21 -3.3.4				
T072-3.3	R21 -3.3.4				
T073-3.3	R21 -3.3.4				

T074-3.3	R21 -3.3.4				
T075-3.3	R21 -3.3.4				
T076-3.4	R22 -3.4.1				
T077-3.4	R22 -3.4.1				
T078-3.4	R22 -3.4.1				
T079-3.4	R22 -3.4.1				
T080-3.4	R22 -3.4.1				
T081-3.4	R22 -3.4.1				
T082-3.4	R22 -3.4.1				
T083-3.4	R22 -3.4.1				
T084-3.4	R22 -3.4.1				
T085-3.4	R22 -3.4.1				
T086-3.4	R22 -3.4.1				
T087-3.4	R23 -3.4.2				
T088-3.4	R23 -3.4.2				
T089-3.4	R23 -3.4.2				
T090-3.4	R23 -3.4.2				
T091-3.4	R24 -3.4.3				
T092-3.4	R24 -3.4.3				
T093-3.4	R24 -3.4.3				
T094-3.5	R25 -3.5				
T095-3.5	R25 -3.5				
T096-3.5	R25 -3.5				
T097-3.5	R25 -3.5				
T098-3.5	R25 -3.5				
T099-3.5	R25 -3.5				
T100-3.5	R25 -3.5				

Table 3.4 TechZone-PHP website : Mapping Test case to requirements

Test Case ID	Req Id	Req Id	Req Id	Req Id
T001-S1	R001			

T002-S1	R001			
T003-S2	R001	R002	R004	R016
T004-S2	R002			
T005-S2	R002			
T006-S2	R002			
T007-S2	R002			
T008-S2	R002			
T009-S2	R002	R003		
T010-S4	R004			
T011-S4	R004			
T012-S4	R016	R004		
T013-S4	R016	R004		
T014-S4	R016			
T015-S4	R016			
T016-S4	R016			
T017-S4	R004			
T018-S4	R004			
T019-S3	R003			
T020-S3	R003			
T021-S3	R003			
T022-S3	R003			
T023-S3	R018			
T024-S3	R018			
T025-S3	R018			
T026-S3	R018			
T027-S5	R005			
T028-S5	R005			
T029-S5	R005			
T030-S5	R005			
T031-S5	R005			
T032-S6	R006			
T033-S6	R006			
T034-S6	R006			
T035-S6	R006			
T036-S6	R006			
T037-S6	R017			

T038-S6	R017			
T039-S6	R017			
T040-S6	R006			
T041-S6	R017			
T042-S6	R006			
T043-S11	R011			
T044-S11	R011			
T045-S11	R011			
T046-S11	R011			
T047-S11	R011			
T048-S11	R011			
T049-S7	R007			
T050-S7	R007			
T051-S7	R007			
T052-S7	R007			
T053-S7	R007			
T054-S7	R007			
T055-S8	R008			
T056-S8	R008			
T057-S8	R008			
T058-S8	R008			
T059-S8	R008			
T060-S8	R008			
T061-S8	R008			
T062-S10	R010			
T063-S10	R010			
T064-S10	R010			
T065-S10	R010			
T066-S10	R010			
T067-S10	R010			
T068-S10	R010			
T069-S9	R009			
T070-S9	R009			
T071-S9	R009			
T072-S9	R009			
T073-S9	R019			

T074-S9	R019			
T075-S9	R019			
T076-S9	R019			
T077-S12	R012			
T078-S12	R012			
T079- S12	R012			
T080- S12	R012			
T081- S12	R012			
T082- S12	R012			
T083- S12	R012			
T084-S15	R015			
T085- S15	R015			
T086- S15	R015			
T087- S15	R015			
T088- S15	R015			
T089- S15	R015			
T090- S13	R013			
T091- S13	R013			
T092- S13	R013			
T093- S13	R013			
T094- S13	R013			
T095- S14	R014			
T096- S14	R014			
T097- S14	R020			
T098- S14	R020			
T099- S13	R020			
T100- S15	R020			

After traceability mapping, the TCW of each test case is computed. The test cases are executed based on the descending order of TCW and the faults are detected. The identified faults are assigned severity values as presented in Figure 3.3. For each project an average of 10 faults are identified. These faults are analyzed to determine the corresponding requirements. Mapping the faults to their corresponding requirements are presented in Figure 3.4.

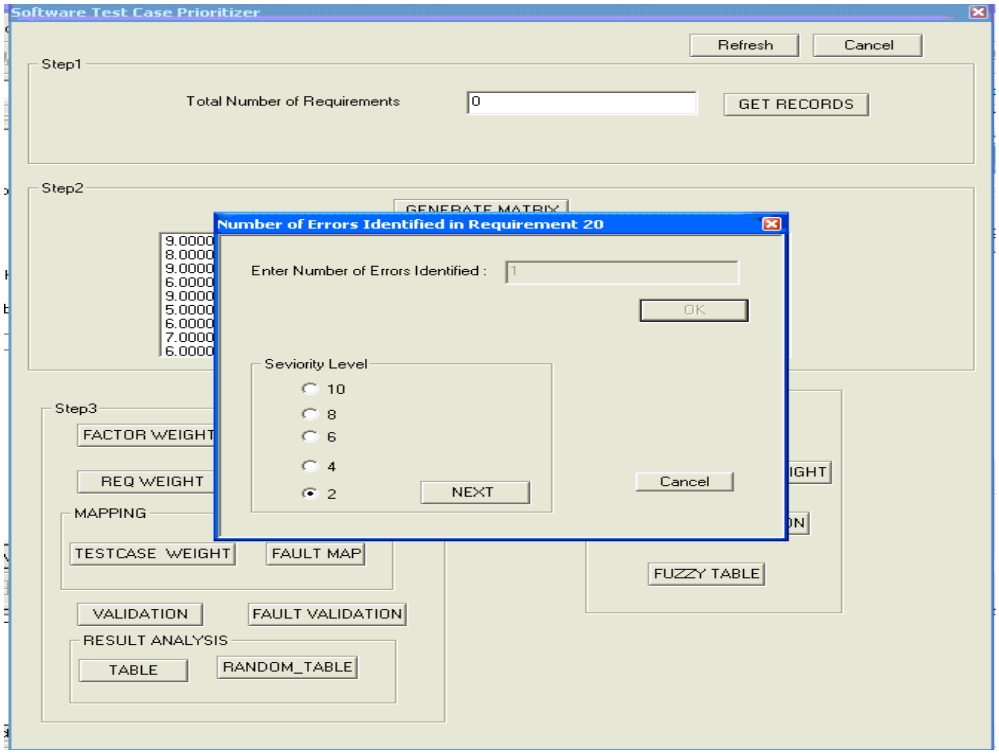


Figure 3.3 Calculation of fault severity

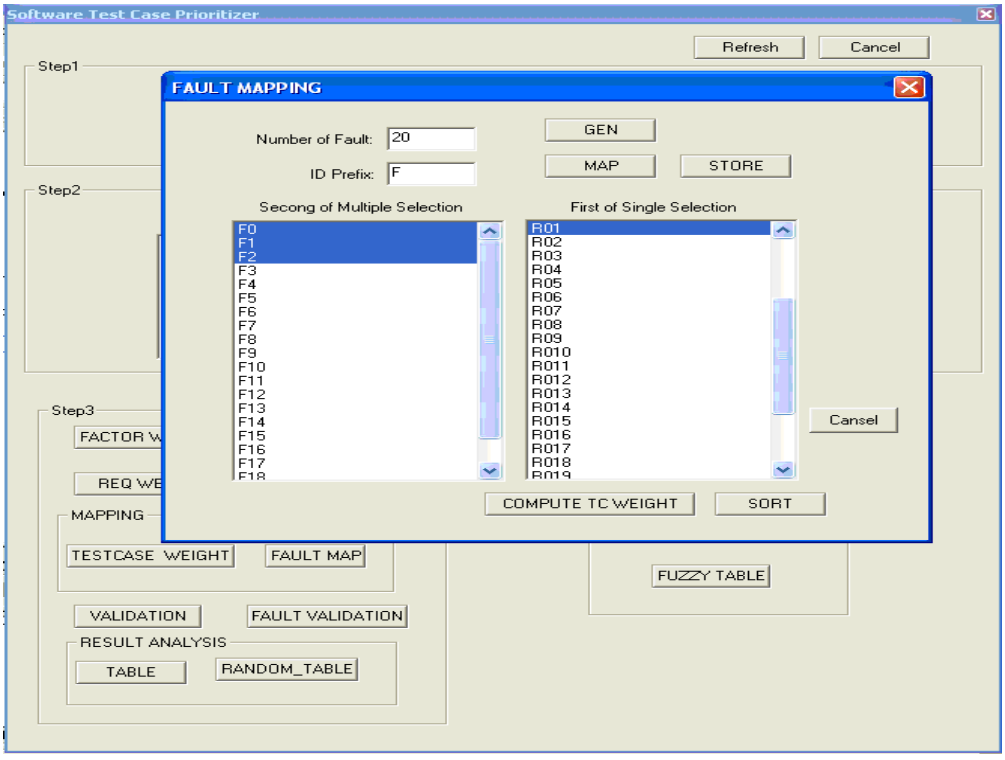
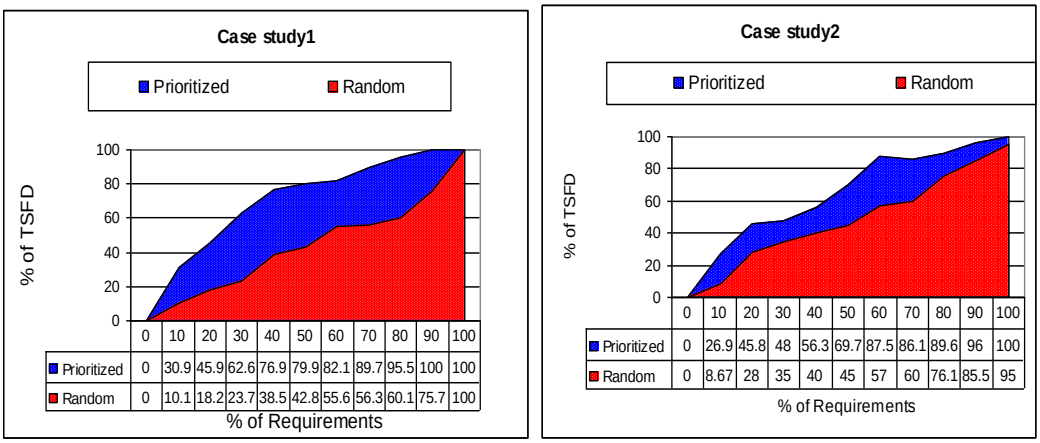


Figure 3.4 Mapping between Faults and Requirements

Based on the faults detected, the two industrial case studies are analyzed in two parts. In the first part, the industrial projects are tested in 20 different random orders of the test cases. The effectiveness of the proposed technique is measured by statistical analysis. The results are presented in the subsection 3.5.1.1. In the second part, the faults identified are evaluated and ASFD computation is made as given in equation 2.7 and the ASFD for each requirement is mapped to their respective RFV. Also the contribution of factors in RFV of requirements is computed. The results obtained are presented in subsection 3.5.1.2.

3.5.1.1 Rate of fault detection

To measure the rate of faults detection, the proposed prioritization technique is compared with a random prioritization strategy for the projects. After executing each test case in the proposed prioritized order, the test result is noted as Pass or Fail and the severity values are noted for the failures identified. TSFD is computed for the projects as shown in equation 2.6. Then the test cases are run in 20 different random orders and the failures are recorded with their severity levels and for each of the two projects TSFD is computed. A graph is plotted with the fraction of requirement executed on X-axis and percentage of TSFD on the Y-axis for both the proposed prioritized order and the random order and the same is presented in Figure 3.5. The rate of detection of fault is computed using TPFD. The mean TPFD values for the 20 random different orderings are compared, with the TPFD, achieved by the proposed prioritization technique.



	0	10	20	30	40	50	60	70	80	90	100
Prioritized	0	30.9	45.9	62.6	76.9	79.9	82.1	89.7	95.5	100	100
Random	0	10.1	18.2	23.7	38.5	42.8	55.6	56.3	60.1	75.7	100

	0	10	20	30	40	50	60	70	80	90	100
Prioritized	0	26.9	45.8	48	56.3	69.7	87.5	86.1	89.6	96	100
Random	0	8.67	28	35	40	45	57	60	76.1	85.5	95

(b)

Figure 3.5 Comparison of rate of fault detection of random tests and proposed prioritized tests

The proposed prioritization is also validated with statistical hypothesis. To allow for statistical comparison, the rate of detection of faults for the proposed prioritization technique is compared with the rate of detection of faults for the 20 random permutations. The TPFD values of the two projects for 20 different random orders and for the proposed prioritized order is presented in Table 3.5. The mean TPFD values for 20 Random orders are also listed in the table. To measure the effectiveness of the proposed prioritization scheme, the following null and alternative hypotheses are considered:

H_0: TPFD for proposed prioritization scheme = Mean TPFD for Random set.

H_a: TPFD for proposed prioritization scheme > Mean TPFD for Random set.

It can be observed from Table 3.5 that TPFD values with the proposed prioritization technique (58.95 and 84.00) are greater than that of random set, for both the projects. So it is evident that the statistical significant results are in favor of H_a and the proposed prioritization scheme is better than random prioritization. The results indicate that the proposed prioritization scheme leads to improved rate of failure detection for both the projects. The difference between proposed

prioritization scheme and mean for twenty permutations is also found to be significant.

Table 3.5 TPFD for Proposed order and Random order

Random Permutations	**TPFD**	
	Cosmosoft-SSSP	**TechZone-PHP**
Random Order 1	51.47	41.05
Random Order 2	47.86	47.03
Random Order 3	53.34	55.34
Random Order 4	49.11	46.21
Random Order 5	49.56	42.26
Random Order 6	46.25	46.37
Random Order 7	56.51	46.85
Random Order 8	47.07	52.10
Random Order 9	43.05	45.94
Random Order 10	50.06	58.42
Random Order 11	52.22	51.01
Random Order 12	46.78	54.58
Random Order 13	50.63	56.34
Random Order 14	47.32	69.46
Random Order 15	43.51	61.46
Random Order 16	57.70	63.70
Random Order 17	48.29	62.96
Random Order 18	53.03	61.68
Random Order 19	53.76	48.72
Random Order 20	51.78	52.56
Mean TPFD	**49.96**	**53.22**
Proposed order TPFD	**58.95**	**84.00**
No of times proposed order is better than random	**20**	**20**
Sign Statistic	**20**	**20**
Sign-test-p-value	**<0.0001**	**<0.0001**

Having described the rate of improving the fault detection, the factor contribution analysis is presented in the next subsection.

3.5.1.2 Factor contribution analysis

i) Analysis of RFV and ASFD

To analyze the values of RFV and ASFD for both the case studies, the RFV of all the requirements are grouped into 5 ranges of values. For each of the RFV range of requirements, the mean ASFD values for requirements in that range is computed. The ASFD for each requirement provides a measure of the severity of faults detected from that requirement. The result of mapping of ASFD values and RFV for requirements for the two industrial case studies is presented in Table 3.6. These results show that the higher percentage of ASFD originates from requirements with RFV range of 4.01 or higher. It is evident from Table 3.6 that the requirements with higher ASFD originate from requirements with higher range of RFV.

Table 3.6 RFV range of total requirements and ASFD

	Cosmosoft-SSSP		**TechZone-PHP**	
RFV range	ASFD	RFV	ASFD	RFV
0-2	0	6.3	0	6.2
2.01-4	0	10.3	5.3	15.7
4.01-6	15.3	21.4	14.7	21.3
6.01-8	24.7	25.2	34.5	22.5
8.01-10	60	36.8	45.5	34.3

ii) Contribution of factors in RFV of requirements

For both the projects, the contribution of each of the six factors towards the RFV pertaining to each requirement and the mean contribution of the overall

project requirements are computed. The mean contribution of the prioritization factors towards RFVs for both the projects is presented in Table 3.7.

Table 3.7 Mean percentage of contributions of factor values

Mean percentage of factor values			
Case studies / Factors	**Case study 1**	**Case study 2**	**Average**
CP	35	42	38.5
IC	25	14	19.5
RC	5	4	4.5
FI	5	2	3.5
US	15	10	12.5
AF	15	28	21.5

It can be observed from Table 3.7 that for both the case studies, the highest contributor is CP followed by IC, AF and US. On an average CP, IC, US and AF contributions are 39%, 20%, 13% and 22% respectively of the total RFV for all projects. The least contributions have come from RC and FI, and are less than 5%. Since the requirements for the projects given are stable, the contribution of RC is less. But RC is likely to be a significant factor in industrial projects when there are numerous changes in requirements. Since there are very few regression test cases, the contribution of FI is less in these case studies. However FI is likely to be a significant factor in case studies that deals with regression testing. The contribution of the six proposed factors for both the case studies is plotted on a graph and is presented in Figure 3.6.

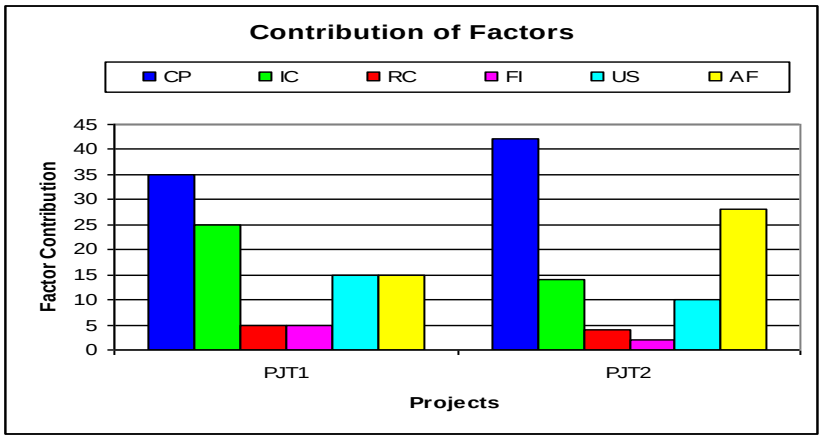


Figure 3.6 Contribution of factors

3.5.2 Category II

In this category of experiment, the effectiveness of the proposed prioritization technique is validated based on the number of test cases executed. For validation purpose the same set of projects (VB project-*Project-1* and PHP project-*Project-2*) each with 20 requirements and 50 test cases, used in section 2.5.2 of chapter 2 is considered. The programs are thoroughly tested by the testers using rational test suite and the project coordinator creates 10 faulty programs (5 faulty programs from each project) by seeding one fault in each, invariant of the severity. On the entire faulty programs the proposed prioritized test cases are run and the total number of test cases executed to find the fault is computed. The test cases are then executed in these 20 different random orders and the total numbers of test cases run to detect the faults are found. The mean value of 20 different results is computed for each of the 10 faulty programs. The results of fault detection in both the cases are compared to strengthen the effectiveness of the proposed prioritization technique. The test cases for the five faulty programs of *Project-1* and five faulty programs of *Project-2* are executed both with the proposed technique and with 20 different random orders.

During the execution of test cases of all faulty programs of *Project-1* in the proposed prioritized order, for the first faulty program, the fault is detected

after running 10 test cases. For second, third, fourth and fifth faulty programs 25, 13, 6 and 4 test cases are executed respectively to detect the faults. TTEI and ATEI are computed with the equations (2.8) and (2.9) respectively as follows:

$$TTEI_{Prioritized} = 10+15+10+6+4 = 45$$

$$ATEI_{Prioritized} = 45/250 = 0.180$$

During the execution of test cases of all faulty programs of *Project-1* in 20 different Random orders, for the first faulty program, the fault is detected after running an average of 20 test cases. For second, third, fourth and fifth faulty programs, an average of 45, 15, 30 and 25 test cases are executed respectively to detect the faults. TTEI and ATEI are computed as follows.

$$TTEI_{Random} = 30+45+18+30+25 = 148$$

$$ATEI_{Random} = 148/250 = 0.540$$

Similarly, for the faulty programs of *Project-2*, $ATEI_{Prioritized}$ and $ATEI_{Random}$ are computed. The values obtained for both the projects are presented in Table 3.8

Table 3.8 ATEI of *Project-1* and *Project-2* for TCP based on factor weight

Project	**$ATEI_{Prioritized}$**	**$ATEI_{Random}$**
Project-1	0.18	0.592
Project-2	0.19	0.700

On comparing $ATEI_{Prioritized}$ and $ATEI_{Random}$ to detect all the induced faults in *Project-1*, 18% of test cases are run in $ATEI_{Prioritized}$ and 59% of test cases are run in $ATEI_{Random}$. Similarly to detect all the induced faults in *Project-2*, 19% of test cases are run in $ATEI_{Prioritized}$ and 70% of the test cases are run in $ATEI_{Random}$. So the number of test cases to be executed to find all the faults is less in the case of proposed prioritized technique and in turn it reduces the cost of testing. Also, lower the value of Average Test Effort Index, better is the prioritization technique. The proposed technique reduces the number of test cases executed approximately by 5% (for *project-1* 23% to 18% and for *project-2* 24% to 19%) when compared with the factor value based technique proposed in

chapter 2. Comparison between the techniques proposed in chapter 2 (with factor values) and chapter 3 (with factor weight) is presented in Table 3.9.

Table 3.9 Comparison of the proposed technique with factor values and the technique with factor weight

Project	**Proposed prioritization techniques**			
	With factor values		**With factor weight**	
	ATEI $_{Prioritized}$	**ATEI $_{Random}$**	**ATEI $_{Prioritized}$**	**ATEI $_{Random}$**
Project-1	0.230	0.540	0.180	0.592
Project-2	0.240	0.640	0.190	0.700

3.6 Conclusion

In this chapter, a new prioritization technique has been proposed with two new prioritization factors for regression testing and weights have been assigned to the proposed six factors. Experiments have been conducted to measure the effectiveness of the proposed prioritization technique in two categories (1) based on the rate of fault detection (2) based on the number of test cases executed. To measure the rate of failure detection, the proposed prioritization technique is compared with a random prioritization strategy for the projects. Statistical tests are employed by conducting Sign test to investigate the null hypothesis that the TPFD for the proposed prioritization technique is no better than that for a randomly chosen prioritization and alternate hypothesis that the TPFD for the proposed prioritization technique is greater than that of a randomly chosen order. The proposed technique increases the rate of fault detection by 5% compared to the technique proposed in chapter 2. A new prioritization technique based on requirement weight and test case cost is proposed and presented in the next chapter.