Journal of Computer Science

Original Research Paper

Combining Model-Based Testing and Failure Modes and Effects Analysis for Test Case Prioritization: A Software Testing Approach

[1]Atifi Meriem and [2]Marzak Abdelaziz

[1]Faculty of Sciences Ben M'sik, University Hassan II,
Avenue Driss El Harti B.P 7955, Sidi Othmane, Casablanca, 20320, Morocco
[2]Faculty of Sciences Ben M'sik, Laboratory of Information Technology and Modeling,
Avenue Driss El Harti B.P 7955, Sidi Othmane, Casablanca, 20320, Morocco

Article history
Received: 09-01-2019
Revised: 07-02-2019
Accepted: 03-04-2019

Corresponding Author:
Atifi Meriem
Laboratory of Information Technology and Modeling,
Faculty of Sciences Ben M'sik,
Morocco
Email: meryematif@gmail.com

Abstract: Due to The complexity of modern software projects and the increasing size of software systems, it becomes difficult to manually perform tests with limited resources. Also, manual testing cannot assure that the software is tested using all possible combinations of inputs. Therefore, automate software testing activities have become primordial in the Software Development Life Cycle (SDLC). Model-based testing is a prominent validation technique in software testing that uses models of the system under test to automatically generate test cases; this generation leads to a large number of test cases; which cannot exhaustively be executed, due to time and cost constraints. Moreover, the test-cases execution order has an influence on the rate at which faults can be detected. Therefore, it is essential to prioritize test cases in order to execute the most important with limited time and cost. On the other hand, The Failure Mode and Effects Analysis (FMEA) considered as one of the formal techniques to do risk based testing. It offers a structured methodology to identifying the system failure modes, analyzing their effects and setting up controls to risk reducing and improve the quality of systems or products. By applying such a technique to Model-based testing projects, we can benefit from FMEA analysis and the automating in same project. Through this paper, we introduced a new model based testing approach for prioritizing and ranking test cases according to the requirements and failure modes. In the suggested approach we used the Failure Mode and Effect Analysis methodology in model based testing way to automatically generate a set of pair {test case, priority number} to prioritize test cases. It differs from existing approaches in that it generates two types of test cases; requirement-based test cases and failure mode-based test cases. For the purpose of experimental evaluation and analysis, we will compare the suggested approach with some well-known prioritization methods and we will present a case study to illustrate the potential application of the proposed approach in a future work.

Keywords: Model-Based-Testing, Software Testing, Test Case Prioritization, Failure Mode and Effect Analysis, Risk Priority Number

Introduction

Software testing is a process of verifying and validating that a software system works as expected and meets the technical and business requirements. It is considered as an important activity to assess software quality. This activity has always been a time and cost consuming task. In this context, much research has

been performed aiming at reducing the time and costs of software testing. Thus, model based testing is being profoundly practiced in most of the software industries to achieve this goal. It represents an attractive solution to do software testing.

Model based testing is a software testing technique that relies on models of a system under test and/or its environment to automatically generate test cases for the system (Utting *et al*., 2012). This generation leads to a large number of test cases; which cannot exhaustively be executed and the execution order has an influence on the rate at which faults can be detected (Huang *et al*., 2017; Rothermel *et al*., 1999). Therefore, it is essential to prioritize test cases in order to execute the most important with limited time and cost. This prioritization of test-cases can increase the rate at which critical faults are detected in the first run.

The main objectives of test case prioritization are to rank the test cases according to an adequacy criterion (Rothermel *et al*., 2001; Elbaum *et al*., 2001) and to reveal faults earlier so as to reduce the total cost of testing.

Rothermel *et al*. (2001) have defined the test case prioritization problem and described several issues relevant to its solution. This problem is defined as follows.

Given:

- T: A test suite
- T' and T'': Are different variations of the test suite
- PT: The set of permutations of T
- f: A function from PT to the real numbers, which represents the preference of the tester while testing

Problem

Find $T' \in PT$ such that:

$$(\forall T'')(T'' \neq T')\left[f(T') \geq f(T'')\right]$$

Different test case prioritization techniques are proposed. These techniques can be categorized into seven categories, we find; code-based test case prioritization, model-based test case prioritization, requirement-based test case prioritization, risk-based test case prioritization and model and risk-based test case prioritization, history-based test case prioritization.

This classification is described and detailed in the form of table as show in Table 1.

For CBTCP, the source code of the system under test is used to prioritize the test cases. This category of techniques involve ranking test cases using the number of covered statements, the number of covered lines of code and the number of functions that were exercised by the test case. Table 2 illustrates some existing test case prioritization techniques that are based on the code coverage.

Table 1: Classification of TCP techniques

Classification	Technique
Category 1	Code-Based Test Case Prioritization (CBTCP)
Category 2	Model-Based Test Case Prioritization (MBTCP)
Category 3	Requirement-Based Test Case Prioritization (ReBTCP)
Category 4	Risk-Based Test Case Prioritization (RBTCP)
Category 5	Requirement and Risk Based Test Case Prioritization (ReRBTCP)
Category 6	Model and Risk Based Test Case Prioritization (MRBTCP)
Category 7	History-Based Test Case Prioritization (HBTCP)
Category 8	Search-Based Test Case Prioritization (SBTCP)

Table 2: CBTCP techniques

Scope	Testing type	Techniques
Code-based	Regression testing New testing	Mishra *et al*. (2018), Chauhan (2018),

Table 3: MBTCP techniques

Scope	Testing type	Techniques
Model-Based	Regression testing	Panigrahi and Mall (2010), Korel and Koutsogiannakis (2009), Tahat *et al*. (2017), Mahali and Mohapatra (2018)
	New testing	Gökçe *et al*. (2015), Belli and Gökçe (2010),

Table 4: ReBTCP techniques

Scope	Testing type	Techniques
Requirement -Based	New testing Regression testing	Srikanth and Williams (2002; 2005a), Kavitha and Sureshkumar (2010), Ashraf *et al*. (2012)

For MBTCP, a model that represents system behavior is used to prioritize test cases. The model based test case prioritization may improve the early fault detection as compared to the code-based test case prioritization (Korel *et al*., 2008). Table 3 illustrates some existing model based test case prioritization techniques.

For ReBTCP, requirements information derived during requirement extraction are used to improve the effectiveness of test case prioritization. Table 4 illustrates some existing Requirement based test case prioritization techniques.

For RBTCP, the risk factor is used to improve the effectiveness of test case prioritization. Table 5 illustrates some existing risk based test case prioritization techniques.

For ReRBTCP, the risks associated with software requirements are used for test case prioritization (James, 1999). Table 6 illustrates some existing requirement and risk based test case prioritization techniques.

For MRBTCP, the risk factor and the models are used in same times to improve the effectiveness of test case prioritization. Table 7 illustrates some existing model and risk based test case prioritization techniques.

Table 5: RBTCP techniques

Scope	Testing type	Techniques
Risk-Based	New testing	Wang *et al*. (2018) Hettiarachchi *et al*. (2016)

Table 6: ReRBTCP techniques

Scope	Testing type	Techniques
Requirement and risk based	New testing	Srikanth *et al*. (2016), James (1999) Srivastva *et al*. (2008)
	New and Regression testing	Srikanth *et al*. (2005b)

Table 7: MRBTCP techniques

Scope	Testing type	Techniques
Model and risk-based	New testing	Wang *et al*. (2018), Zhang *et al*. (2018),
	Regression testing	Rhmann and Saxena (2017)

Table 8: HBTCP techniques

Scope	Testing Type	Techniques
History-based	Regression testing	Huang *et al*. (2012), Kim and Porter (2002),

Table 9: SBTCP techniques

Scope	Testing type	Techniques
Search-based	Regression testing Acceptance testing	Li *et al*. (2007), Shin *et al*. (2018)

Table 8 illustrates some existing history based test case prioritization techniques.

Table 9 illustrates some existing search based test case prioritization techniques.

In this paper, we proposed a new model based test case prioritization approach. We exposed how to perform test-case prioritization in model based testing way using failure mode and effect analysis technique. The proposed approach is both model-based and failure mode-based. It combines model based testing and risk based testing to automatically generate test cases. The main purpose of this approach is to generate test cases associated with their prioritization to rank test execution, consequently, higher priority test cases are executed before lower priority test cases.

The rest of this paper is organized as follows. Section 2 presents discussion of related works. Section 3, presents the necessary background about model based testing technique and their process. Section 4, describes Failure Mode and Effect Analysis (FMEA) method and its process. In Section 5, the main contribution of the paper is introduced, it presents the proposed approach to the test case prioritization problem in model based testing way. Section 6 discusses the benefits and limitations of the new approach versus existing approaches. Finally, Section 7 concludes the paper and describes future work.

Related Works

In the literature, several techniques for Test Case Prioritization have been proposed. However, Huang *et al*. (2017), have examined and investigated test cases prioritization techniques for abstract test cases. They have categorized the existing techniques into four categories; a Non-Information-Guided Prioritization category (NIGP), Interaction Coverage Based Prioritization category (ICBP), Input-model Mutation Based Prioritization category (IMBP) and similarity based prioritization category (SBP). The main finding of this work is that ICBP category has better testing effectiveness than other categories.

Sultan *et al*. (2017), have performed an analytical review and comparison of different test cases prioritization techniques. The comparison study have exposed and compared Fault Severity technique carried out by Varun Kumar and Kumar (2010), fault Localization technique proposed by Kavitha and Sureshkumar (2010), Mutation faults technique effectuate by Malhotra and Bharadwaj (2012), Ordered Sequence of Program Elements technique, Average Percentage of Faults Detected (APFD) technique performed by Srivastava (2008), Model Checker technique performed

by Korel and Koutsogiannakis (2009; Korel *et al*., 2008) and Search Algorithm technique proposed by Maia *et al*. (2010). This comparison is based on key idea, advantages and limits of each technique.

Mohanty *et al*. (2011), have realized a survey of test case prioritization techniques. They have exposed and analyzed some existing techniques in code based, requirement based and model based prioritization. They have exposed methodology and some examples for each category, for example, for code based test case prioritization category, they have presented Srivastava' technique (Srivastava, 2008) which is based on APFD (Average percentage of Faults detected) value, Rothermel' technique (Rothermel *et al*., 1999), Prashant' technique (Malangave and Kulkarni, 2008) and Li' technique (Li *et al*., 2007). For model based test case prioritization category, they have presented the techniques performed by Korel and Koutsogiannakis (2009; Korel *et al*., 2008). For requirement based test case prioritization category, they have presented Srikanth' techniques (Srikanth and Williams, 2005; 2002), Acharya' technique (Acharya and Jena, 2010) and Wu' technique (Wu *et al*., 2001).

On the other hand, in model Based testing area, Belli and Gökçe (2010), have introduced a new approach that use Event Sequence Graphs (ESG) and focus on model based specification and coverage-oriented testing to rank tests according to their preference degrees. The main advantage of this approach is that no priori information is needed about the tests.

Also, Gökçe *et al*. (2015), carried out a search for test case prioritization in model-based testing. They have proposed a new approach that is based on event-oriented graph models. In the proposed approach, the prioritization technique is performed by means of cluster analysis. It provides an effective algorithm for ordering a given set of test cases with respect to their degree of preference as perceived by the tester, which results in a set of priority-sorted test cases. The main contribution this study is the introduction of 13 attributes that enable generating test cases from a model hierarchy with several levels as if it is a single-level model.

In the same context of model based testing, Panigrahi and Mall (2010), have proposed a new technique to prioritize test cases for regression testing of object-oriented programs. They have proposed a new model named "Extended Object-oriented System Dependence Graph" (EOSDG) to model object oriented programs. This model is based on System Dependence Graph for object-oriented software of Larsen and Harrold. Panigrahi's technique involves constructing EOSDG models for original and modified programs to represent control and data dependences as well as static object relations such as inheritance, aggregation and association. For prioritization of test cases, Panigrahi and Mall have constructed a forward slice of EOSDG to identify directly or indirectly affected model elements and they have constructed a backward slice to identify the model elements indirectly tested by a test case. They have also considered the dependencies among test cases while maintaining the prioritized test suite. This approach results in significant increase in the detection of regression faults arising from object relations as compared to related approaches.

Model Based Testing (MBT)

Model-based testing is a software testing technique that relies on models of the behavior of the system and/or its environment to automatically generate test cases (Utting *et al*., 2012; Utting and Legeard, 2007; Pretschner, 2005). The model of the System Under Test (SUT) is created mainly by analyzing system requirements. It defines the expected behavior of the SUT with respect to a set of inputs and is provided as an input to a MBT tool to automatically generate a set of test cases. These test cases are executed and compared with respect to the expected results to report any deviation from the expected behavior. In principle, test cases execution activity can also be automated. However, MBT is mainly concerned with the automation of test case generation. This automation decreases the testing time and helps to achieve increased (and measured) coverage of possible execution scenarios. In addition, test models and generated test cases help to document and analyze the system behavior. Changes in requirements can be reflected to models for generating new test cases. Hence, this approach also reduces the maintenance effort. The model-based testing solution consists of producing test cases from System Under Test (SUT) model and/or its environment model by following process that is composed of five main phases (Atifi *et al*., 2017) (Fig. 1):

1. Requirements management
2. Modeling of an abstract test model dedicated to test of the system
3. Generation of abstract test cases from the test model
4. Concretization of abstract test cases to concrete test cases that can be executed on the system under test
5. Execution of concrete test cases on the system and the constitution of their verdict

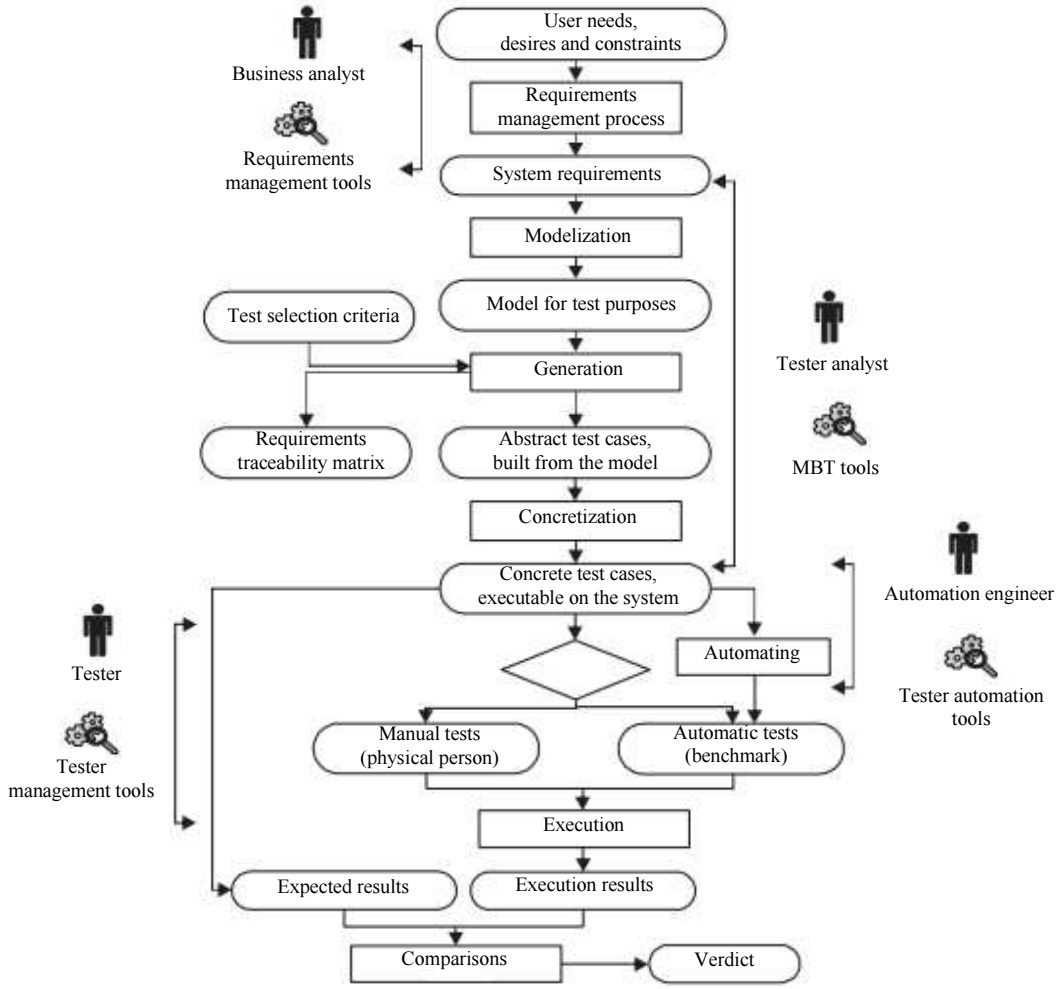

Fig. 1: Model-based testing general process (Atifi *et al*., 2017)

Failure Mode and Effects Analysis (FMEA)

FMEA has been used in many fields to analyze and manage risk. It was used in the late 1950s to study problems that might arise from malfunctions of military systems, precisely for the flight control systems. It appeared as a formal technique in the aerospace and defense industries. Then it spread to the American automotive industry in the late 1970s. In 1985, FMEA was later adopted by the International Electrochemical Commission. Many authors adapted FMEA to various areas in industries, such as nuclear power industry, environmental concerns, software, semi-conductor, web-based distributed design and healthcare (Ben-Daya, 2009; Ben-Daya and Raouf, 1996). In general, there are many types of FMEA, viz; mechanical FMEA, system FMEA, electrical FMEA, product FMEA, software FMEA, process FMEA, human-use or misuse FMEA and health care FMEA. Although the purposes, terminologies and details of each FMEA type are different, but they share the same basic concept that follows the following process (FMEA General process) (Fig. 2):

1. Identify and choose the items to be analyzed
2. Identify the potential failure mode for each item
3. List the effects of each failure mode
4. Rate how severe each effect and categorize the severity of each failure mode (Assign a severity ranking for each effect)
5. Identify the root causes of each failure mode
6. Assess the probability of occurrence of each failure mode (Assign an occurrence ranking for each failure mode)
7. Identify the controls in place to detect the failure mode
8. Assign a delectability ranking for each failure mode
9. Calculate the risk priority number (RPN) of each effect: The RPN is calculated as the multiplication of the probability index, severity index and delectability index
10. Based on RPN number, identify most critical issues and determine actions needs to be taken into consideration
11. Implement controls and recommendations to eliminate or reduce the high-risk failure modes

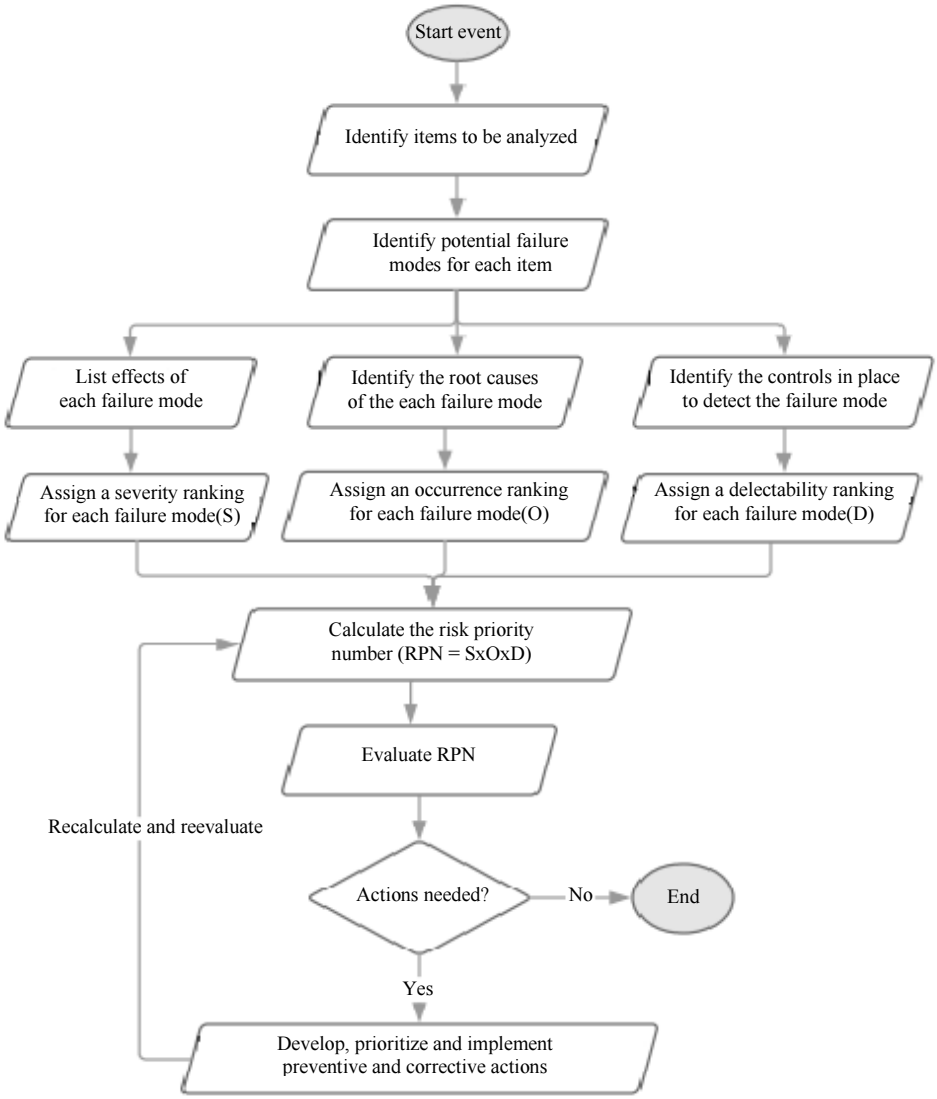

Fig. 2: FMEA General process

In software testing area, FMEA is considered as one of the formal techniques to do risk based testing. It offers a structured methodology to identifying the system failure modes, analyzing their effects and setting up controls to improve the quality of systems or products. FMEA is based on three factors or indexes which are usually evaluated through easily interpreted expressions, each factor is correlated to a score vary between a minimum value and maximum value:

- Severity factor (S) that determining the consequence or the cost of the failure mode. It defines the seriousness of consequences of failure effects. Typically, it is rated on a scale of 1 to 5 or 1 to 10, corresponding to negligible or no effects to catastrophic or very high hazardous or effects
- Occurrence factor (O) that defines the likelihood of occurrence of a failure mode. It examine cause(s) of each failure mode and how often failure occurs. Typically, it is rated on a scale of 1 to 5 or 1 to 10, with highest occurrence corresponding to highest probability and lowest occurrence to lowest probability
- Detectability factor (D) that indicates the probability of detecting the failure. It evaluates the likelihood that a detection method will detect the failure of a potential failure mode before it occurrence. Also, it is ranking in a scale of 1 to 5 or 1 to 10 to rank from lowest to highest detectability

The FMEA factors are used to calculate Risk Priority Number (RPN) to measure risk and severity of a failure to prioritize potential failure modes and root causes.

To express mathematically, the RPN of an artifact x_i can be calculated as follows:

$$RPN(x_i) = S(x_i) \times O(x_i) \times D(x_i)$$

Proposed Approach

Methodology and General Process

In the proposed MBT approach, the main purpose of performing FMEA is to include risk notion in the MBT concept. It is used to identify potential failure modes for each requirement and to calculate RPN for each failure mode using Severity (*S*), Occurrence (*O*) and Detection (*D*) factors. Potential failure mode is defined as the manner in which the system under test could potentially fail to meet the requirements intent. To offer several choices for evaluating failure modes factors, we chose to follow the usual practice that rates failure modes factors on a scale of one to 10 where 1 is lowest and 10 is highest. Table 10 to 12 list the scales used to measure Severity, Occurrence and detectability factors to calculate RPN.

The risk levels of all identified failure modes can be measured by the three characteristics below:

- Frequency of occurrence (probability of occurrence)
- The seriousness of their consequences (Severity)
- Detectability factor

In order to prioritizing the risks we associated with each risk a weight according to five levels (Fig. 3):

- Level 1 = Category 1 => Weight = 1
- Level 2 = Category 2 => Weight = 2
- Level 3 = Category 3 => Weight = 3
- Level 4 = Category 4 => Weight = 4
- Level 5 = Category 5 => Weight = 5

We can then determine the acceptable combinations (risk/severity/frequency/detectability).

Table 10: FMEA scale for probability of occurrence (O)

Probability rating	Class	Meaning
1	None	Failure never occurred
2	Very minor	Failure unlikely occurred
3	Minor	Failure is Relatively low (Rarely occur)
4	Below average	Failure occasionally occurred
5	Average	Failure infrequently occurred
6	Above average	Failure little Occasional
7	High	Failure repeated frequently
8	Very high	Failure is almost inevitable
9-10	Extremely high (with warning)	Failure is inevitable (Always)

Table 11: FMEA scale for severity (S)

Severity rating	Class	Meaning
1	None	Failure with no impact or effect
2	Very Minor	Failure with a very low impact
3	Minor	Failure with a low impact
4	Below average	Failure with less moderate impact on system operations (Tolerable impact)
5	Average	Failure with moderate impact on system operations (not tolerable in critical situations)
6	Above Average	Failure with very moderate impact on system operations
7	High	Failure with high impact on system operations
8	Very high	Failure with very high impact on system operations (Reduced performance)
9	Extremely High (with warning)	Failure with very critical impact on system operations but with warning in advance.
10	Extremely High (without warning)	Failure with very critical impact on system operations and without warning.

Table 12: FMEA scale for Detectability (D)

Detectability rating	Class	Meaning
1	Certain	Failure mode very likely to be detected, it will be detected certainly
2	Very high	Very high chance to detect the potential and subsequent failure mode
3	High	High chance to detect the potential and subsequent failure mode
4	Above average	Above average chance to detect the potential and subsequent failure mode
5	Average	Moderate chance to detect the potential and subsequent failure mode
6	Below Average	Low chance to detect the potential and subsequent failure mode
7	Remote	Remote chance to detect the potential and subsequent failure mode
8	Very Remote	Very remote chance to detect the potential and subsequent failure mode
9	Almost Uncertainly	There is almost no chance to detect the potential and subsequent failure mode
10	Uncertainly	There is no chance to detect the potential and subsequent failure mode

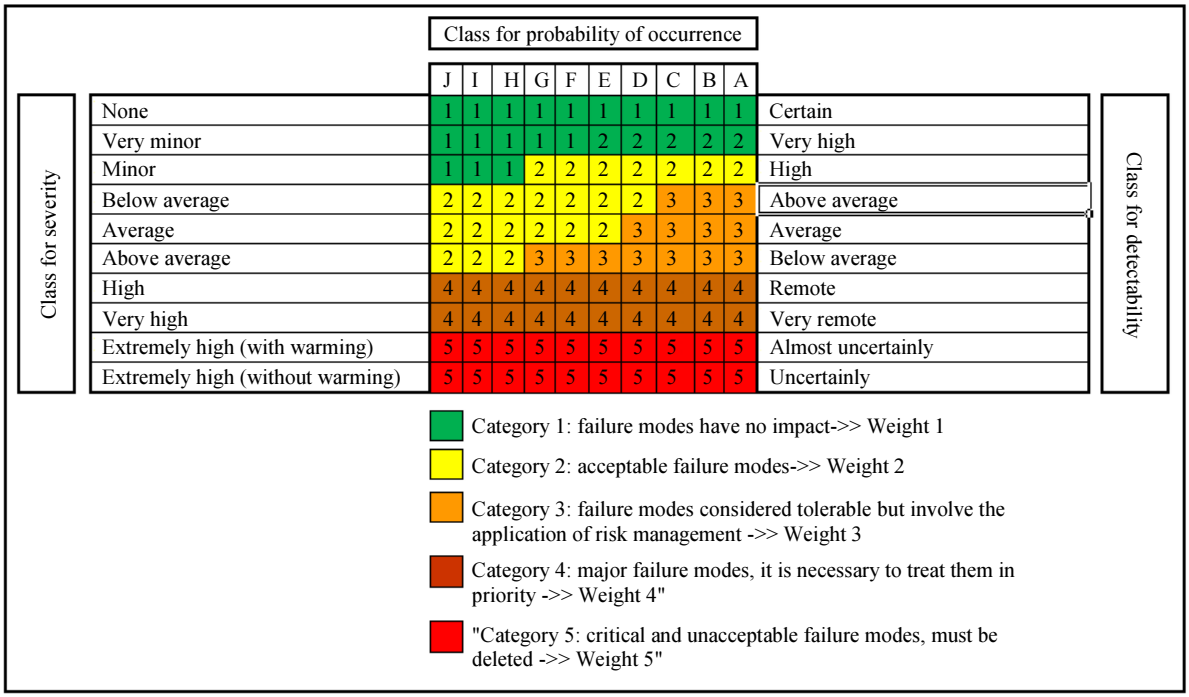

Fig. 3: Probability-severity-detectability Matrix; A and B: Failure Mode is inevitable (Always); C: Failure Mode is almost inevitable; D: Failure Mode is repeated frequently; E: Failure Mode is occasionally occurred; F: Failure Mode is infrequently occurred; G: Failure Mode is little Occasional; H: Failure is relatively low (Rarely occur); I: Failure unlikely occurred; J: Failure never occurred

In our approach we associated a new factor to requirements in order to give a prioritization for each of them. We define it as Requirement Priority Number (RePN).

To calculate the RePN, there are two possibilities according to the generation directives:

- If we consider that the requirement with the highest risk is the highest priority requirement, we can calculate the RePN using the risks weights. In this case, RePN represents the average of RPN of failure modes multiplied by their weights associated to a requirement. Mathematically, the RePN for a requirement is calculated as follow:

$$RePN_{requirement} = \frac{1}{n}\sum_{(i=1)}^{n} RPN(f_i) * Weight_i$$
$$= \frac{RPN(f_1) * Weight_1 + RPN(f_2) * Weight_2 + \cdots + RPN(f_n) * Weight_n}{n}$$

Where:
f_i: Represent a failure mode for a requirement
$Weight_i$: Represent the weight of a failure mode
n: The total number of failure modes for a requirement

Or:

- If we consider that the requirement that has the average of the associated risks high is the highest priority requirement, we calculate the RePN as the average of RPN of failure modes associated to a requirement. Mathematically, the RePN for a requirement is calculated as follow:

$$\text{Re}PN_{requirement} = \frac{1}{n}\sum_{i=1}^{n} RPN(f_i)$$
$$= \frac{RPN(f_1) + RPN(f_2) + \ldots + RPN(f_n)}{n}$$

With:
f_i: Represent a failure mode for a requirement
n: The total number of failure modes for a requirement

General Process

Process Steps

The generic process steps for the new approach are as follows (Fig. 4):

Step-1

Requirements management process, which is the first step of model based testing. It consists to collect customer needs, desires and constraints, manage and then classify

them as requirements. This step is potentially the most important step in any software testing process based on requirements such as MBT. It involves the collection, analysis, prioritization, validation, definition and control of all customer business requirements, it serves to create a requirement repository that is the basis of communication between analysts and testers, is define in a structured way the expected result for the software, in different terms (functional, technical, security, load and response time…).

Step-2

Identify potential failure modes for each requirement, i.e. describe ways in which the system might fail to perform its intended requirement. In this step, Items considered could be previous lessons learnt/problems or new issues from brainstorming session.

Step-3

Identify the list effects of each failure mode. For each requirement, describe the effects of each failure mode from an internal or external view point. Typically, failure effects could be Safety issues, non-compliance to standards, non-functional features, performance issues, intermittent operations and robustness issues in the system under test. A "failure mode" could have multiple effects.

Step-4

Severity assessment step, which consists to assess the seriousness of the effects of each potential failure mode and then assign a severity ranking for the failure mode. The severity levels can be based on a 1-10 scale as per the guideline in Table 2 (FMEA scale for severity (S)) above.

Step-5

Identify the root causes of each failure mode i.e., indicate the weakness that causes the potential failure mode. This step involves constructing a concise, clear and comprehensive list of all root causes of failure mode.

Step-6

Assign an occurrence ranking for each failure mode i.e. estimate a likelihood that a specific failure cause will occur (the probability of occurrence). The probability of occurrence can be based on a scale from 1to10 as per guideline in Table 1 (FMEA scale for probability of occurrence (O)) above:

Step-7

Identify the controls to detect the failure mode of each requirement i.e., list all existing controls and procedures (inspection and test) that prevent either the cause or the failure Mode.

Step-8

Assign a detectability ranking for each failure mode. This step involves assessing the ability of the current control method to detect the failure mode or the failure cause. However we can give a higher value for detectability ranking for a failure mode that can be detected only in testing than the one that can be detected in design itself. Even so, Higher the ability to detect is lower; the value of detection is higher on a scale of 1-10. For example, if the current control mechanism is absolutely certain of detecting the failure mode then the detection would be 1 and so-on as per the guideline in the Table 3 (FMEA scale for Detectability (D)) above.

Step-9

Calculate Risk Priority Number (RPN) for each failure mode as product of severity, occurrence and detection rankings. This ranking prioritizes failure modes, namely, the more the RPN number is higher, the risk of that particular failure mode should be treated first. At the same time, problems with low RPN still deserve special attention if the severity ranking is high.

Step-10

Calculate Requirement Priority Number (RePN) for each requirement as average of RPN of failure modes associated to a particular requirement. This ranking prioritizes requirements.

Step-11

The purpose of the modeling step in our approach is to model the system requirements and their failure modes. It consists constructing a generic model in testing purpose. This model represents in the same time the requirements of the system under test and their failure modes. It is created by a test analyst using requirements resulting from the requirements management step and potential failure modes resulting from the potential failure modes identification step and annotated by Requirement Priority Number (RePN) and Risk Priority Number (RPN). The test model is described in many ways, depending on the discipline. It can be described by use of diagrams, tables, text, or other kinds of notations. In future works, we will present how to create models that represents requirements and failure modes in testing purpose and how to annotate these models by RePN and RPN.

Step-12

The generation step that is realized on the basis of a test generator which takes as input tree elements; the model designed in the modeling step, the test selection criteria selected by the test analyst and the generation directives. The generation directives are new elements in our approach. Each generation directive is classified into one of the following categories follows (Fig. 5):

- A directive for requirement (R)
- A directive for failure mode (FM)
- A directive for both requirement and failure mode (R&FM)

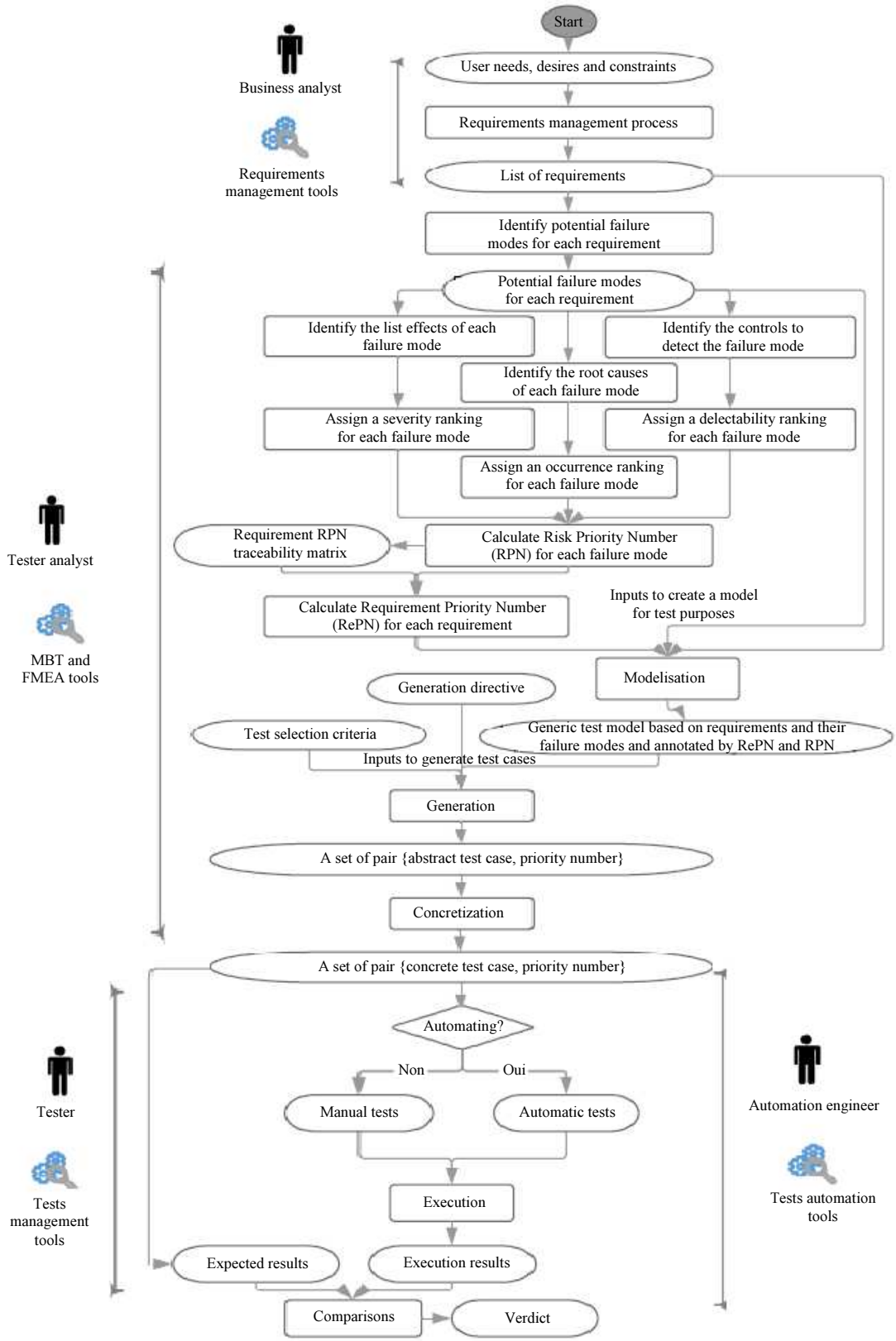

Fig. 4: General process

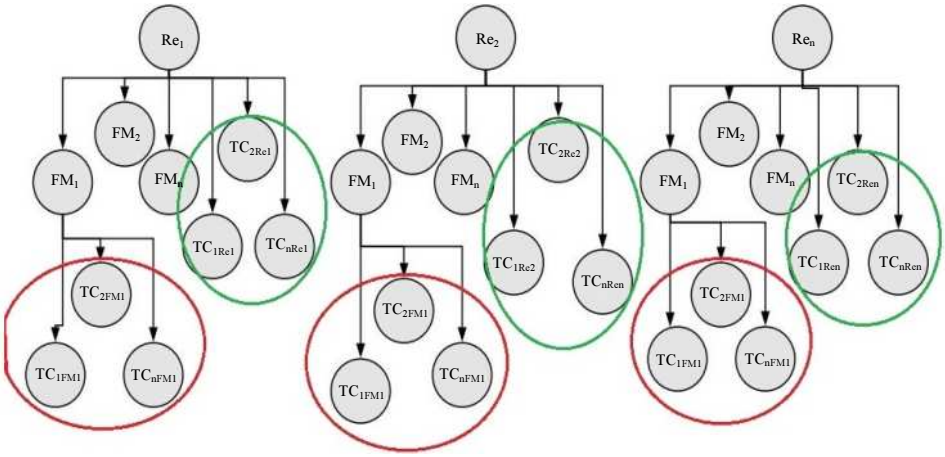

Fig. 5: Test cases, requirements and potential failure modes

A generation directive is a comment with a special syntax; it can be placed wherever comments are allowed. The main purpose of generation directives is to direct automatic test generation if we want to generate test cases based on the failure modes or to generate test cases based on the requirements or to generate test cases based on both the requirements and the failure modes. The generation step produces the abstract test cases from the test model and a traceability matrix that illustrates the link between test cases and model elements covered by the test cases. The traceability matrix is named as Requirement RPN Traceability Matrix (ReRPNTM), it used to track the relationships between test cases, requirements, potential failure modes, risk priority numbers and Requirement priority number. It links each test case with their requirement, their potential failure modes and their priority number and each failure mode with their risk priority number. The outputs of generation step are; a set of pair {test case, priority number}: The priority number of each test case is inherited from the associated requirement if the generation directive is « requirements » or is inherited from the failure mode if the generation directive is « failure modes ».

In the future works, we will detail the generation directives for the automatic test generation and adopt an existing test generator to take in consideration our inputs to generate appropriate test cases.

Step-13

Concretization step, which consists to translate the abstracts test cases to executables test cases in order to be executed on SUT. It consists in making the link between the model elements and the system's concrete elements and involving specific adapters and manual intervention that requires the expertise of the test engineer.

Step-14

Execution step, which can be realized manually or automatically by using an automation testing tool. In this phase, the test cases are executed on the system under test by following the prioritization order generated in the generation phase. Eventually the obtained results are then compared with the expected results to give a verdict for each test case and consequently give a status on the operation of the system.

Global Algorithm for FMEA and Model-based Test Case Prioritization

Our proposed GA is shown in Table 13. It is used to schedule the test cases in test suite T based on their potential failure modes, such as the failure mode severity, the failure mode detectability and the failure mode occurrence.

The Generic algorithm to prioritize test case shown in Table 14 use the output results of FMEA and model-based prioritization algorithm to prioritize test cases and to generate a sorted test suite.

Traceability Matrix

The generated traceability matrix « ReRPNTM » shown in Table 15, identifies links between requirements, potential failure modes, risk priority numbers and Requirement priority number. This matrix tracks a many-to-many relationship – many requirements to many failure modes. One requirement can require multiple failure modes and one failure mode can cover multiple requirements. Typically, the matrix shows the requirements across the top as columns and the associated failure modes down the right side as rows. Each failure mode is associated to a risk priority number (RPN) and each requirement is associated to a requirement priority number (RePN).

Table 13: FMEA and model-based prioritization algorithm

Algorithm 1 FMEA and model-based prioritization

Step 1. Construct a set of requirements R = {X*i*}, where *i* = 1...*n*; *i*∈*N* is a requirement index.

Step 2. For *i* = 1 to *n*

{

Construct a set of potential failure modes FM = {*Yj*} for the requirement X*i*, where *j* = 1...*p*; *j*∈*N* is a failure mode index.

Step 3. For *j* = 1 to *p*

{

Assign the severity ranking for the failure mode Y*j*

Y*j* ← S

Assign the detectability ranking for the failure mode Y*j*

Y*j* ← D

Assign the occurrence ranking for the failure mode Y*j*

Y*j* ← O

Calculate the Risk Priority Number (RPN) for the failure mode Y*j*

$RPN_{Yj} = S(Yj)* D(Yj)* O(Yj)*$

FM[*j*] = *RPN*(*Yj*)

}

Calculate the Requirement Priority Number (RePN) for the failure mode Y*j*

$$\mathrm{Re}\,PN_{Xi} = \frac{1}{n}\sum_{j=1}^{p} RPN(Yj)$$

Or (based on the generation directives) $\mathrm{Re}\,PN_{Xi} = \frac{1}{n}\sum_{j=1}^{p} RPN(Yj) * Weight_j$

}

Step 4. Calculate the Requirement RPN Traceability Matrix ReRPNTM.

Step 5. Construct a generic test model that represent in the same time requirements and failure modes TM.

Step 6. Annotate test model elements with "@Re" for requirements and "@FM" for failure modes.

Step 7. Annotate test model elements with "@RePN" values and "@RPN" values using ReRPNTM matrix.

Table 14: Generic algorithm to prioritize test case.

Algorithm 2 Generic algorithm to prioritize

Input:

TM: Generic Test Model

TSC: Test selection criteria

GD: Generation Directives = ["Re", "FM ", "R- FM "]

$T_{generator}$:

Output:

STS: A sorted test suite STS = A set of pair (Abstract Test Case, Priority Number) sorted by priority number

1. If (GD == Re) Then
   Generate TS' based on Test selection criteria and requirements.
   Sort TS' based on RePN values.
   STS ← TS'.
2. Else If (GD == FM) Then
   Generate TS'' based on Test selection criteria failure modes.
   Sort TS'' based on RPN values.
   STS ← TS''.
3. Else If (GD == R-FM) Then
   Generate TS''' based on Test selection criteria, requirements and failure modes.
   Sort TS''' based on RePN and RPN values.
   STS ← TS'''.

TS', TS'', TS''': Test Suites.

Table 15: Requirement RPN traceability matrix

	Requirement # 1	Requirement # 2	Requirement # 3	Requirement # 4	Requirement # 5	…	Requirement # n
Failure Mode # 1	X			X		X	RPN# 1
Failure Mode # 2		X			X		RPN# 2
Failure Mode # 3		X	X			X	RPN# 3
Failure Mode # 4	X	X					RPN# 4
Failure Mode # 5			X			X	RPN# 5
…	…	…	…	…	…	…	…
Failure Mode # m				X		X	RPN# m
	RePN # 1	RePN # 2	RePN # 3	RePN # 4	RePN # 5	…	RePN # n

Discussion

From the fact that model-based testing has emerged as a major research area in academic and industrial, a large number of publications and new approaches are produced in this field. Most of the previous publications give new MBT approaches. For example Graf-Brill and Hermanns (2017), have proposed an approach that use model-based testing to test asynchronous communicating systems. Also, Wang *et al*. (2017) used model based testing to validate quorum-based systems implemented using the Gorums library through a new approach. On the other hand, some studies give the publications covering supporting techniques for modelling and test generation (Gebizli and Sozer, 2017), integration into industrial practice (Peleska, 2013), taxonomies (Utting *et al*., 2012), industrial evaluations (Blom *et al*., 2016), surveys and classification. For example Utting *et al*. (2012) that have provided taxonomy of model based testing in which the principal aspects of MBT approaches are covered. This paper provides a new model based testing approach to overcome some challenges involved in model based test case prioritization.

Conclusion

In this paper, we have presented a new testing approach to perform test-case prioritization in model based testing way. This makes it possible to efficiently apply prioritization when generation test-cases in model based testing. It consists of generating test cases associated with their priority numbers. In Our approach we have used Failure Mode and Effect Analysis (FMEA) method to analyze failure modes associated to the requirements of the system under test. The generation method is based on a generic test model that represents in the same time requirements and their potential failure modes. This model is annotated with RPNs for potential failure modes to generate test cases with their priority numbers based on failure modes and RePNs for requirements to generate test cases based on requirements. Also, we have introduced a new element named generation directives to direct test generation if it is based on the requirements or failure modes. We concluded that the proposed approach is capable to generate prioritized test cases based on requirements and/or failure modes. As future works, we will extend how to create models that represent requirements and failure modes in testing purpose and how to annotate these models by RePN and RPN. Also, we will detail the generation directives for the automatic test generation and we will adopt an existing test generator to take in consideration our approach' inputs to generate appropriate test cases. And then we will make a case study to illustrate test cases generation by our approach.

Significance Statement

This paper provides a new model based testing approach to overcome some challenges involved in model based test case prioritization. We have developed a new model based testing approach that use Failure Mode and Effect Analysis (FMEA) to prioritize test cases during test generation. The proposed approach aim to generate two types of test cases: Test cases based on requirements and test cases based on potential failure modes.

The generated test cases in both types are associated with their priority number to rank test cases during execution and increase the rate of fault detection. An increased rate of fault detection can give an earlier status of the system under test in order to find and correct bugs as soon as possible.

Acknowledgment

I would like to express here the very thanks to my dissertation advisor, Prof. Dr. Marzak Abdelaziz, University Hassan II, who provided me the opportunity to do such a research in his laboratory.

Author's Contributions

ATIFI Meriem: He proposed this approach of using, he participated in all experiments, coordinated the data-analysis and contributed to the writing of the manuscript. He designed the research plan and organized the study.

MARZAK Abdelaziz: He participated in the design of the research plan and organized the study. He coordinated and validated the research.

Ethics

On behalf of my co-author, we inform you that this article has no ethical issues, we confirm that it is an original work which hasn't been published elsewhere. Each author has personally and actively reads this work before submission and that in no case will there be an ethical issues.

References

Acharya, A.A. and S.K. Jena, 2010. Component interaction graph: A new approach to test component composition. J. Comput. Sci. Eng.

Ashraf, E., A. Rauf and K. Mahmood, 2012. Value based regression test case prioritization. Proceedings of the World Congress on Engineering and Computer Science, Oct. 24-26, San Francisco, USA, pp: 24-26.

Atifi, M., A. Mamouni and A. Marzak, 2017. A comparative study of software testing techniques. Proceedings of the International Conference on Networked Systems, May 17-19, Marrakech, Morocco, pp: 373-390. DOI: 10.1007/978-3-319-59647-1_27

Belli, F. and N. Gökçe, 2010. Test Prioritization at Different Modeling Levels. In: Advances in Software Engineering, Kim, T., H.K. Kim, M.K. Khan, A. Kiumi and W. Fang *et al*. (Eds.), Springer, Berlin, Heidelberg, pp: 130-140.

Ben-Daya, M. and A. Raouf, 1996. A revised failure mode and effects analysis model. Int. J. Quality Reliability Manage., 13: 43-47. DOI: 10.1108/02656719610108297

Ben-Daya, M., 2009. Failure Mode and Effect Analysis. In: Handbook of Maintenance Management and Engineering, Ben-Daya, M., S.O. Duffuaa, A. Raouf, J. Knezevic and D. Ait-Kadi (Eds.), Springer, London, UK, pp: 75-90.

Blom, J., B. Jonsson and S.O. Nystrom, 2016. Industrial evaluation of test suite generation strategies for model-based testing. Proceedings of the IEEE 9th International Conference on Software Testing, Verification and Validation Workshops, Apr. 11-15, IEEE Xplore Press, Chicago, IL, USA, pp: 209-218. DOI: 10.1109/ICSTW.2016.42

Chauhan, N., 2018. A Multi-factored Cost- and Code Coverage-Based Test Case Prioritization Technique for Object-Oriented Software. In: Software Engineering: Advances in Intelligent Systems and Computing, Hoda, M., N. Chauhan, S. Quadri and P. Srivastava (Eds.), Springer, Singapore, pp: 27-36.

Elbaum, S., A. Malishevsky and G. Rothermel, 2001. Incorporating varying test costs and fault severities into test case prioritization. Proceedings of ICSE, Toronto, pp: 329-338. DOI: 10.1109/ICSE.2001.919106

Gebizli, C.S. and H. Sozer, 2017. Automated refinement of models for model-based testing using exploratory testing. Software Qual. J., 25: 979-1005. DOI: 10.1007/s11219-016-9338-2

Gökçe, N., F. Belli, M. Eminli and B.T. Dincer, 2015. Model-based test case prioritization using cluster analysis: A soft-computing approach. Turk. J. Electr. Eng. Comput. Sci., 23: 623-640. DOI: 10.3906/elk-1209-109

Graf-Brill, A. and H. Hermanns, 2017. Model-based testing for asynchronous systems. Proceedings of the International Workshop on Formal Methods for Industrial Critical Systems, (ICS' 17), Springer, Cham, pp: 66-82. DOI: 10.1007/978-3-319-67113-0_5

Hettiarachchi, C., H. Do and B. Choi, 2016. Risk-based test case prioritization using a fuzzy expert system. Inform. Software Technol., 69: 1-15. DOI: 10.1016/j.infsof.2015.08.008

Huang, R., W. Zong, D. Towey, Y. Zhou and J. Chen, 2017. An empirical examination of abstract test case prioritization techniques. Proceedings of the IEEE/ACM 39th International Conference on Software Engineering Companion, May 20-28, IEEE Xplore Press, Buenos Aires, Argentina, pp: 141-143. DOI: 10.1109/ICSE-C.2017.105

Huang, R., Y. Zhou, W. Zong, D. Towey and J. Chen, 2017. An empirical examination of abstract test case prioritization techniques. Proceedings of the IEEE 41st Annual Computer Software and Applications Conference, May 20-28, IEEE Xplore Press, Buenos Aires, Argentina, pp: 3-12. DOI: 10.1109/ICSE-C.2017.105

Huang, Y.C., K.L. Peng and C.Y. Huang, 2012. A history-based cost-cognizant test case prioritization technique in regression testing. J. Syst. Software, 85: 626-637. DOI: 10.1016/j.jss.2011.09.063

James, B., 1999. Risk and requirements-based testing. Computer, 32: 113-114. DOI: 10.1109/MC.1999.10066

Kavitha, R. and N. Sureshkumar, 2010. Test case prioritization for regression testing based on severity of fault. Int. J. Comput. Sci. Eng., 2: 1462-1466.

Kim, J.M. and A. Porter, 2002. A history-based test prioritization technique for regression testing in resource constrained environments. Proceedings of the 24th International Conference on Software Engineering, May 19-25, ACM, Orlando, Florida, pp: 119-129. DOI: 10.1145/581339.581357

Korel, B. and G. Koutsogiannakis, 2009. Experimental comparsion of code based and model based test prioritization. Proceedings of the International Conference on Software Testing, Verification and Validation Workshops, Apr. 1-4, IEEE Xplore Press, Denver, CO, USA, pp: 77-84. DOI: 10.1109/ICSTW.2009.45

Korel, B., G. Koutsogiannakis and L.H. Tahat, 2008. Application of system models in regression test suite prioritization. Proceedings of the 24th IEEE International Conference Software Maintenance, Sept. 28-Oct. 4, IEEE Xplore Press, Beijing, China, pp: 247- 256. DOI: 10.1109/ICSM.2008.4658073

Li, Z., M. Harman and R.M. Hierons, 2007. Search algorithms for regression test case prioritization. IEEE Trans. Software Eng., 33: 225-237. DOI: 10.1109/TSE.2007.38

Mahali, P. and D.P. Mohapatra, 2018. Model based test case prioritization using UML behavioural diagrams and association rule mining. Int. J. Syst. Assurance Eng. Manage., 9: 1063-1079. DOI: 10.1007/s13198-018-0736-7

Maia, C.L.B., R.A.F. do Carmo, F.G. de Freitas, G.A.L. de Campos and J.T. de Souza, 2010. Automated test case prioritization with reactive GRASP. Adv. Software Eng. DOI: 10.1155/2010/428521

Malangave, P. and D.B. Kulkarni, 2008. Efficient test case prioritization in regression testing.

Malhotra, R. and A. Bharadwaj, 2012. Test case prioritization using genetic algorithm. Int. J. Comput. Sci. Inform., 2: 63-66.

Mishra, D.B., N. Panda, R. Mishra and A.A. Acharya, 2018. Total fault exposing potential based test case prioritization using genetic algorithm. Int. J. Inform. Technol. DOI: 10.1007/s41870-018-0117-0

Mohanty, S., A.A. Acharya and D.P. Mohapatra, 2011. A survey on model based test case prioritization. Int. J. Comput. Sci. Inform. Technol., 2: 1042-1047.

Panigrahi, C.R. and R. Mall, 2010. Model-based regression test case prioritization. ACM SIGSOFT Software Eng. Notes, 35: 1-7. DOI: 10.1145/1874391.1874405

Peleska, J., 2013. Industrial-strength model-based testing-state of the art and current challenges. Proceedings of the 8th Workshop on Model-Based Testing, Mar. 17-17, Rome, Italy, pp: 3-28. DOI: 10.4204/EPTCS.111.1

Pretschner, A., 2005. Chap Model-based testing in practice. Proceedings of the International Symposium of Formal Methods Europe, Springer, Berlin, pp: 537-541.

Rhmann, W. and V. Saxena, 2017. Fuzzy expert system based test cases prioritization from UML state machine diagram using risk information. Int. J. Math. Sci. Comput., 3: 17-27. DOI: 10.5815/ijmsc.2017.01.02

Rothermel, G., R.H. Untch, C. Chu and M.J. Harrold, 1999. Test case prioritization: An empirical study. Proceedings of the IEEE International Conference on Software Maintenance, Aug. 30-Sept. 3, IEEE Xplore Press, Oxford, England, UK, UK, pp: 179-188. DOI: 10.1109/ICSM.1999.792604

Rothermel, G., R.H. Untch, C. Chu and M.J. Harrold, 2001. Prioritizing test cases for regression testing. IEEE Trans. Software Eng., 27: 929-948. DOI: 10.1109/32.962562

Shin, S.Y., S. Nejati, M. Sabetzadeh, L.C. Briand and F. Zimmer, 2018. Test case prioritization for acceptance testing of cyber physical systems: A multi-objective search-based approach. Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, Jul. 16-21, ACM, Amsterdam, Netherlands, pp: 49-60. DOI: 10.1145/3213846.3213852

Srikanth, H. and L. Williams, 2002. Requirements-based test case prioritization. IEEE Trans. Software Eng.

Srikanth, H. and L. Williams, 2005. Requirements-based test case prioritization. ACM SIGSOFT Software Eng.

Srikanth, H., L. Williams and J. Osborne, 2005. System test case prioritization of new and regression test cases. Proceedings of the International Symposium on Empirical Software Engineering, Nov. 17-18, IEEE Xplore Press, Noosa Heads, Qld., Australia, pp: 10-10. DOI: 10.1109/ISESE.2005.1541815

Srikanth, H., C. Hettiarachchi and H. Do, 2016. Requirements based test prioritization using risk factors: An industrial study. Inform. Software Technol., 69: 71-83. DOI: 10.1016/j.infsof.2015.09.002

Srivastava, P.R., 2008. Test case prioritization. J. Theor. Applied Inform. Technol.

Srivastva, P.R., K. Kumar and G. Raghurama, 2008. Test case prioritization based on requirements and risk factors. ACM SIGSOFT Software Eng. Notes, 33: 7-7. DOI: 10.1145/1384139.1384146

Sultan, Z., R. Abbas, S.N. Bhatti and S.A.A. Shah, 2017. Analytical review on test cases prioritization techniques: An empirical study. Int. J. Adv. Comput. Sci. Applic. DOI: 10.14569/IJACSA.2017.080239

Tahat, L., B. Korel, G. Koutsogiannakis and N. Almasri, 2017. State-based models in regression test suite prioritization. Software Quality J., 25: 703-742. DOI: 10.1007/s11219-016-9330-x

Utting, M. and B. Legeard, 2007. Practical Model-Based Testing: A tools Approach. Morgan Kaufmann Publishers Inc., San Francisco, CA.

Utting, M., A. Pretschner and B. Legeard, 2012. A taxonomy of model-based testing approaches. Software Test. Verificat. Reliability, 22: 297-312. DOI: 10.1002/stvr.456

Varun Kumar, S. and M. Kumar, 2010. Test case prioritization using fault severity. IJCST.

Wang, R., L.M. Kristensen, H. Meling and V. Stolz, 2017. Application of model-based testing on a quorum-based distributed storage. PNSE, 17: 177-196.

Wang, Y., Z. Zhu, B. Yang, F. Guo and H. Yu, 2018. Using reliability risk analysis to prioritize test cases. J. Syst. Software, 139: 14-31. DOI: 10.1016/j.jss.2018.01.033

Wu, Y., D. Pan and M. Chen, 2001. Techniques for testing component-based software. Proceedings of the 7th IEEE International Conference on Engineering of Complex Computer Systems, pp: 222-232.

Zhang, T., X. Wang, D. Wei and J. Fang, 2018. Test case prioritization technique based on error probability and severity of UML models. Int. J. Software Eng. Knowl. Eng., 28: 831-844. DOI: 10.1142/S0218194018500249