GU J Sci, Part A, 8(1): 1-15 (2021)

Gazi University

Journal of Science

PART A: ENGINEERING AND INNOVATION

http://dergipark.gov.tr/gujsa

Derleme Makale Review Article

An Introduction to Software Testing Methodologies

A. Ziya AKTAŞ[1*], Eray YAĞDERELİ[1], Doğa SERDAROĞLU[2]

[1]*Başkent University, Faculty of Engineering, Computer Engineering Department 06810, Ankara, Turkey*
[2]*Credit Suisse Poland - Szczytnicka 9, 50-382 Wrocla, Poland*

Keywords

PRISMA
SDLC
SW Testing Methodology
TMMi

Abstract

It seems to the authors that 'testing' for many of the Computer Engineering or Software Engineering professionals or to laymen means the same as 'code testing' or 'program testing'. Yet, recalling the SDLC (Software Development Life Cycle) steps such as Planning, Analysis, Design, Implementation or Construction and Maintenance, obviously will tell us that coding is just a piece of SDLC included in the Implementation. Therefore, unit testing, integration testing, black box texting or acceptance testing etc. would not cover the whole SDLC. One then needs any methodology, if available, to encompass all steps of SDLC.
The major objective of this article has been to investigate the available methodologies applicable for software testing. During our research, we met TMMi (Test Maturity Model integration) and PRISMA (Product Risk Management) methodologies available in the professional arena. After understanding the characteristics of these methodologies, we attempted to propose a new methodology as a synthesis of these two methodologies.
The details of these two methodologies are given in this article, while a newly proposed methodology described in the article will be applied on a web based problem to be reported later.

Cite

Aktaş, A. Z., Yağdereli, E. & Serdaroğlu, D., (2021). An Introduction to Software Testing Methodologies. *GU J Sci, Part A, 8*(1), 1-15.

Author ID (ORCID Number)	Article Process	
A. Z. Aktaş, 0000-0002-1829-7821	**Submission Date**	28.01.2019
E. Yağdereli, 0000-0002-5384-3912	**Revision Date**	18.12.2020
D. Serdaroğlu, 0000-0001-5705-8551	**Accepted Date**	24.03.2021
	Published Date	29.03.2021

1. INTRODUCTION

1.1. Preliminaries

In the early years of digital computers, testing process had focused on hardware [Gelperin & Hetzel **(**1988) as an example]. The major concern was hardware reliability and software was then checked only for that concern. ["Once a programmer wrote a program, the differences between the concepts of program 'checkout', 'debugging', and 'testing' were not clear at that time. Baker distinguished in 1957 first, 'testing' from 'debugging'". This part is taken from the articles of Baker (1957) and Alam & Khan (2013) as an example].

As we all know now that computer applications have become an intrinsic part of our lives and virtually every business in every sector have depended on software. Software almost in all fields have increased in number, size and complexity and thus, testing has become more significant because of its greater economic risk. Software industry has been investing substantial effort and money to get higher quality products. In fact, software engineer is now defined as an engineer whose objective is to develop cost-effective and high quality software. However, software engineering field is still far from expected results. In addition to this problem, diversification of application platforms with mobile computing, customer demands to access their

*Corresponding Author, e-mail: zaktas@baskent.edu.tr data and applications from everywhere and competitive market pressures are facts to live with. Additionally, very high amount of data / information and knowledge that is shared between applications have made the problem worse (Aktaş, 1987; RTI, 2002).

Having a general introduction of software testing methodologies is the major objective of this article. After understanding the characteristics of the available methodologies and summarizing them, a new methodology is proposed briefly.

In the first Chapter, namely 1. INTRODUCTION, the article briefly introduces some preliminaries which is followed by subsections that deal with software-testing related concepts such as Risk Management, Capability Maturity Model Integration (CMMI), Test Maturity Model integration (TMMi), and Product Risk Management. Their elaborations are given in the Chapters 2-4 of the article. Chapter 5 has a proposed methodology for software testing. Article ends in Section 6: SUMMARY, CONCLUSIONS and FUTURE WORK.

1.2. Software Testing and Risk Management

As noted by Broadleaf Capital International PTY Ltd (2012) "Risk is not something new to human being. Yet, this knowledge is turned into a science only first in the last century". This topic is elaborated in the next Chapter, Chapter 2.

1.3. Software Testing and CMMI (Capability Maturity Model Integration)

The software industry has often tried to improve its development methodologies in order to improve its product quality. A guideline that has been widely used for this purpose is known as the Capability Maturity Model (CMM). Black (2011) has claimed that "The CMM and the Capability Maturity Model Integration (CMMI) have often been regarded as a de-facto standard in the market for software development process improvement". ICT (Information and Communications Technology) spending on quality assurance (QA) and testing has been rising steadily. ICT budget allocation for application quality and testing in software development field has grown from 18 percent to 35 percent between 2012 and 2015 and it is expected to be about 40 percent in the year 2018 (Capgemini, Sogeti & HP, 2015). Yet, CMM or CMMI has given only a limited attention to testing (van Veenendaal & Cannegieter, 2013). The CMMI has two dedicated process areas (Verification, VER and Validation, VAL). They provide more focus on testing at maturity Level 3 and they are summarized and presented in Table 1 (CMMI, 2010).

Therefore, testing community has created their own test process improvement models such as Test Maturity Model integration (TMMi) and Test Process Improvement (TPI) and other test methodologies.

Because risk and risk management are closely related concepts to software testing, the next Chapter, namely Chapter 2, is devoted to this topic, as noted earlier.

Table 1. *CMMI Process Areas Related with Testing*

Process Area	**Description**
Verification – VER	Aims to ensure that selected work products would meet the specified requirements of the product.
Validation – VAL	Incrementally validates products against customer's requirements

1.4. Software Testing and TMMi (Test Maturity Model integration)

"TMMi (Test Maturity Model integration) was developed to serve as a guideline and a reference framework for test process improvement. It functions as a complementary model for CMMI to serve test managers, test engineers and software quality professionals" (CMMI, 2010; van Veenendaal, 2012a,b). TMMi is a generic model and it is applicable to various life cycle models in various environments. One may also implement TMMi for test process improvement even for agile software development methodologies.

1.5. Software Testing and Risk-Based Testing (RBT) and PRISMA (Product Risk Management)

The British Standards Institute has produced the ***BS 7925/2*** standard which has two sections as test design techniques and test measurement techniques. One may round out the list of techniques as "Random Testing" and "Other Testing Techniques". Random Testing (RT) has been the primary software-testing technique for decades. Its effectiveness in detecting software failures has been often controversial (Liu et al., 2011a,b; van Veenendaal, 2014). As a result of these concerns, Risk-based testing (RBT) concept has evolved from "good enough quality" and "good enough testing" concepts and "heuristic risk-based testing" (Bach, 1997; 1998; Liu et al., 2011a,b). Its fundamental principles can be listed as (Bach, 1999):

- Exhaustive testing (exercising every possible path and every combination of inputs and preconditions) is infeasible except for trivial cases;
- Selective testing techniques are needed to allocate test resources properly;
- The focus of testing has to be placed on features and aspects of software to reduce the failure risk and to ensure the quality of the applications.

As noted by Graham et al. (2008) "These can be achieved by applying the principle of risk-based prioritization of tests, known as Risk-Based Testing (RBT). The primary role of risk-based testing is to optimize available resources and time without affecting the quality of the product".

The PRISMA model is a risk-based testing approach and developed in a bottom-up fashion (van Veenendaal, 2011, 2012a,b). It depends on:

- Identifying the areas that have the highest level of business and/or technical risk;
- Assessing the impact of possible defects and the likelihood of the occurrence of these defects;
- Visualizing product risks (test items) in a product risk matrix by assigning numeric values to both impact and likelihood.

2. SOFTWARE TESTING AND RISK MANAGEMENT

2.1. General

Due to lack of a common and comprehensive international standard, many varieties of standards or regulations have been developed during the last two decades. Finally, the first global risk management standard ***ISO 31000:2009*** (ISO, 2009a) was released. A list of important risk management standards were given by Ciocoiu & Dobrea (2010).

2.2. The Definition of Risk

As noted above, the first global risk management standard is ***ISO 31000:2009*** (ISO, 2009a). It is together with a revised set of definitions that are contained in ***ISO/IEC Guide 73***:***2009*** (ISO, 2009b).

The definition of risk in the ***ISO 31000*** standard is "the effect of uncertainty on objectives". Guide 73 also states that "an effect may be positive, negative or a deviation from the expected, and that risk is often described by an event, a change in circumstances or a consequence" (ISO, 2009a).

Organizations often have to contend with internal and external factors and influences which they may not control and that generate uncertainty and thus risk. Such factors might assist or speed up the achievement of objectives; but also prevent or delay the organization achieving its objectives.

2.3. Risk Management Process (RMP)

According to ISO 31000:2009, risk management process is defined as: "systematic application of management policies, procedures and practices to the activities of communicating, consulting, establishing the context, and identifying, analyzing, evaluating, treating, monitoring and reviewing risk" (ISO, 2009a). The main elements and overview of the risk management process are given as Figure 1.

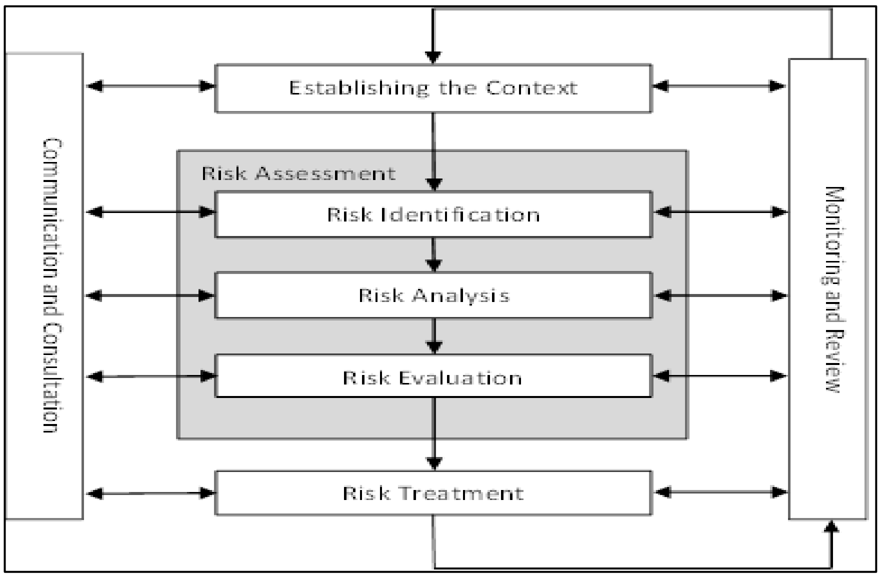

Figure 1. *Main Elements of the Risk Management Process*

2.4. Risk Characterization and Measurement

Risk analysis phase of risk management process involves consideration of the causes and sources of risk, their positive and negative consequences, and the likelihood that those consequences can occur. Fundamentally, risk is characterized and measured by considering consequences of what could happen and the likelihoods of those consequences. From the definition, risk is a function of both likelihood and a measure of consequence. In its simplest form risk can be defined as (Standards Australia/New Zealand, 2004):

$$\textbf{\textit{Risk = A function of (Consequence and Likelihood)}} \tag{1}$$

Consequences and their likelihoods are often combined to define a level of risk. The way in which consequences and likelihood are expressed and the way in which they are combined to reflect the type of risk and the purpose for which the risk assessment output is to be used. These should all be consistent with the risk criteria. Risk analysis can be qualitative, semi-quantitative or quantitative, or a combination of these, depending on the risk, the purpose of the analysis, and available data and resources.

Qualitative analysis is any method of analysis which uses description rather than numerical means to define a level of risk. This information may be brought together and summarized as simple descriptions of consequences and likelihoods of risks to use in a risk inventory table. However, criteria on which the ranking can be based must be decided and/or devised.

If it is taken that the level of risk is proportional to each of its two components (consequence or likelihood) the risk function simply becomes a product. As given in ***AS/NZS 4360:2004***, this can be presented mathematically as (Standards Australia/New Zealand, 2004):

$$\textbf{\textit{Risk = Consequence}} \times \textbf{\textit{Likelihood or (R = C}} \times \textbf{\textit{L)}} \tag{2}$$

If the level of risk is not linearly dependent on one or two of its components (consequence and likelihood), the risk function should include required weighting factor(s) and/or exponential operator for one or both of the components (Roy, 2004).

3. CAPABILITY MATURITY MODEL (CMM) AND CAPABILITY MATURITY MODEL INTEGRATION (CMMI)

As noted earlier, a methodology named TMMi (Test Maturity Model integration) has an objective of supporting test engineers in evaluating and improving their test processes. In developing this methodology two other methodologies, namely CMMI (Capability Maturity Model Integration) and TMM (Test Maturity Model) were used as guidelines (van Veenendaal & Cannegieter, 2013).

Capability Maturity Models are based on the premise of "the quality of a system or product is highly influenced by the quality of the process used to develop and maintain it" (van Veenendaal, 2012a,b). CMMs including CMMI focus on improving processes in an organization. They contain the essential elements of effective processes and describe an evolutionary improvement path (through capability/maturity levels) from ad hoc, immature processes to disciplined, mature processes with improved quality and effectiveness (van Veenendaal, 2012a,b).

In the context of software development, the capability maturity model is a group of related strategies for improving the software development process, irrespective of the actual software life-cycle model used. Capability/Maturity Levels are used in CMMI for (Software) Development to describe an evolutionary path recommended for an organization that wants to improve the processes it uses to develop software products or services. Maturity Levels of the CMMI for Development are given in the Figure 2 (CMMI, 2010).

In the context of software development, the capability maturity model is a group of related strategies for improving the software development process, irrespective of the actual software life-cycle model used. Capability/Maturity Levels are used in CMMI for (Software) Development to describe an evolutionary path recommended for an organization that wants to improve the processes it uses to develop software products or services. Maturity Levels of the CMMI for Development are given in the Figure 2 (CMMI, 2010).

4. SOFTWARE TESTING AND TMMi (TEST MATURITY MODEL INTEGRATION)

4.1. General

Testing process in the TMMi (Test Maturity Model integration) includes all software product quality-related activities. TMMi also uses the concept of maturity levels for process evaluation and improvement similar to the CMMI stages. Each TMMi maturity level also includes process areas, goals and practices. Thus, "the focus of testing changes from defect detection to defect prevention" (Capgemini, Sogeti & HP, 2015; CMMI Product Team 2010). "One also should note that the model is compliant with ISO 15504 standard, that is, Software Process Improvement and Capability Determination (SPICE) standard, for process assessment" (CMMI, 2010).

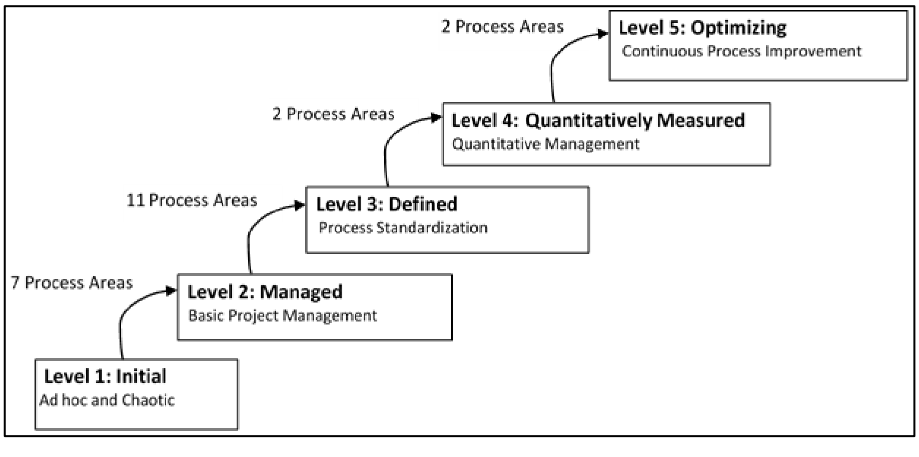

Figure 2. *Maturity Levels of CMMI for Development*

As noted earlier, TMMi similar to CMMI, has a leveled architecture for process improvement. There are five stages and/or levels in the TMMi architecture. Each level/stage has a set of process areas that an organization needs to implement to achieve maturity at that level. Thus, they form a necessary foundation for the next level (van Veenendaal, 2012a,b). TMMi maturity levels and process areas as given in TMMi reference model are presented in Figure 3 (CMMI, 2010). In the following subsections, summary of five levels of TMMi are given (CMMI, 2010; Capgemini-Sogeti-HP, 2015).

4.2. Level 1: Initial

Level 1 of TMMi has no defined process areas. Testing at this level is an undefined process; it is often taken as part of debugging. In order to get the bugs out of the system, testing and debugging are interleaved. The objective of testing at this level is to show that the software runs without major failures. Level 1 of TMMi has no defined process areas. Testing at this level is an undefined process; it is often taken as part of debugging. In order to get the bugs out of the system, testing and debugging are interleaved. The objective of testing at this level is to show that the software runs without major failures.

4.3. Level 2: Managed

Testing becomes a managed process and it is clearly separated from debugging at this level. A company-wide or program-wide test strategy is now established. The process discipline of maturity Level 2 makes it possible that existing practices are made available during times of stress. The main objective of testing at this level is to verify that the product satisfies the specified requirements. Yet, many consider testing as being a project phase that follows coding.

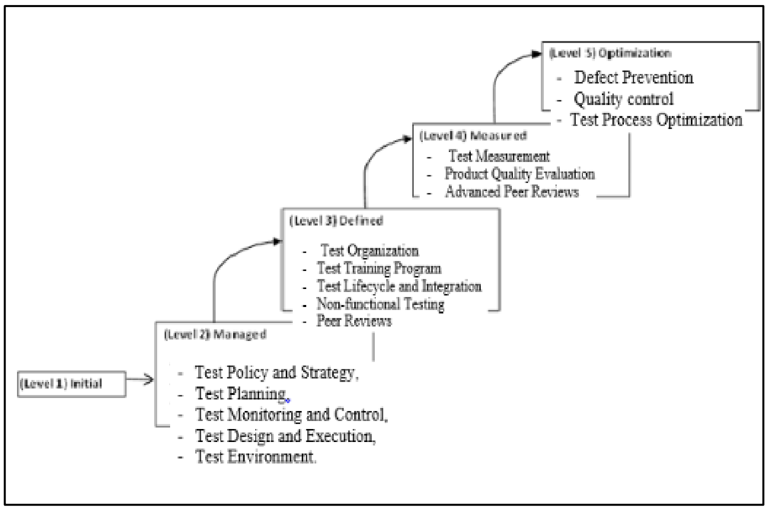

Figure 3. *TMMi Maturity Levels and Process Areas of TMMi*

The process areas at TMMi Level 2 are given as (CMMI, 2010; Capgemini-Sogeti-HP, 2015):

- Test Policy and Strategy,
- Test Planning,
- Test Monitoring and Control,
- Test Design and Execution,
- Test Environment.

4.4. Level 3: Defined

At TMMi Level 3, testing is not a lifecycle phase after implementation. Testing is now completely integrated into the development life-cycle and its associated milestones. Test planning is performed at an early stage of the project, usually during the requirements phase, and is documented by means of a master test plan. Organizations at TMMi Level 3 understand the importance of reviews for developing a quality product. A formal review program is implemented. At Level 3, test designs and techniques are expanded to include nonfunctional testing such as usability and/or reliability, depending on the business objectives, although they focus mainly on functionality testing at TMMi Level 2. The process areas at TMMi Level 3 are given as (CMMI, 2010; Capgemini-Sogeti-HP, 2015):

- Test Organization
- Test Training Program
- Test Lifecycle and Integration
- Non-functional Testing
- Peer Reviews

4.5. Level 4: Management and Measurement

Testing is a thoroughly defined, well-founded and a measurable process at this level. An organization-wide test measurement program is used to evaluate the quality of the testing process, to assess productivity, and to monitor improvements. Measurements are incorporated into the organization's repository. Test measurement program also supports fact-based decision-making and predictions related to test performance and cost.

The presence of a measurement program allows an organization to implement a product- quality evaluation process by defining quality needs, quality attributes such as reliability, usability and maintainability and quality metrics. Product quality is understood in quantitative terms and is managed to the defined objectives throughout the lifecycle.

Reviews and inspections are considered as parts of the test process at this level. The process areas at TMMi Level 4 are given as follows (CMMI, 2010; Capgemini-Sogeti-HP, 2015):

- Test Measurement
- Product Quality Evaluation
- Advanced Peer Reviews

.4.6. Level 5: Optimization

An organization has continuous focus on process improvement and fine-tuning at this level. The organization is capable of continually improving its processes depending on a quantitative understanding of statistically controlled processes. Test process performance is improved through incremental and innovative process and technological improvements. The testing methods and techniques are continuously optimized.

The three process areas at TMMi Level 5 are given as follows (CMMI, 2010; Capgemini-Sogeti-HP, 2015):

- Defect Prevention
- Quality control
- Test Process Optimization

5. SOFTWARE TESTING, RISK-BASED TESTING AND PRISMA APPROACH

5.1. General

Risk management has been the subject of research in software engineering since the works of Boehm (1989) and Charette (1989). Many risk management models and frameworks such as, SERIM (Software Engineering Risk Index Management), SERUM (Software Engineering Risk Understanding and Management), SEI-SRE (Software Engineering Institute - Software Risk Evaluation), Riskit, ProRisk, etc. have been proposed after two studies by Stern & Arias (2011) and Roy (2004). All these studies took risk management in the context of software development and did not address supervising testing activities according to risk management principles. In this respect, James Bach's "heuristic risk-based testing" proposal (1999) which was evolved from "good enough quality" and "good enough testing" concepts (1997-1999), were the first studies applying risk management concepts to testing activities.

The risk-based testing depends on the intuitive idea of to focus (identify and prioritize) test activities on those scenarios that trigger the most critical situations for a software system and ensure that these critical situations are both effectively mitigated and sufficiently tested (Roy, 2004; Wendland et al., 2012). In its first form, the way proposed by Bach (1998), risk-based testing fundamentally depends on heuristic risk analysis which can be performed by using one of two approaches: inside-out or outside-in. Inside-out begins with details about the situation and identify risks associated with them, also known as bottom-up approach. Outside-in begins with a set of potential risks and match them to the details of the situation, a more general top-down approach. According to Bach (1998), risk-based testing aims at testing the right things of a system at the right time and each test process is actually carried out in a risk-based way due to its sampling characteristics. He further states that in most cases, the consideration of risk is rather made implicitly.

After Bach's proposal, Amland's (2000) work was the first study utilizing risk analysis fundamentals and metrics to manage the test process by intelligent assessment by identifying the probability and the consequences of individual risks. In this study, he defined a risk-based testing approach based on Karolak's risk management process (Karolak, 1995). Risk-based testing approach as defined and diagrammed by Amland (2000) is given as Figure 4. Oval boxes in Figure 4 explain how his risk-based testing model fits in with the Karolak's (1995) risk management process. In this regard, Amland's study was considered to be the first to introduce a systematic risk-based testing approach (Felderer & Ramler, 2014).

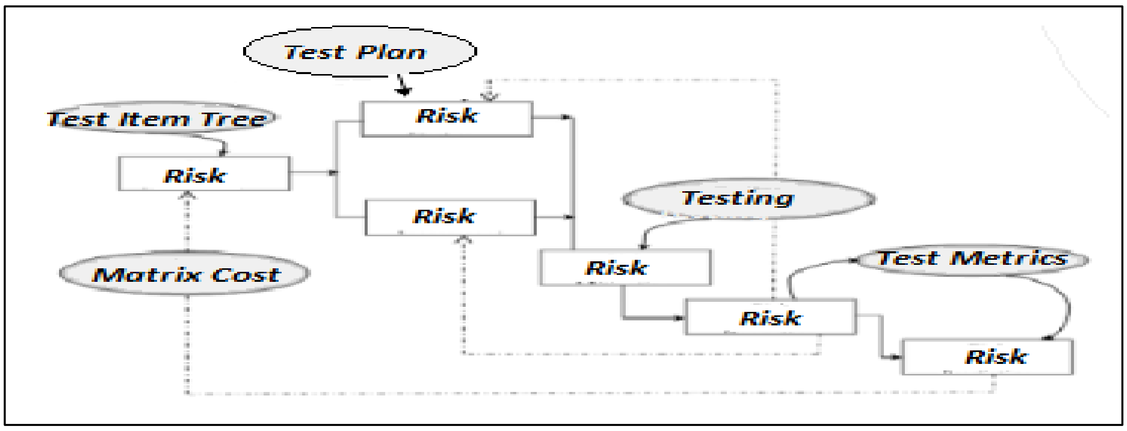

Figure 4. *Risk-Based Testing*

5.2. Related Work on Risk-Based Testing (RBT)

After these two pioneering works (Bach, 1998; Amland, 2000), many risk-based testing approaches have been proposed by academia and industry due to its practical relevance and importance. Because of the growing number of available risk-based testing approaches and its ever increasing usage in industrial test processes there is a need about surveying, comparing and categorizing of all these studies. There are three valuable articles that we identified serving this purpose. The first one is the article by Erdogan et al. (2014). This article presented "a systematic literature review on risk-based testing approaches." They identified a total of 32 peer-reviewed articles. The articles were then grouped into 24 approaches of which only two approaches were found out about test-based risk assessment, while the remaining 22 were classified as addressing various aspects of risk-based testing.

The second article was given by Felderer & Schieferdecker (2014). In this article, "the taxonomy of risk-based testing has been developed by analyzing the works presented by 23 articles and it has been applied to four selected works on risk-based testing."

The third article was given by Felderer et al. (2014, 2015). "This article has focused on how risk estimation is performed in different risk-based testing approaches, since risk estimation determines the importance of the risk values assigned to tests and consequently the quality of the overall risk-based testing process."

5.3. PRISMA (Product Risk Management Approach)

The PRISMA model is a risk-based testing approach developed in bottom-up fashion. It can work at every level of testing (Graham et al., 2008; van Veenendaal, 2011). PRISMA approach utilizes a risk quantification method defined in (ISO 2009b) and in principle similar to the Barry Boehm's concept of "risk exposure"**.** According to Boehm (1989), **"Risk** exposure is defined as the relationship where the probability of an unsatisfactory outcome and the loss due to the unsatisfactory outcome determine the valence of the risk event (the expected value of the risk)" (Standards Australia/New Zealand, 2004). The innovative aspect of the PRISMA approach is its assessment of every risk item from the point of many impact factors and likelihood factors. It groups and calls impact factors as "business risks" (on x axis) and likelihood factors as "technical risks" (on y axis) as shown in Figure 5.

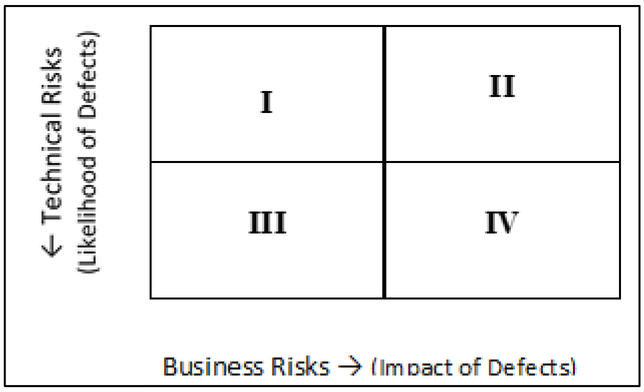

Figure 5. *Product Risk Matrix*

The standard risk matrix is divided into four areas (quadrants I, II, III and IV) as shown in Figure 5. Each represents a different risk level and type. A different risk level/type most likely will lead to a different test approach as documented in a test plan. The product risk matrix provides a general preview of the product risks to the stakeholders and the decision makers. It is much easier to understand a visual overview than a list of numbers.

Although there may be different impact and likelihood factors for every product, common impact factors that are usually utilized to determine and quantify (assess) impact of risk (test) items in PRISMA approach are given as follows (Graham et.al., 2008; van Veenendaal, 2012a,b):

- Critical areas (cost and consequences of failure);
- Visible areas;
- Mostly used areas;
- Importance to business;
- Cost of rework.

Likelihood factors that are often utilized to determine and assess importance of risk (test) items in PRISMA approach are given as (Graham et.al., 2008; van Veenendaal, 2012a,b):

- Complex areas;
- Changed areas;
- New technology and methods;
- People involved ;
- Time pressure;
- Optimization;
- Defect history;
- Geographical spread.

Other likelihood factors to be considered are new development versus re-use, size of application and number of entities interfaced.

The standard PRISMA process consists of the following activities as given in Figure 6 (Graham et.al., 2008; van Veenendaal, 2012a,b).

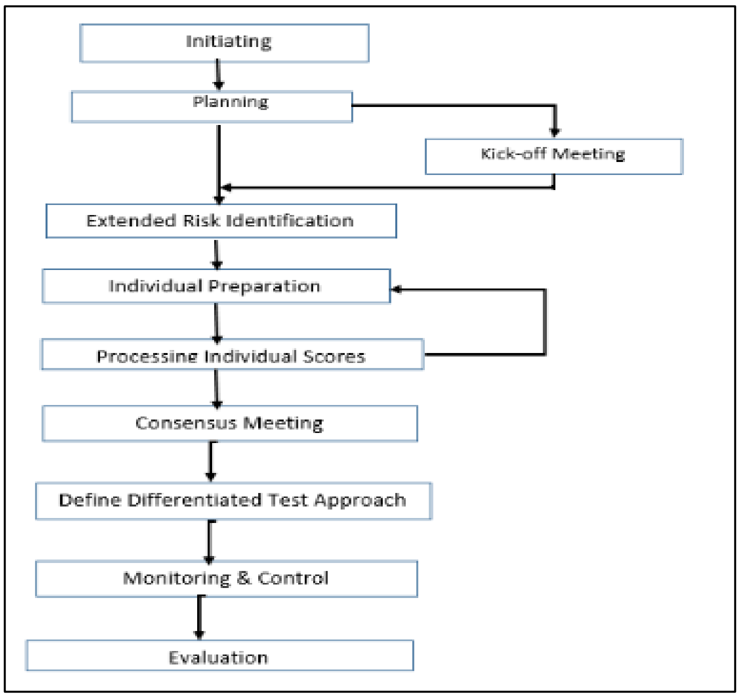

Figure 6. *PRISMA Process Overview*

Each step is outlined briefly in the following subsections:

a) Initiating

The main objectives in the initiating step are definition of the PRISMA process and setting of rules, factors, value ranges and weights for factors. Initiating step includes the following tasks:

- Definition of the PRISMA process to be used in the organization and its documentation;
- Define impact and likelihood factors;
- Determine scoring ranges for factors;
- Define a weight for each factor;
- Scoring rules.

While implementing the PRISMA approach a standard process is defined. Process variations can be implemented for different situations. This process shall be documented and disseminated as a handbook within the organization. The likelihood and the impact factors shall be defined in a comprehensive way for all stakeholders. The scoring range can be defined as 1 – 3 (low as 1, intermediate as 2 and high as 3), or the range of 1 - 5 or 1 - 9. The experiences show that the best range suitable to most cases for scoring is the 1 - 5 range which is divided as low (1-2), intermediate (3-4) and high (5).

b) Planning

In the planning step, scope, stakeholders and product risks are identified and defined. The planning step is essentially carried out by the test manager and includes the following tasks:

- Definition of scope;
- Select stakeholders;
- Gathering input documents;
- Identifying risk items;
- Review of impact and likelihood factors and rules;
- Assignment of factors to stakeholders.

The determined product risk items should be tagged uniquely and expressed in a comprehensive way for the stakeholders. This will help the stakeholders follow the risk items in the future easily. Stakeholders can be chosen among the project managers, developers, software architects, marketers, consumers, business managers. The stakeholders are those who install, use the product and affected by it. The factors are assigned to the stakeholders according to interests and roles of the stakeholders for scoring. Generally, the impact factors are assigned to business-related stakeholders and the likelihood factors to stakeholders on the technical side and they should be explained clearly to stake holders.

c) Kick-off Meeting

After that all assignments are done, the first meeting with all the stakeholders is held and the test manager explains the product evaluation to all the stakeholders. The purpose of the kick-off meeting is to inform all the stakeholders about the process, risk assessment and the purpose of the activities, and to get their commitments. The kick-off meeting is optional but it is highly recommended. The risk items and factors should be explained in detail to stakeholders.

d) Extended Risk Identification

Main objectives of this step are to extend identified risk set, to verify risk set determined in planning step, to identify inconsistencies between requirements and risks. During this step, brainstorming sessions are used to verify and extend the initial risk set.

e) Individual Preparation

During the individual preparation step, participants score and register the impact and likelihood factors for each registered risk item. The values assigned by the stakeholder are based on perceived risks which are stakeholder's expectation that something could go wrong (likelihood of defects) and importance for the business (impact of defects). For the stakeholders not to be affected by the other stakeholders, the scoring process should be performed independently.

As a result of this step, "documents containing the scoring values of the assigned factors of each participant will be obtained."

f) Processing Individual Scores

The purpose of this step is summarized as follows:

- To verify and confirm that every participant scored factors assigned to him/her;
- To input and update draft product risk matrix with individual scores;
- To check whether scorings have been done correctly and identify violation of rules;
- To prepare an item list to be discussed later in the consensus meeting.

The test manager gathers the individual results of the stakeholders, controls them and puts them in process. The average value for each risk item is calculated for all of its likelihood and impact factors. A weighted sum of average values of likelihood and impact factors for each product risk item will be computed and a draft scoring total for the likelihood and impact factors will be gathered. After this calculation, each risk item is to be positioned in the product risk matrix.

g) Consensus Meeting

The resulting product risk matrix is analyzed by the stakeholders in the consensus meeting. During the meeting the items from the items list are discussed between stakeholders and test manager. At the end of the discussion final scores are determined and product risks are positioned in the resulting risk matrix as shown in Figure 7 (Serdaroğlu, 2015).

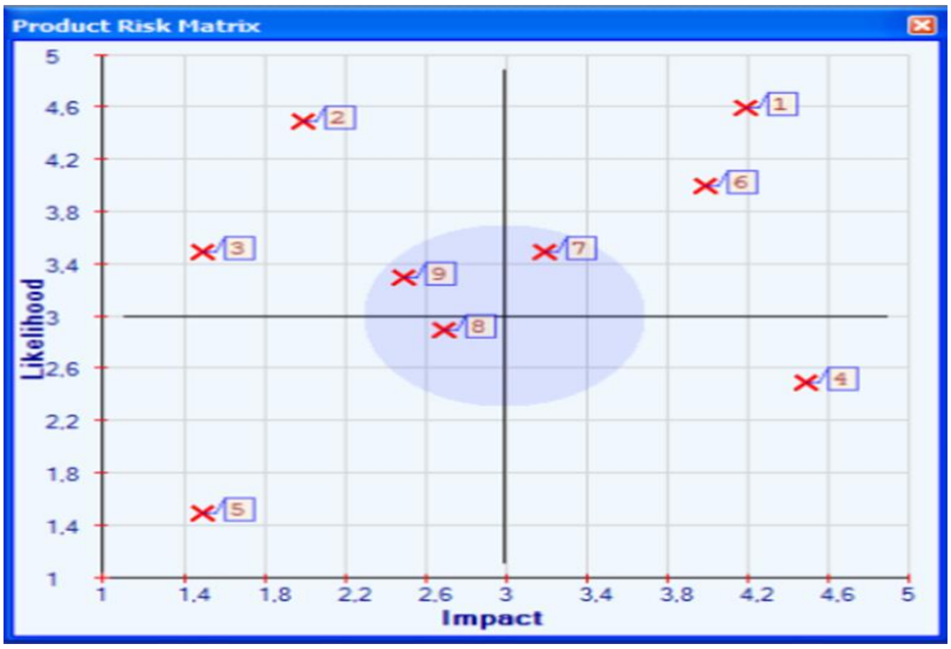

Figure 7. *Example Product Risk Matrix*

The resulting product risk matrix should always be validated with the stakeholders. If results are not according to the expectations of the stakeholders, they should be re-discussed to reach a consensus. If needed, a follow-up meeting or a specific discussion meeting in a smaller group could be organized (Serdaroğlu, 2015).

Some risk items are on the edges of the matrix. Items with low impact and low probability such as the risk item 5 are negligible. Items with a high impact and high probability such as the item 1 should be paid more attention in the test process. It is much harder to decide about the risk factors near the center of the matrix. Because it can be difficult to determine which area they belong to. Normally all the risk items within the circle are contained in the subject list of the test manager and they have to be discussed and analyzed. After the analysis, it should be decided into which quarter they are going to be placed.

h) Define Differentiated Test Approach

At this step a test approach is defined according to agreed product risk matrix (based on test items' position in the risk matrix). Main tasks of this step are defined as follows (Serdaroğlu, 2015):

- Make prioritization of test items;
- Identify test depth for each test item;
- Determine test design techniques.

A certain test approach is going to be determined for each quarter. It contains decisions about test design techniques to be applied, test and evaluation types to be held.

i) Monitoring & Control

Test monitoring and control, including reporting, should be organized around product risks. For technical risks (likelihood), technical reviews are more helpful, whereas for business risks (impact), it is more useful to determine the ways to be followed together with customer (Serdaroğlu, 2015).

j) Evaluation

Risks are not static and ever changing during test process. Therefore risk identification and assessment needs to be reviewed regularly and it has to be updated according to the results of tests. During the development of project and execution of the test, the test manager has to maintain the matrix, based on lessons learned e.g. defects found or other measured indicators like DDP (Defect Detection Percentage), (changed) assumptions, updated requirements or architecture. "'Changes in the projects' scope, context or requirements will often require updates to the risk assessment" (Serdaroğlu, 2015).

Additionally, the test manager and the project manager can analyze product risk matrix together and use this information to determine the priorities of development, integration and the service schedule accordingly.

6. SUMMARY, CONCLUSIONS AND FUTURE WORK

6.1. Summary

This article includes an introductory study on the software testing methodologies. Two of the relevant methodologies, namely TMMi and PRISMA were studied in detail for developing another methodology as a synthesis of two, to apply on a real-world case study.

6.2. Conclusions

According to TMMi, testing becomes a managed process at Level 2, and testing approach is defined based on the result of a product risk assessment performed in the test policy and strategy process area of this level. The authors, therefore, have proposed incorporation of PRISMA approach into the TMMi framework at Level 2. The proposed incorporation of PRISMA to the TMMi model will augment efficiency and affectivity of testing process in terms of time and budget, since it allows allocation of valuable test resources on the areas that have the highest level of risk.

6.3. Future Work

The proposed hybrid methodology based on both TMMi and PRISMA will be applied on a web based real life problem and its results will be reported later.

Three of the recent publications such as Patrício et al. (2021) A Study on Software Testing Standard Using ISO/IEC/IEEE 29119-2: 2013. In: Al-Emran M., Shaalan K., Hassanien A. (eds) Recent Advances in Intelligent Systems and Smart Applications. Studies in Systems, Decision and Control, Vol 295. Springer, Cham.; Laksono et al. (2019) Assessment of Test Maturity Model: A Comparative Study for Process Improvement; and Hrabovská et al. (2019) Software Testing Process Models Benefits & Drawbacks: a Systematic Literature Review will be studied in detail and they will be added to the references of a next expected publication.

CONFLICT OF INTEREST

No conflict of interest was declared by the authors.

No fund support is received during this study.

ACKNOWLEDGEMENT

The authors express their wishes to acknowledge their cordial thanks and gratitude to the Editors and Anonymous Reviewers for their kind comments and supports on the article.

REFERENCES

Aktaş, A. Z. (1987). Structured Analysis and Design of Information Systems, Prentice-Hall, USA.

Alam, M., & Khan, A. I. (2013). Risk-Based Testing Techniques: A Perspective Study. *International Journal of Computer Applications*, *65*(1), 33-41.

Amland, S. (2000). Risk-Based Testing: Risk Analysis Fundamentals and Metrics for Software Testing Including a Financial Application Case Study. *Journal of Systems and Software*, *53*(3), 287-295. doi:10.1016/S0164-1212(00)00019-4

Bach, J. (1997). Good Enough Quality: Beyond the Buzzword. *IEEE Computer*, *30*(8), 96-98. doi:10.1109/2.607108

Bach, J. (1998). A Framework for Good Enough Testing. *IEEE Computer*, *31*(10), 124-126. doi:10.1109/2.722304

Bach, J. (1999). Risk-Based Testing: How to conduct heuristic risk analysis. *Software Testing and Quality Engineering Magazine*, *November/December*, 23-28. www.satisfice.com/articles/hrbt.pdf

Baker, C. L. (1957). Review of "Digital Computer Programming, by D.D. McCracken". *Mathematical Tables and Other Aids to Computation*, 11(60), 298-305. doi:10.2307/2001950

Black, R. (2011). Advanced Software Testing, Vol.1, Rocky Nook, USA.

Boehm, B. W. (1989). Tutorial: Software Risk Management, IEEE Computer Society Press, New York.

Broadleaf (2012). A Simple Guide to Risk and Its Management. Broadleaf Capital International Pty Ltd. (Accessed:21/02/2017) broadleaf.com.au/resource-material/a-simple-guide-to-risk-and-its-management/

Capgemini, Sogeti & HP (2015). World Quality Report 2015-16 Seventh Edition.

Charette, R. N. (1989). Software Engineering Risk Analysis and Management. McGraw-Hill, New York.

Ciocoiu, C. N., & Dobrea, R. C. (2010). The Role of Standardization in Improving the Effectiveness of Integrated Risk Management. In: Nota, G. (Eds.) Advances in Risk Management (pp. 1-18). IntechOpen. doi:10.5772/9893

CMMI (2010). CMMI for Development, Version 1.3. (Technical Report: CMU/SEI-2010-TR-033). CMMI Product Team, Software Engineering Institute, Carnegie Mellon University. doi:10.1184/R1/6572342.v1

Erdogan, G., Y. Li, Y., Runde, R. K., Seehusen, F., & Stølen, K. (2014). Approaches for the combined use of risk analysis and testing: a systematic literature review. *International Journal on Software Tools for Technology Transfer*, *16*(5), 627-642. doi:10.1007/s10009-014-0330-5

Felderer, M., & Ramler, R. (2014). Integrating risk-based testing in industrial test processes. *Software Quality Journal*, *22*(3), 543-575. doi:10.1007/s11219-013-9226-y

Felderer, M., & Schieferdecker, I. (2014). A taxonomy of risk-based testing. *International Journal on Software Tools for Technology Transfer*, *16*(5), 559-568. doi:10.1007/s10009-014-0332-3

Felderer, M., Haisjackl, C., Pekar, V., & Breu, R. (2014). A Risk Assessment Framework for Software Testing. In: Margaria, T., & Steffen, B. (Eds.), Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications. Proceedings of the 6th International Symposium, ISoLA 2014, Part II, (pp. 292-308), October 8-11, Corfu, Greece. doi:10.1007/978-3-662-45231-8_21

Felderer, M., Haisjackl, C., Pekar, V., & Breu, R. (2015). An Exploratory Study on Risk Estimation in Risk-Based Testing Approaches. In: Winkler, D., Biffl, S., & Bergsmann, J. (Eds.), Software Quality. Software and Systems Quality in Distributed and Mobile Environments. Proceedings of the 7th International Conference, SWQD 2015, (pp. 20-23), January 20-23, Vienna, Austria. doi:10.1007/978-3-319-13251-8_3

Gelperin, D., & Hetzel, B. (1988). The Growth of Software Testing. *Communications of the ACM*, *31*(6), 687-695. doi:10.1145/62959.62965

Graham, D., van Veenendaal, E., Evans, I., & Black, R. (2008). Foundations of Software Testing ISTQB Certification. Cengage Learning Emea.

Hrabovská, K., Rossi, B., & Pitner, T. (2019). Software Testing Process Models Benefits & Drawbacks: a Systematic Literature Review. arXiv:1901.01450. arxiv.org/abs/1901.01450

ISO (2009a). ISO 31000, Principles and Guidelines on Risk Management.

ISO (2009b.) ISO/IEC Guide 73:2009, Risk Management -Vocabulary.

Karolak, D. W. (1995). Software Engineering Risk Management. Wiley-IEEE Computer Society Press.

Laksono, M. A., Budiardjo, E. K., & Ferdinansyah, A. (2019). Assessment of Test Maturity Model: A Comparative Study for Process Improvement. In: Proceedings of the 2nd International Conference on Software Engineering and Information Management, ICSIM 2019, (pp. 110-118), January 10-13, Bali, Indonesia.

Liu, H., Kuo, F-C., & Chen, T. Y. (2011a). Comparison of adaptive random testing and random testing under various testing and debugging scenarios. *Software: Practice and Experience*, *42*(8), 1055-1074. doi:10.1002/spe.1113

Liu, H., Xie, X., Yang, J., Lu, Y., & Chen, T. Y. (2011b). Adaptive Random Testing Through Test Profiles. *Software: Practice and Experience*, *41*(10), 1131-1154. doi:10.1002/spe.1067

Patrício, C., Pinto, R., & Marques, G. (2021) A Study on Software Testing Standard Using ISO/IEC/IEEE 29119-2: 2013. In: Al-Emran M., Shaalan K., & Hassanien A. (Eds) Recent Advances in Intelligent Systems and Smart Applications (pp. 43-62). doi:10.1007/978-3-030-47411-9_3

RTI (2002). Planning Report 02-3, The Economic Impacts of Inadequate Infrastructure for Software Testing. Prepared by Research Triangle Institute (RTI) for National Institute of Standards & Technology (NIST). (7007.011) (Accessed:20/02/2017) www.nist.gov/director/planning/upload/report02-3.pdf

Roy, G. G. (2004). A risk management framework for software engineering practice. In: Strooper, P. (Eds.) Proceedings of the Australian Software Engineering Conference, ASWEC 2004, (pp. 60-67), April 13-16, Melbourne, Australia. doi:10.1109/ASWEC.2004.1290458

Serdaroğlu, D. (2015). Yazılım Test Süreci ve TMMi Modelinde Prisma Yaklaşımı Uygulaması. MSc. Thesis (in Turkish), Başkent Üniversitesi, Bilgisayar Mühendisliği Bölümü.

Standards Australia/New Zealand (2004). Risk Management Guidelines Companion to AS/NZS 4360:2004 (HB 436:2004)

Stern, R., & Arias, J. C. (2011). Review of Risk Management Methods. *Business Intelligence Journal*, *4*(1), 59-78.

van Veenendaal, E. (2011). Practical Risk-Based Testing Product Risk Management: The PRISMA Method. EuroSTAR 2011, Novermber 21-24, Manchester, UK. (Accessed:08/03/2017) www.erikvanveenendaal.nl/NL/files/e-book%20PRISMA.pdf

van Veenendaal, E. (2012a). Test Maturity Model integration (TMMi) Release 1.0. TMMi Foundation.

van Veenendaal, E. (2012b). Practical Risk-Based Testing - The PRISMA Approach. UTN Publishers.

van Veenendaal, E. (2014). Test Process Improvement and Agile: Friends or Foes?. *Testing Experience*, *27*, 24-26. (Accessed:07/03/2017) www.erikvanveenendaal.nl/NL/files/testingexperience27_09_14_van_Veenendaal.pdf

van Veenendaal, E., & Cannegieter, J. J. (2013). Test Maturity Model integration (TMMi) Results of the first TMMi benchmark - where are we today?. EuroSTAR. (Accessed:07/03/2017) www.erikvanveenendaal.nl/NL/files/e-book%20TMMi.pdf

Wendland, M-F., Kranz, M. & Schieferdecker, I. (2012). A Systematic Approach to Risk-Based Testing Using Risk-annotated Requirements Models. In: The Seventh International Conference on Software Engineering Advances, ICSEA 2012 (pp. 636-642), Lisbon.