Risk-based Testing Techniques: A Perspective Study

Md. Mottahir Alam
(Research Scholar)
CMJ University, Shillong,
Meghalaya, India

Asif Irshad Khan
(CS Department)
FCIT, King Abdulaziz University,
Jeddah, Saudi Arabia

ABSTRACT

Doing more with less has become a mantra for IT organization in today’s business environment. Nowadays, there are more projects, more competitive pressures and greater failure risk which needs to be managed with fewer resources with tighter timelines. But with all these constraints, there's simply no room for compromise on quality and stability in today’s competitive world especially in case of important business critical applications. So, instead of doing more with less and risking late projects, increased costs or low quality, we need to find ways to achieve better with less. The focus of testing has to be placed on aspects of software that matter most with an aim of reducing the risk of failure as well as ensuring the quality and stability of the business applications. This can be achieved by applying the principle of Risk Based Prioritization of tests, known as Risk-based testing (RBT). The aim of Risk Based testing approach is to ensure that appropriate testing activities are identified and prioritized based on risk. The primary role of risk-based testing is to optimize available resources and time without affecting the quality of the product. RBT approach reduces the risk of failure to the business and increase customer satisfaction. In this light, this paper presents the progress different risk-based testing metrics to measure and control test cases and test activities progress, efforts and costs.

IT organizations must adopt a focused approach and a comprehensive methodology for end-to-end testing. Risk-based testing helps quantify and mitigate risks in the lifecycle of applications, and prioritize tests more effectively. Under RBT, we create Optimized Regression Test Suite based on Business Severity and Priority. The Success of testing team will be the ability to identify high risk defects in software and ensure they are fixed.

Keywords

Risk-based Testing, Value-based Testing, Value-based quality analysis, Software testing and test case prioritization

1. INTRODUCTION

The main objective of software testing is to detect defects in a code or in a module or in a program, in other words Process of giving assurance to the client that, the software under test is performing as intended (meeting the requirements). In general, tests are performed in order to show a lack of quality as discovered by defects, testing always involves comparing between the product and the requirements. Both the development organization and the system users may encounter undesirable consequences if a defect is found in a running /operated system. Software testing is a process that should be done during the development process.

Testing is of different types and risk-based testing (RBT) is considered as one testing which is used to optimize available resources and time without affecting the quality of your product. In RBT each test is intended to probe a specific risk that was previously identified through risk analysis. A simple example is that in Database applications, there is a risk of injection attacks, where an attacker fools the server into displaying results of arbitrary SQL queries. A risk-based test might actually try to carry out an injection attack, or at least provide evidence that such an attack is possible [1], Risk-based testing helps address the rise in business and technological complexity and the growing size of applications by prioritizing test cases based on the defined criticality of a function, encouraging impact assessment of an application functionality failure and increasing testing effectiveness. Despite these benefits, risk-based testing is still used or deployed in a limited manner across organizations.

The paper is organized as follows: section 2 describes the overview of risk-based software testing, its general process, why RBT is needed, and some of its advantages and disadvantages. Section 3 outlines some selected related work. Section 4 presents comparison of some of the state of art of risk-based software testing in terms of differences and issues. Section 5 suggests some criteria for the selection of risk-based software testing approaches and finally, section 6 describes conclusion and future work.

2. OVERVIEW OF RISK-BASED SOFTWARE TESTING

Before going into the details of risk based testing, we should first discuss what Risk is and how it impacts software projects. A risk which may not happened yet and it may never happen in future, but it is a potential problem. In other words, Risks are future uncertain events those may or may not occur. Problems are events that are actually occurred.

Budget, Time and Requirements are the three limiting factors, which are considered for defining success of a project in the Software Development Process. If a software project satisfy all user requirements within estimated time and budget it is considered as a successful project.

Any uncertainty or possibility of loss may result in non conformance of any of these key factors, leading to overtime / over-budget or poor quality project. Software risks, which impact above 3 key factors, can be broadly categorized as [2]:

- Requirement Risks: Requirements is unclear or poorly defined requirements, missing requirement analysis, requirements that are not in-line with customers’ needs, inconsistent ambiguous requirements, inadequate requirements, invalid or impossible requirements, unable to measurable the requirements in terms of specific values etc.

- Technical Risks: Continuous changing requirements, Complexity of architecture or Product is complex to implement, technology change, configuration change, inadequate technical support/ knowledge or No advanced technology available, lack of domain knowledge etc.

- Schedule Risks: Cost overruns, Wrong budget estimation unrealistic time line, Failure to address priority conflicts, No communication in team, inadequate skilled resources, improper resource planning etc.

Risk need to be handled because if it happens then it may cause very negative impact. The occurrence/or non-occurrence of a risk can never be guaranteed beforehand but it may be neutralized through pre-mediated action. For example, let us assume that we are planning to play a cricket match on the weekends. In this case, if it rains on weekends, our cricket match is bound to be spoilt. But we cannot know for sure that on weekend if it is going to rain. Hence, occurrence of rain is a risk that we have to deal with. A test approach that takes into account a risk is called risk-based testing.

Now, when we test a product with a limited number of testers in a small amount of time, we have to curtail the amount of testing that can be performed. This is where the concept of Risk Based Testing comes in. Let us understand the concept in detail.

2.1 Risk-Based Software Testing

Risk Based Testing is a method for prioritizing the tests based on the risk of their failure because tester cannot test everything within available resource, usually it starts early in the project cycle. Idea is to organize testing efforts in such an approach that it reduces the level of product risk at the time of delivery.

In other words in this approach, testing is prioritized in terms of the probability that some feature of the program will fail and the estimated cost of failure. The greater the probability of an expensive failure, the more important it is to test that feature as early as possible and as carefully as possible. [3]

In a project development where there is an enormous business requirement, limited timelines and inadequate resources then there is a must that the testing should cover the most critical functions. So in risk-based testing, we focus on evaluating the critical parts of a requirement on high priority basis for our construction of business and test scenarios to provide the greatest quality at the lowest cost [2].

The objective of Risk-based testing are: most feasible coverage and effective usage of limited resources. Thus, in RBT, we organize the testing processes in order to maximize business value and resources. RBT perform the right level and type of coverage on the right parts and at the right time.

2.2 Why Risk-Based Software Testing needed

As explained in the above section, risk can be termed as an unwanted event that has unfavorable or negative consequences. So, in order to negate these risks or make them less severe, there is a need to identify various scenarios of risks in the software and to build a testing approach based on the concept of software risks. This gave rise to the need of Risk Based Testing.

The main reason for adopting RBT can be the aggregated impact of limited resources – time, budget and human. Since testing generally comes in the last stage of the SDLC, the project's calendar doesn't allow sufficient time for a thorough testing of all functions. Furthermore, the project's budget limits the number of skilled human / software resources. In such a situation, test coverage of all the minute detail of the application would not be possible. In order to balance such situations, more focus should be given on testing those areas that represent the largest risk, if a fault occurred.

2.3 Risk Based testing general Process

The RBT process can be carried out through following important steps [2]:

Step1 - Describe all requirements in terms of risk associated to them also, looks at ways of establishing what the risks are and where they are.

Step2 - Prioritize the requirements, based on risk assessment, looks into the critical, complex and potential error prone areas.

Step3 - According to requirement prioritization define and plan tests also, look for risk Mitigation where tests are built to mitigate the risk.

Step4 - Execute test according to prioritization and acceptance criteria or Monitor / Report regarding the risks.

When deciding on what parts of testing to outsource, you will have to look at it from different angles, mainly at the dimensions of test levels, test types and test activities:

Test Levels: Low-level testing (Unit & Integration test) is generally carried out by the developers themselves. If development is outsourced, these activities are also outsourced with it. System testing on the other hand should be performed by an independent test team and is therefore an excellent candidate for test outsourcing. Finally, Acceptance testing requires business know how (for user acceptance) and a production-like test environment. It is therefore difficult to outsource.

Test Types: Generally speaking, all types of testing, both functional and non-functional, can be outsourced. By its nature, regression test is a good candidate for cost saving, because it involves regular repetition and test automation. Know-how intense test types like load & performance, usability and security are best outsourced to a specialist organization, on a case-by-case basis.

Test Activities: defining what activities within the test process should be outsourced (e.g., test planning, specification, execution and reporting) requires a strategic decision on how much control and knowledge is given away. It can range from test execution only, to performance of the whole process.

2.4 Advantages of Risk Based Testing

The usage of RBT brings several advantages to the testing organization. Some of them are listed below:

- Running the tests in risk order gives the highest likelihood of discovering defects in severity order.
- Preventive activities can be started immediately as problem areas are discovered early in RBT.
- With limited time, money and qualified resources, testing concentrates on the most important matters first, thus delivering the most optimal test, by selecting better strategies and test objects/cases.

- In RBT the focus is not only lies on the risk associated with functionality of the information system but also, on risks to the business.
- Since RBT provides a method to prioritize tests against deadlines, Test cases can be reduced and focused on the most critical areas in other words, because testing is effectively prioritized against what is important to your business - Testing becomes a much more targeted and organized activity.
- During testing, communication (Test Reporting) takes place in a language (risks) that all stakeholders understand.
- It offers a negotiating instrument to client and test manager alike when available means are limited.
- RBT provides clear information on test coverage. Using this approach, we know what has/have not been tested and how much business risks this mitigates.
- RBT provides Flexible Approach as the data, information, and knowledge decision cycle adapts well to change.
- Business Visibility - Risk-based reporting provides a way of communicating with senior stakeholders in a language they understand - effectively raising the value of testing within your organization.
- Allocating test effort based on risk is the most efficient way to minimize the residual quality risk upon release. In other words, RBT associated with picking the right tests out of the infinite cloud of possible tests.
- Measuring test results based on risk will allows the organization to know the residual level of quality risk during test execution, and to make smart release decisions .
- If schedule requires, dropping tests in reverse risk order reduces the test execution period with the least possible increase in quality risk.
- Risks can be continuously monitored to know the status of the project and its quality.

All of these benefits allow the test team to operate more efficiently and in a targeted fashion, especially in time constrained and/or resource-constrained situations.

2.5 Disadvantages of Risk Based Testing

Although risk-based testing has several advantages, it also includes some disadvantages. Following are some of the disadvantages of RBT:

- Unrecognized risks or risks assessed as too low may cause problems if it becomes a reality.
- If the risks are described too abstractly, it may be difficult to attach a test to an identified risk.
- Some mitigation may tasks much cost and more time that they actually add more problems to the project than they're worth in terms of the problems they help detect.
- Risk assessment can be based on too subjective criteria, the reason for that is simply the lack of reliable objective criteria and in that case it is quite common to trust to experts' judgments.
- It is difficult to identify and select the right stakeholders for risk assessment.

3. RELATED WORK

Over the last decade, risk-based testing has received considerable attention in both academics and industry. Several researchers have proposed their approaches for the implementation of risk-based testing in the software projects. Amland [5] stated that IT projects are very rarely on time, schedule or budget, so when it comes down to testing, the time to delivery is extremely short and there is no extra budget left due to the development overrun [5]. To manage such scenarios, test case design techniques should be able to identify the most important test cases to be carried out in view of limited time. Thus, the test cases need to be prioritized to be comparable with each other.

Amland [6] introduces a risk-based testing approach in which resources should be focused on those areas representing the highest risk exposure. He practically applied his approach in a retail banking application. He introduced a methodology that would identify functions in their system where the consequence of a fault would be most costly (either to the vendor or to the vendor's customers) and also a technique to identify those functions with the highest probability of faults.

A risk analysis was performed and the functions with the highest risk exposure, in terms of probability and cost, were identified. A risk based approach to testing was introduced, i.e. during testing resources would be focused in those areas representing the highest risk exposure.

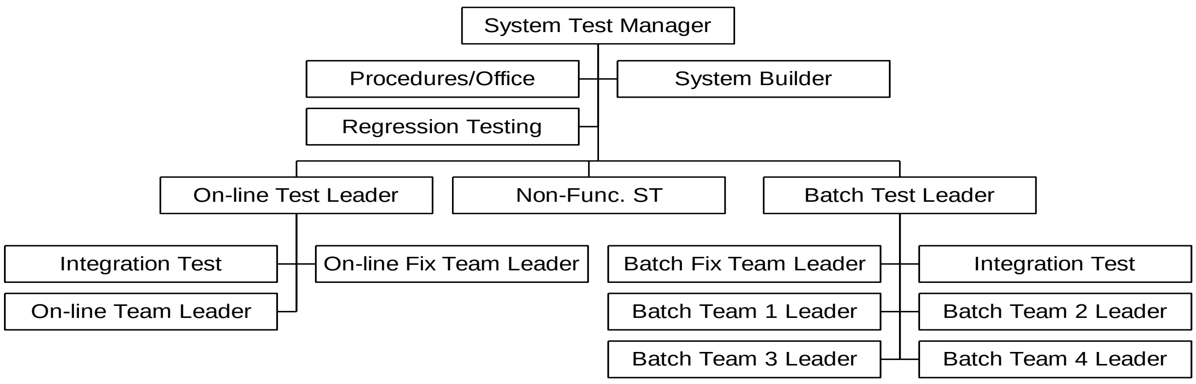

Fig 1: The Risk Based Approach to testing requires a flexible organisation, focused on fixing bugs related to critical functions [6].

Amland[6] stressed that risk-based testing must be supported by an organization (as shown in Fig 1) where in roles and responsibilities should be defined. His approach gave certain metrics to help software developers determine how best to use their testing time. His suggested metrics include ones that help to identify high-risk areas, a minimum level of testing, and additional testing along with some that monitor project quality and progress to calculate estimated effort to complete, as well as managing the test process.

The method proposed for conveying the results of these metrics is a matrix with probability values on the rows and consequence values on the columns. Table1 shows an example of calculated Risk Exposure for the function Close Account in the case study.

He compared the estimated resource requirements using his risk-based approach against the traditional approach and concluded that the risk based approach consumed less resources relative to the original estimate based on a traditional test approach as shown in Figure 2. Bach [7] describes a heuristic analysis to do risk-based testing.

Heuristic refers to experience-based techniques for problem solving, learning, and discovery. Where the exhaustive search is impractical, heuristic methods are used to speed up the process of finding a satisfactory solution; mental short cuts to ease the cognitive load of making a decision [8].

Bach proposed two approaches to his heuristic risk-based testing: inside-out and outside-in. With an in-side out approach, the test team begins with the situation details and then performs risk identification by looking for vulnerabilities, threats and victim's .With an outside-in approach, the test team begins with a predefined risk list and reacts to those risks that are visible in the present situation. Bach suggests three types of lists: quality criteria categories, a generic risk list, and risk catalogs.

Quality Criteria Categories are used to help elicit new requirements or clarification on existing requirements such as: capability, reliability, usability, and performance, install ability, compatibility, supportability, testability, maintainability, portability, and localizability.

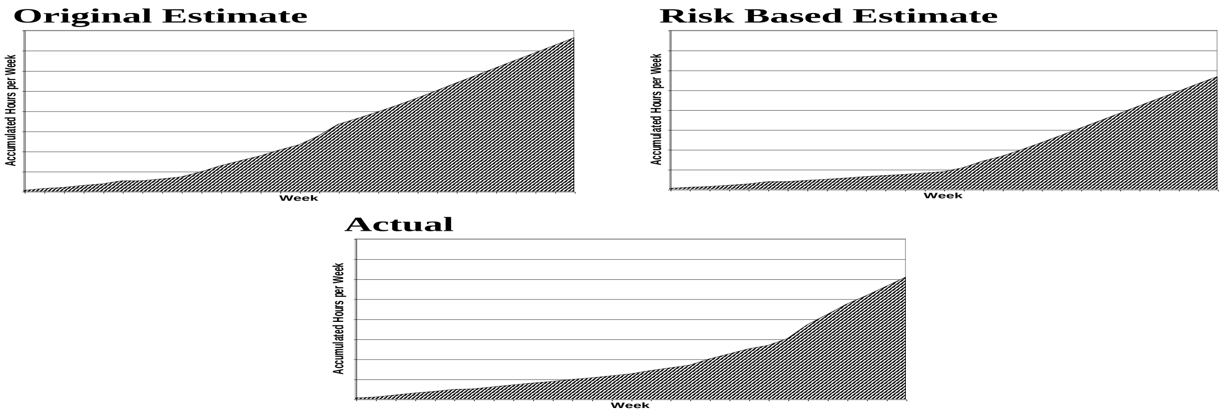

Fig2: Resource profiles for Original Estimate (i.e. Traditional Approach), Risk Based Estimate (Risk Based Approach) and Actual (i.e. actual accumulated number of hours spent).

Table 1. Example of calculated Risk Exposure for the function Close Account [6].

	Cost			**Probability**						
Func.	**C(v)**	**C(c)**	**Avrg. Cost**	**New Func.**	**Design Quality**	**Size**	**Compl.**	**Weight**	**Probability**	**Risk Exp. funct.**
				5	5	1	3	Avrg.	P(f)	Re(f)
Close Accnt.	1	3	2	2	2	2	3	7,75	0,74	1,48

Generic Risk List is a list of risks that are common to all software systems, and risk catalogs are domain specific. Table 2. Shows the generic list of Bach [7].

Risk catalogues is a list of risks that belong to a particular domain. Risk catalogs are motivated by testing the same technology pattern over and over again. A risk catalog can be created by categorizing the kinds of problems one has observed during testing in a particular domain.

Below follows an example of risk catalog from installation. Problems that may occur in this domain are listed. Table 3 displays an example of an installation risk catalog.

Bach [7] suggested that the above three kinds of lists can be used in any of the following ways:

1. Decide what component or function to be analyzed.

2. Determine the scale of concern. In a general term, everything is assumed to have a normal risk unless there is some reason to believe it's a higher or a lower risk.

3. Gather information about the things you want to analyze.

4. Determine the importance of each risk in the present situation.

5. Record other potential risks that are not on the list.

6. Record any unknowns which may impact the ability to analyze the risk.

7. Double-check the risk distribution

Table 2. A Generic Risk List [7]

Complex: anything disproportionately large, intricate, or convoluted.

- New: anything that has no history in the product.

- Changed: anything that has been tampered with or "improved".

- Upstream Dependency: anything whose failure will cause cascading failure in the rest of the system.

- Downstream Dependency: anything that is especially sensitive to failures in the rest of the system.

- Critical: anything whose failure could cause substantial damage.

- Precise: anything that must meet its requirements exactly.

- Popular: anything that will be used a lot.

- Strategic: anything that has special importance to your business, such as a feature that sets you apart from the competition.

- Third-party: anything used in the product, but developed outside the project.

- Distributed: anything spread out in time or space, yet who elements must work together.

- Buggy: anything known to have a lot of problems.

- Recent failure: anything with a recent history of failure.

Table 3. A Risk Catalogue for installation [7]

1. Wrong files installed
   o temporary files not cleaned up
   o old files not cleaned up after upgrade
   o unneeded file installed
   o needed file not installed
   o correct file installed in the wrong place
2. Files clobbered
   o older file replaces newer file
   o user data file clobbered during upgrade
3. Other apps clobbered
   o file shared with another product is modified
   o file belonging to another product is deleted
4. HW not properly configured
   o HW clobbered for other apps
   o HW not set for installed apps
5. Screen saver disrupts install
6. No detection of incompatible apps
   o apps currently executing
   o apps currently installed
7. Installer silently replaces or modifies critical files or parameters
8. Install process is too slow
9. Install process requires constant user monitoring.
10. Install process is confusing
   o UI is unorthodox
   o UI is easily misused
   o Messages and instructions are confusing

He suggested three ways to organize risk-based testing namely, risk watch list, risk/task matrix and component risk matrix. The risk watch list is a list of risks that you periodically review during the project to be aware of the most common risks. Risk/task matrix sorts the risks according to their importance. It provides us a list of the risk mitigation tasks to be invested in to minimize the risk associated with each risk. It is very useful in negotiations for more testing resources.

The component risk matrix consists of a table with three columns. It breaks the product into 30 or 40 areas or components. In the left column the components are listed. In the middle the scale of concern, “low”, “normal” or “high” risk is listed. In the right column the risk heuristics for that component are listed. The risk heuristic indicates the risk for that component.

During testing, the components are tested according to their risks as specified in the matrix. Table 4 shows an example of component risk matrix.

In 2002, Chen[9] gives an approach for risk-based regression testing optimization [13]. He suggested a specification-based method for regression test selection. The author applies a risk value to each test case to prioritize them. The formula for calculating the Risk Exposure RE(f) is taken from Amland [1] which is as follows:

$$RE(f) = P(f) \times C(f)$$

where C(f) is the cost of fault for each test case and P(f) is the severity probability for each test case.

Cost is categorized on a scale from one to five. Two types of cost can be considered: The cost for the customer (losing market share) and the cost for the vendor (high maintenance cost).Also, P(f) is found by looking on number of earlier defects and severity of these defects.

Table 4. A component risk matrix [7]

Component	Risk	Risk Heuristics
Printing	Normal	Distributed, popular
Report Generation	Higher	New, strategic, third-party, complex, critical
Installation	Lower	Popular, usability, changed
Clipart Library	Lower	Complex

Scenarios covering one or more test case are created. A traceability matrix is created, mapping the test cases with each scenario. The Risk Exposure for each scenario is calculated .Based on these risk values, the test cases are comparable and can be prioritized to either be included in, or excluded from, a re-running regression testing process.

Gerrard [10] employs a risk-based approach to test e-business using Failure Modes and Effect Analysis (FMEA), but his method can apply to other kinds of system also. FMEA is mostly a qualitative analysis with an aim to identify the parts of the system that will need improvements to meet the safety and reliability requirements. The process consists of five stages: risk identification, risk analysis, test scoping and test process definition.

A table called Test Process Worksheet is the main working document in the method and is completed in stage 1 to 4. Each row in the Test Process Worksheet consists of a failure mode, also called risk. The columns consist of scoring and prioritization, assignment of test objectives, effort, costs and so on for this failure mode. In Risk identification stage, an inventory of potential failure modes, similar to the checklists of Bach [7], is prepared. These are derived from checklists.

In Risk analysis stage, a risk workshop is convened with representatives from the business, development, technical support and testers. Each risk is considered, and the probability and the consequence are assessed. The risk exposure is calculated. Table 2.5 shows an example of this stage. In Risk response stage, if the risk is testable, it is turned into a test objective using the risk description. In the Test scoping stage, a budget for testing is passed .In the Test process definition stage, the stage-by-stage test process is documented.

Scheafer [11] approach of risk-based testing focus on prioritizing what to test, by finding the most important and worst parts of the product. The most important parts of the system are found using factors like cost of failure, most visible and most used parts of the product. The worst parts of the system are found using defect generators like complexity, changed areas, new technology, new solutions, new methods, new tools, number of people involved, where there was time pressure and local factors.

Table 5. An example calculating the risk, taken from Schaefer [11].

Area to test	Business Criticality	Visibility	Complexity	Change Frequency	RISK
Weight	3	10	3	3	
Order registration	2	4	5	1	46*18
Invoicing	4	5	4	2	62*18
Order statistics	2	1	3	3	16*18
Management Reporting	2	1	2	4	16*18
Performance of order registration	5	4	0	1	55*3
Performance of statistics	1	1	0	0	13*0
Performance of Invoicing	4	1	0	1	22*3

Weights are assigned for each relevant cost factor and defect generator. For each part of the system, values are assigned for the factors and the defect generators. Higher values mean that the area is more important or worse. This is illustrated in Table 5. These values are multiplied by the weights and added together. The highest values give the most risky parts and should be prioritized.

Another formal and systematic approach of doing risk analysis is HazOp [12].It is carried out in the later stage of the development, when the architecture design is already built. The study nodes are the points of the system where our analysis is focused. These can be points where the system interacts with its environment or where parts of the system exchange information. HazOp contains two structuring devices, the table and the guide words.

Stålhane [13] describes how to use HazOp to find the subsystems with most hazards.UML use cases are used as a starting point. Since use cases don't study nodes; therefore a standard HazOp with guide words could not be used. Instead, a functional HazOp is performed based on the functionality offered by the system. This can be done in the following step:

Step 1: Prepare use cases for the subsystem to be analyzed.

Step 2: A warm up exercise looking at previous risk analysis.

Step 3: Perform function-based HazOp by addressing questions like:

-How can this function fail?

-What will be the consequences for the stakeholders, the service receiver, the service provider and the development company? This will give a list of hazards.

Step 4: Document the result obtained and get the feedback from the participants.

Step 5: Assess severity of each hazard.

A score from 0 to 3 is given to indicate the severity of each hazard, where 3 is the most serious. Different stakeholders may assign different severities to the same hazard. For each subsystem and stakeholder group, the number of functions that receive each hazard value is registered. The score for each function can be found in two ways, the weighted average and the score to the majority of the functions.

Redmill [14] has presented a high-level approach for risk-based testing. He discusses what constitutes "risk" and how it can be used for test focusing. In his paper, risk is defined as a fairly general term, covering aspects such as safety risks, financial risks to the customer if the system fails or economical risks to the manufacturer if the software does not have the desired quality. This approach is applicable for all kinds of product risks, but does not provide direct support for the construction of test cases.

Wallace and Keil [15] in their research, analysed the effect of risks from both process and project viewpoints. The results from this research are noteworthy as they are based on a study of 500 software development projects by members of the Project Management Institute. At the end of each project, the project managers participated in an online survey to indicate how many and to what level each of a set of 53 risks existed within the project. The 53 risks were categorized into the four categories of customer mandate, scope and requirements, environment, and execution. They used their survey results to indicate the perceived relative importance of risks.

Raparla and Sherrell [16] presented a risk assessment tool called QUART-ER (QUick Assessment of Risks Tool for Engineering Requirements). QUART-ER allows users to analyse,

Plan and monitor project risks, especially those encountered during requirements. In the design of QUART-ER, they first identified primary risks with the aid of an initial set of software developers from local industry. Next, they developed and distributed an on-line survey to software engineers and managers at software firms.

Based on survey responses, risk categories and associated risk factors within categories were collated into a risk assessment form, which was implemented in QUART-ER. This tool allows team leaders and software developers to assign rankings to risk categories and/or to rank the more detailed, informative risk factors. After that QUART-ER compares these rankings to those of previous projects providing a flag if the risks are considered “too high” for project completion.

Van Veenendaal [17] introduced PRISMA (PRoduct RIsk MAanagement) method for finding product risks that can be applied to all the level of testing, .i.e., from component testing to acceptance testing. In this method, the components which are classified as highest risk are given a higher priority (tested first and more rigorously) than those with a low risk.

He compared the risk-based testing to the concept of “good enough testing” where instead of aiming for the unrealistic goal of zero defects, testers intend for a product that has no critical problems and has an acceptable number of benefits, so that the benefits sufficiently outweigh the non-critical problems, and a release date that cannot be pushed back for further improvements because the delay may cause greater damage in a business sense. The PRISMA model is an implementation of a product risk matrix where the impact and likelihood of defects are calculated and assigned to the matrix. The different steps in the PRISMA process are planning, kick-off, individual preparation, gathering individual scores, consensus meeting, and defining the test approach.

During the planning step, requirements or architectural documents are collected, the risks are identified from these input documents, and are ranked or weighted, stakeholders that will participate in the risk analysis are determined, and scoring rules are established. In the kick-off step, which is optional, a meeting is held with the test manager and all the stakeholders to make sure that all players understand their roles. In the individual preparation step, the stakeholders assign a score to each risk individually and documents their perceptions and assumptions. Then, in the gathering individual scores step, the team manager checks all scores for correctness, processes the scores by tabulating the average value of the impact and likelihood respectively, and places the results in a risk matrix to be discussed in the consensus meeting. Based on the final positioning of the risks on the matrix or matrices, the risks are prioritized and a test approach is determined based on the prioritization.

Stallbaum et. al.[18] made a first step towards automated generation of risk-based test suites based on previously calculated requirements metrics. They presented a prototype research tool called RiteDAP has which can generate test cases out of weighted activity diagrams in a two-stage process. In the first step, paths through the activity are derived in a non-risk based way.

Then in the second step, the paths are ranked due to the risk they include. The traversal algorithm of the test case generator is predefined and is non-adjustable. The risk-based selection of test cases in that approach is a simple ordering of paths due to their subsumed risk exposures. Zimmermann et. al. [19] has presented a methodology called sequence-based specification to express formal requirements models as low-level mealy machines for safety-critical systems. They first build a system model based on the requirements specification and then the outcome of a hazard analysis is weaved into the mealy machine.

The correctness of the natural language requirements is assumed to hold as there is no thorough approach to verify or validate the natural language requirements prior to performing the hazard analysis. Finally, they describe an algorithm that derives test models that include critical transitions out of the system model for each single identified hazard in order to verify the implementation of a corresponding safety function.

Q. Li et al.[20] in their paper demonstrated a value-based approach for prioritizing features for testing which aligns the internal test process with the value objectives coming from the customers and the market. This involves prioritizing features based on their business importance, quality risk, and testing cost of each feature; adjusting feature’s value priority during the testing process; and providing stop-testing decision criteria based on the market pressure.

They also carried out a case study in a real-life business project and showed that their helps the test manager to identify features with high business importance, high quality risk and low cost, to focus testing effort on these features and to control and adjust testing plan toward success-critical stakeholders (SCSs) win-win realization[21,22]. Their result shows that this method can help to improve ROI of testing investment at the early stage, especially when the market pressure is high.

Kloos et.al. [23] has described an approach for transitioning from a fault tree as produced by a fault tree analysis (FTA). It is used in combination with a system model, expressed as mealy machine, to generate a test model. A test model is in their definition a system model with failure modes and critical transitions leading to the failure modes. This approach is useful for risk-based testing of safety functions for safety-critical systems.

Recently, Zech [24] gave an approach to risk-based security testing using models for cloud environments which is still in a very early state. Due to a cloud’s openness, in theory there exist an infinite number of tests. Taking this into account, they proposed a new model–driven methodology for the automatic risk analysis and subsequent deduction of misuse cases, defined by negative requirements derived from risk analysis for the security testing of cloud environments. The risk analysis is also planned to be carried out completely automatically by using a vulnerability repository. Neither one of the involved models has been described in greater detail, nor have the involved transformations been specified so far.

4. COMPARISON SOME STATE OF ART OF RISK-BASED SOFTWARE TESTING IN TERMS OF DIFFERENCES AND ISSUES.

All the approaches discussed here use risk to prioritize what to test. Many use similar methods to the one of Amland [6] where factors that can increase the cost and consequences of failure for some parts of the product are found. Amland [6] look at the cost in respect of maintenance, legal issues, and reputation for the vendor and the customer, while Schaefer [11] look at what functions have the highest importance and cause bigger inconveniences of failure for the user. His important areas are more or less the same as Amland [6] calls cost.

Schaefer [11] consider many factors that will increase the likelihood for an error done by the developers – called defect generators. From these defect generators he can find the parts with most defects. Chen [9] uses many of the same ideas as Amland [6], but looks at test cases, not functions. Gerrard [10] use a different way to prioritize what to test. Instead of different areas of the product, a risk analysis on different the failure modes is performed. Tests are generated from the failure mode with the priority on the failure modes with highest risk. This is similar to Stålhane et.al. [13] where they used HazOp to find hazards. The number and severity of the hazards for each function is used to prioritize the functions. This is another way to sort out problematic areas in a risk analysis. The risk analysis was done to decide what functions to put extra effort into, but could also be used to decide what functions to test.

Stålhane [13] analyze how use cases can fail. This is similar to Bach's inside-out where he looks at how functions can fail. The consequences for each failure are considered. Stålhane [13] assess severity for each failure mode. These two approaches does not try to give guidelines on what to test, they rather look at the risk in order to find possible faults. These methods can be helpful when use cases are gathered, but it is difficult to make a software tool that can help the tester.

Analyzing the approach of Kloos et.al. [23], we conclude that although they claimed their approach to be risk-based, they do not provide a clear explanation on how to use the identified risks for the generation of test cases.

Going through the approach presented by Redmill [14]**,** we conclude that although it is applicable for all kinds of product risks, but does not provide direct support for the construction of test cases.

The drawback of the RiteDAP tool given by Stallbaum et. al.[18] is that the algorithm used for the test case generation is fixed and can't be modified.

The methodology described by Zimmermann et. al. [19] failed to address the question of ranking the critical transitions in the test models with respect to their risk priority. Further, it is not clear, whether and how the algorithm they present can be modified in order to vary the test case generation process.

Although the author [24] claims to offer a quite sophisticated approach to risk-based security testing of cloud environments, neither one of the involved models has been described in greater detail, nor have the involved transformations been specified.

5. CRITERIA FOR THE SELECTION RISK-BASED SOFTWARE TESTING APPROACHES

After analysing the different approaches of risk-based testing, it has become clear that the most suitable approach depends entirely upon the specific criteria of any given project. The approach that may be most suitable to one project may well be ineffective for another project.

Provided that the best method is selected and implemented by all parties concerned there should be every chance that the element of risk in a project can substantially be reduced.

Some of the factors which should be kept in mind while selecting a suitable approach for any given project are system complexity of the project, timeframes /deadlines, available resources, and acceptable risk levels.

6. CONCLUSION

Risk based testing is a powerful testing technique that helps the testing teams to streamline their testing efforts, which in turn helps in mitigating the risk and minimizing the testing efforts, thus, bringing an objectivity to test designing and test management activities.

The goal of risk-based testing cannot practically guarantee a risk-free project. What we can expect from risk-based testing is to carry out the testing with best practices in risk management to accomplish a project outcome that balances risks with quality, features, budget and schedule. Based on our analysis of different approaches of RBT, we will propose a novel risk-based testing model in future. Furthermore, we will apply that model to do some case studies in order to get empirical results for our methodology.

7. REFERENCES

[1] C. C. Michael, Will Radosevich, Cigital, 2007, [Online], available, https://buildsecurityin.us cert.gov/bsi/articles/best-practices/testing/255- BSI.html

[2] Unnati Bajpai , 2012 Risk Based Testing 102 , [Online], available, http://help.utest.com/testers/crash-courses/general/risk-based-testing-102

[3] Pramod Lumb, 2012 Risk Based Testing 101, [online], http://help.utest.com/testers/crash-courses/general/risk-based-testing-101

[4] Mika Lehto, 2011, The concept of risk-based testing and its advantages and disadvantages, [online], https://www.ictstandard.org/article/2011-10-25/concept-risk-based-testing-and-its-advantages-and-disadvantages.

[5] S. Åmland, 1999 Risk Based Testing and Metric, 5th International Conference EuroSTAR, Barcelona, Spain,

[6] S. Åmland, "Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study", Journal of Systems and Software 53(3), 2000, pp. 287-295.

[7] J. Bach, 1999 Heuristic Risk-Based Testing. Software Testing and Quality Engineering Magazine, November 1999, pp. 96-98.

[8] Wikipedia, 2012, the free encyclopedia, [Online], http://en.wikipedia.org/wiki/Heuristic.

[9] Y. Chen, R. Probert, and P. Sims, 2002 Specification-based Regression Test Selection with Risk Analysis, In: Proceedings of the conference of the Centre for Advanced Studies on Collaborative research (CASCON '02), pp. 1.

[10] Paul G, 2002, Risk-Based E-Business Testing, ISBN 1580533140, page 3 – 29 and 51 – 80

[11] Hans Schaefer, Strategies for Prioritizing Tests against Deadlines Risk Based Testing, Undated, [Online], http://home.c2i.net/schaefer/testing/risktest.doc

[12] Marvin Rausand, Risiko Analyse veiledning til NS 5814, 1991, ISBN 82-519-0970-8, page 41 – 100

[13] Tor Stålhane, Gunhild Sivertsen Sørvig,2003 RiskAnalysis as a Prioritizing Mechanism in SPI, EuroSPI

[14] F. Redmill, 2004 Exploring risk-based testing and its implications: Research articles," Softw. Test. Verif. Reliab., vol. 14, no. 1,pp. 3–15, 2004.

[15] Wallace, L. and M. Keil 2004 Software Project Risksand their Effect on Outcomes. Communications of the ACM,47 (4), 68-73.

[16] Raparla, R. and L. Sherrell 2007 A Tool for Risk-based Testing"

[17] Veenendaal, E. 2006 Practical Risk-Based testing PRoduct RIsk MAanagement: the PRISMA method.Improve Quality Services BV, [online], www.improveqs.nl

[18] H. Stallbaum, A. Metzger, and K. Pohl 2008 An Automated Technique for Risk-based Test Case Generation and Prioritization", In: Proceedings of the 3rd Workshop on Automation of Software Test,AST'08, at 30th Intl. Conference on Software Engineering (ICSE), Germany, pp. 67-70.

[19] F. Zimmermann, R. Eschbach, J. Kloos, and T. Bauer 2009 Risk-based Statistical Testing: A Refinement-based Approach to the Reliability Analysis of Safety-Critical Systems", In: Proceedings of the 12th European Workshop on Dependable Computing (EWDC), France.

[20] Qi Li1, Mingshu Li2, Ye Yang2, Qing Wang2, Thomas Tan1, Barry Boehm1,and Chenyong Hu2 2009 Bridge the Gap between Software Test Process and Business Value- A Case Study" Springer-Verlag Berlin Heidelberg , pp. 212–223.

[21] Boehm, B., et al.: 1998 Using the WinWin spiral model: a case study. IEEE Computer 31(7), 33–44 (1998)

[22] Boehm, B. 1988 A Spiral Model of SoftwareDevelopment and Enhancement. IEEE Computer 21(5), 61–72.

[23] J. Kloos, T. Hussain, and R. Eschbach 2011 Risk-Based Testing of Safety-Critical Embedded Systems Driven by Fault Tree Analysis In: Proceedings of the IEEE Fourth International Conference onSoftware Testing, Verification and Validation (ICST) IEEE Computer Society, Berlin, 2011, pp. 26-33.

[24] P. Zech, 2011 Risk-Based Security Testing in Cloud Computing Environments IEEE Fourth International Conference on Software Testing, Verification andValidation (ICST), 2011, pp. 411-414

AUTHOR'S PROFILE

Md Mottahir Alam has around six years of experience working as Software Engineer (Quality) for some leading software multinationals where he worked on projects for companies like Pearson and Reader's Digest. He is ISTQB certified software tester. He has received his Bachelors degree in Electronics & Communication and Masters in Nanotechnology from Faculty of Engineering and Technology, Jamia Millia Islamia University, New Delhi.

Currently, he is a research scholar at CMJ University, Shillong, Meghalaya, India.

Asif Irshad Khan received his Bachelor and Master degree in Computer Science from the Aligarh Muslim University (A.M.U), Aligarh, India in 1998 and 2001 respectively. He is presently working as a Lecturer Computer Science at the Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia.

He has more than eight years experience of teaching as lecturer to graduate and undergraduate students in different universities and worked for four years in industry before joining academia full time.

He has published more than 14 research papers in reputed International journals, His current research interests include software engineering with a focus on Component Based and Agent Oriented Software Engineering.