Abstract

This paper describes research in heuristic risk-based testing. The aim of this research is to design and implement a risk assessment tool called QUART-ET (Quick Assessment of Risks for Engineering Testing) to facilitate the risk management process. The purpose of QUART-ET is to help software development teams manage risks specifically related to testing and to aid management in discovering an acceptable risk appetite for software maintenance. The tool provides automated assistance to members of the testing team during the process steps of risk identification and risk analysis. The researchers use an outside-in approach to heuristic-based testing by furnishing users with predefined product and process risk lists within the tool.

1. Introduction

Over the last decade, more companies have become interested in performing risk management during the software development life cycle [11, 6]. One reason for the increased interest may be due to the importance placed on process improvement by the Carnegie Mellon Software Engineering Institute (CMU/SEI) [8], as risk management is one of 25 process areas in the Capability Maturity Model Integrated (CMMI). More specifically, risk management must be conducted in order for an organization to be rated as Level 3 in the Staged Model of the CMMI, where Level 5 is the best rating.

In order to manage risks, software development teams need to be familiar with risk assessment methods. The majority of such methods focus on the risks associated with late software delivery, over-budget projects, and software systems that fail to meet their requirements [13]. In other words, there is an emphasis on the final delivery of the product and the majority of the assessment occurs during the requirements phase as opposed to risk assessment throughout the software development life cycle. Some analysts feel that such a narrow view of risk hampers our ability to reduce software failures [10].

In particular, testing a system thoroughly requires a lot of resources (time, personnel, etc., the very forces with which the project managers may be in conflict). In order to deliver the project on time and within budget, project managers may have to limit the number of test cases. That leaves us with the following questions:

- Which parts of the system need to be tested first?
- How much testing is enough?

According to [4], “about 80 percent of the defects come from 20 percent of the modules, and about half the modules are defect free.” In order for a project to be successful, the project manager must decide which parts of the system need to be thoroughly tested, and this requires careful analysis of the risks involved.

This paper describes research in heuristic risk-based testing. In particular, the ideas for a tool QUART-ET (QUick Assessment of Risks Tool for Engineering Testing) are presented. The aim of QUART-ET is to help software development teams manage risks specifically related to testing. The remainder of the paper is organized as follows. Section 2 provides the reader with definitions and background information, while Section 3 summarizes related work. Section 4 describes plans for the design of QUART-ET, and Section 5 presents the generic risk list that will be available within the tool. Section 6 contains the conclusions.

2. Risk Terms

Before a software developer can make plans to
mitigate or remove a risk, he or she must know what constitutes a risk. In [15], the author defines risk as “any threat to the achievement of one or more of the cardinal aims of the project”. Cardinal aims fall under one or more of the following categories: whole life costs, system goals and side effects. Another definition for risk is the following circular description. “A risk is a problem that has yet to occur, and a problem is a risk that has already materialized.” [9]. This definition leads to a one-line summarization of risk management, namely the “process of thinking out corrective actions before a problem occurs.” A more complete, delineated explanation for a risk management process[15, 9] follows:

- Risk identification. Uncovering of risks through initial brainstorming along with a process for continued discovery.
- Risk analysis. Quantification of risks in terms of possible repercussions.
- Risk planning. Contingency and mitigation plans to handle situations if the risk materializes.
- Risk monitoring. Continuous assessment of the risks and revision of risk mitigation plans.

These are the basic steps needed to perform risk management. The same steps are organized slightly differently or given different names by some authors. According to [3], the practice of risk management involves two primary steps: risk assessment and risk control. Risk assessment involves risk identification, analysis and prioritization, while risk control covers risk planning, monitoring and resolution.

With risk identification a list of risks (a risk register) is made from project workshops, corporate checklists, prompt-lists, and taxonomies [15]. Other risk identification techniques that can be used are decision-driver analysis, assumption analysis and decomposition, risk catalogs, generic risk lists and quality criteria categories [3, 2]. Input from all stakeholders is vital to identifying most of the risks.

The risk register that is compiled may end-up being very long; and dealing with each and every risk might not be feasible. Hence, the risks need to be analyzed, so the most important and relevant risks for this project can be identified. In order to analyze the risks, cause-effect analysis [15], or risk-exposure [3] can be used. Every risk will have a cause and once materialized, it will have some effect on the project. Cause has a probability (likelihood of risk) and effect has a size (impact of risk), both of which can be estimated. Probability and impact can each be assigned either numeric values (1-10) or a qualitative scale (e.g., high, medium, low, unlikely) can also be used. Based on these values, the risk exposure [3] is calculated as follows:

\[
\text{Risk Exposure (RE)} = \text{Probability} \times \text{Impact}
\]

Risks can be prioritized based on either the overall RE value, or the individual probability and impact values. Risks with high probability and impact have the top priority.

Once the risks are prioritized, steps to control the risk must be planned. A plan for each of the high priority risks is compiled. According to [3], this plan should be “oriented around answering the standard questions of why, what, when, who, where, and how”, so that the plan can be easily implemented and monitored.

Risk resolution involves executing the plan. Based on the plan and how it is implemented, the probability and impact of a risk might increase or decrease. So, the risks should be constantly monitored and the plan should be changed as needed.

Related to the risk management process steps is the concept of risk appetite, namely the “amount of risk that management is willing to take so that the software placed into operations will be risk-free” [17]. The aim of this research is for the tool QUART-ET to aid management in discovering an acceptable risk appetite for software maintenance. Here maintenance is the activity that ensues when “software undergoes modifications to code and associated documentation due to a problem or the need for improvement or adaptation” [12]. Note that maintenance includes changes made to the product during the software developmental life cycle as well as post-delivery.

As previously mentioned, this research involves heuristic risk-based testing as a means to manage risks. In heuristic risk-based testing an inside-out or an outside-in approach may be used [2]. With an in-side out approach, the test team begins with the situation details and then performs risk identification. With an outside-in approach, the test team refers to a predefined risk list and reacts to those risks that are apparent in the current circumstance. Bach recommends three types of lists: quality criteria categories, a generic risk list, and risk catalogs. Quality categories are used to help elicit new requirements or clarification on existing requirements. Generic risks are common to all software systems, and risk catalogs are domain specific.

3. Related Research

PRISMA [19] is a method for finding product risks that can be applied to every level of testing from
component to acceptance. Its risk analysis plays a vital role in determining test design techniques and test approaches so that components classified as highest risk are given a higher priority (tested first and more rigorously) than those with a low risk. Van Veenendaal compares risk-based testing to the concept of “good enough testing” where instead of aiming for the unrealistic goal of zero defects, testers strive for a product that has no critical problems, has an acceptable number of benefits, where the benefits sufficiently outweigh the non-critical problems, and a release date that cannot be pushed back for further improvements because the delay would cause greater damage.

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 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, the risks are ranked or weighted, stakeholders that will participate in the risk analysis are determined, and scoring rules are established. The kick-off step, which is optional, is a meeting with the test manager and all the stakeholders to make sure that all players understand their roles in risk-based testing. During individual preparation, a stakeholder assigns a score to each risk and documents this or her perceptions and assumptions. Then 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.

Amland [1] puts a methodology from Ovstedal and Stalhane [16] into practical use by applying risk-based testing and metrics to system testing in retail banking applications. Amland emphasizes that risk-based testing must be supported by an organization in combination with roles and responsibilities. He also recommends that certain metrics be applied 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.

Wallace and Keil [20] analyze the effect of risks from both process and project viewpoints. The results from this research are significant 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. Survey results indicate the perceived relative importance of risks.

With respect to process, the researchers found that risks associated with scope/requirements and execution have the most effect on the outcome of the process. Notably execution risks were found to be twice as important. Customer mandate and environment had the least effect. In terms of project risks, they found that if execution risk is minimized the effect from other project risks can also be minimized. In addition, higher levels of execution risk can be compensated by low levels of scope/requirements risk. Similarly, if scope/requirements risks are high, execution risks must be minimized to have a successful project.

Raparla and Sherrell [18] have designed a risk assessment tool called QUART-ER (QUick Assessment of Risks Tool for Engineering Requirements). QUART-ER allows users to analyze, plan, and monitor project risks, especially those encountered during requirements. In the design of QUART-ER, Raparla and Sherrell first identified primary risks with the aid of an initial set of software developers from local industry. Next, an on-line survey was developed and distributed to software engineers and managers at software firms in the Mid-South area. 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. The tool allows team leaders and software developers to assign rankings to risk categories and/or to rank the more detailed, informative risk factors. Then QUART-ER compares these rankings to those of previous projects providing a flag if the risks are considered “too high” for project completion.

4. Project QUART-ET

4.1 Purpose and Scope

The purpose of the risk assessment tool QUART-
ET is to aid testing teams during test design to ascertain the likelihood of project and process risks. In other words, the tool provides automated assistance to members of the testing for risk identification and risk analysis, the first two steps of the risk management process. The researchers use an outside-in approach to heuristic-based testing by furnishing users with a predefined risk list within the tool. This list includes a list of generic risks for both the product and the process.

The research path of QUART-ET has followed a similar path as that of QUART-ER in that the initial risk list was designed based on a literature review, and refinements were made to this list based on comments from practitioners. In particular, testing team members at FedEx analyzed initial product and process risk lists constructed by the researchers for the tool QUART-ET, and provided suggestions for additional risks. Many of the risks that the FedEx employees named were specific to their environment, so the researchers generalized these risks to allow the tool to be used in multiple environments. Plans are underway to distribute an on-line questionnaire with the current risk list to developers and testers within FedEx so that even more employees can provide valuable feedback to the researchers.

4.2 Risk Management with QUART-ET

During the risk identification step of the risk management process, the testing team will identify the major risks that can occur during testing using a predefined risk list. As in QUART-ER [18], the risk list for QUART-ET has two levels: categories and factors. Users may choose to rank risks at the high-level category mode or to obtain a more fine-tuned ranking through the detailed factor level. For an example category and its respective factors, see Table 1.

<table>
<thead>
<tr>
<th>Table 1: Example category and factors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Failure History:</strong></td>
</tr>
<tr>
<td>Risks involved due to lack of or existence of failure history</td>
</tr>
<tr>
<td>A. Feature/module/system has recent failure history (has failed tests before).</td>
</tr>
<tr>
<td>B. Adding a new feature/module (feature/module has no failure history).</td>
</tr>
</tbody>
</table>

QUART-ET acts as a black-box risk assessment tool in that users enter inputs in the form of category or factor rankings based on their perceptions of risk likelihoods, and the tool provides an overall quantified risk perception as output. This risk perception score is based on the composite score for the current project and a stored score that is derived from historical rankings.

The risk perception has associated qualified anchor points namely: No Risk, Minimal Risk, Moderate Risk, Serious Risk, Very Serious Risk and Showstopper. Each member of the testing team will rate each category or factor using the perception ratings.

The next step in the risk management process model is risk analysis. During this step, prioritization of selected risks occurs. The prioritization of risk categories and factors is accomplished through weights. The weights can be automatically calculated by QUART-ET or manually entered by the user. Weights range from 0-10 with score of 0 meaning no perceived threat or influence of the category/factor on the successful completion of the project while 10 means highest perceived threat or influence on the project. The overall risk perception will be a raw score based on weights plus a qualified ranking using the anchor points named above.

To assure good weight estimates, a Kalman Filter [21] is employed. Using this statistical approach offers two major advantages. The feedback loop of the Kalman Filter allows for the evolution of finely tuned weights for the categories and factors so that more accurate weights eventually emerge. Furthermore, when using a Kalman Filter, only the current state need be stored thereby reducing the amount of data that needs to be cached.

A survey/questionnaire to collect the perceptive influence of categories of risk and their factors on a project is under preparation. The values obtained from the questionnaire will be used to seed the initial values in the Kalman filter. Once the mathematical model based on the Kalman Filter has been developed, the implementation of QUART-ET can be completed.

5. Risk List

The major categories under Product Risks include failure history, internal dependency, external dependency, complexity, distributed components, data, environment/system configuration set up, unfamiliar, resources, global, and general. Below we include a description of each major category. For a complete list of the factors and corresponding factors for both product risks and process risks, see Appendix A.

The risk category of failure history represents features, modules, or systems that are either completely new to the project or have no failure
history, or existing components or systems that are being retested due to past failures. Internal dependency risks are related to the interdependency (upstream or downstream) of features, modules, or systems with other features, modules, or systems. Note that an upstream dependency is something whose failure will result in cascading failures in the remainder of the system, whereas a downstream dependency is something that is particularly sensitive to failures in the rest of the system [2]. Meanwhile, external dependency risks are risks based on the testing group’s dependency on other groups or departments to perform assigned tasks.

Complexity risks include features or modules that contain large amounts of code or are highly complex [2]. Risks associated with features, modules, and systems that are developed in separate locations or at separate times comprise the distributed components risk category. Lack of test data or changes in test data are also risks, and these risks comprise the category heading of data risks. The testing environment and system configuration are important to the testing process. Environment/system configuration setup covers the risks associated with the lack of a proper testing environment and system configuration.

The unfamiliar category represents the use of new or unknown technologies in a project and the development of unfamiliar features, which are also risks. For whatever reasons, management may not always provide testing groups with the necessary resources such as time, documentation, personnel, etc. to fulfill the testing team’s duties. This lack of resources can result in a product risk as well; the resources category encompasses these risks.

Testing execution and access to support availability may be affected by differences in time zones and/or dates. The global category covers these types of risks. Finally, the general category refers to risks that may be introduced when modifying or correcting errors in a module.

The major categories under Process Risks are divided into software development lifecycle workflows or disciplines, namely: requirements, design, testing, and all disciplines. The risks related to a discipline may be present if established techniques are not followed or tasks associated with the discipline are performed incorrectly. Some risks may occur within any of the major disciplines, hence the category all disciplines.

6. Conclusions

According to Boehm and DeMarco, “our culture has evolved such that owning up to risks is often confused as defeatism”. Project managers would be more inclined to admit to risks and perform risk mitigation if they knew that similar organizations have a risk management process in place.

This paper has reviewed the literature to provide readers with background knowledge about risk management and to discuss the benefits of risk-based testing. By understanding the basics of risk management, we can answer the questions posed in the introduction.

1. Which parts of the system need to be tested first?

2. How much testing is enough?

The answers to these questions follow.

1. High risk areas need to be tested first and more intensely than low risk areas.

2. We recommend the “good enough testing” concept as discussed in [19]. In other words, if a system part has sufficient benefits, no critical problems and if delaying it causes more harm than good, then “good enough testing” should be conducted on that system part.

By performing risk analysis, companies can make informed decisions and take proactive measures to avoid or eliminate the factors that lead to project failure [14]. However, when adopting a risk management procedure, a company should be aware that performing risk analysis has its own risks associated with it. In particular, Marvin Carr has noted that risk-aversion or deficient infrastructure and lack of organizational support can prevent a company from doing effective risk management [7].

The aim of the tool QUART-ET is to help a testing team to identify and prioritize risks associated with the testing process. By introducing the tool at FedEx, the company should see a natural cultural shift towards increased adoption of risk management strategies.

Acknowledgements

The authors would like to thank Dr. Mark Gillenson for his comments and contributions to the risk list.
Appendix A

Product Risks

- Failure History:
  o Feature/module/system has recent failure history (has failed tests before)
  o New feature/module (feature/module has no failure history)

- Internal Dependency
  o Adding or modifying a new module/feature/system that is likely to cause failure in other system features
  o Adding or modifying a new module/feature/system that is easily susceptible to failure when failure occurs in other system features

- External Dependency
  o Adding a requirement that will affect multiple modules
  o Changes/additions to business requirements require Software Configuration Change Board approval
  o Dependency on another system being successfully completed
  o Dependencies on external groups (legal, branding, etc)
  o Late arriving software

- Complexity
  o Adding or modifying a new feature/module that is highly complex
  o Modifying a feature/module with a large number of lines of code

- Distributed Components:
  o Adding or modifying a new module/feature/system that has components that have been developed at different locations
  o Adding or modifying a new module/feature/system that has components that have been developed at different times
  o Merging new feature/module with legacy feature/module/system

- Data:
  o Unavailability of specific test data
  o Modifying test data after testing has started (inconsistent data/source data)

- Environment/Configuration/System setup:
  o Test environment is not available
  o Improper configuration of system
  o Stable applications are not received early enough to modify/update the automation scripts
  o Test environment is unstable

- General
  o Modifying a feature/module

  o Feature/module implemented incorrectly

- Unfamiliar
  o Adding a new feature/module using a new/unknown technology
  o Adding special or non-traditional features

- Resources
  o Lack of adequate testing documentation
  o Resources not available

- Global
  o Time zone/date may impact testing execution
  o Time zone/date may impact contact/support availability

Process Risks

- Requirements discipline:
  o Pressure to meet hard deadline
  o Failure to produce acceptance test cases during requirements
  o Checklist for Quality Criteria (does a checklist exist? is it developed during requirements?)
  o Lack of requirements documentation
  o Requirements team did not include testing personnel
  o Modifying requirements late in development

- Testing discipline:
  o Inconsistency of testing process
  o Lack of full regression testing
  o Lack of performance testing
  o Lack of testing documentation
  o Lack of security testing
  o Lack of unit testing
  o Lack of integration testing
  o Lack of systems testing

- Design discipline:
  o Modifying the design late in development

- All disciplines:
  o Lack of stakeholder involvement
  o Outsourcing during any point in the project
  o Test team is not diverse enough (limits scope of testing)
  o Lack of backward traceability
  o Lack of forward traceability
  o Noncompliance with company process
  o Scope of project changes during testing
  o Required feature/module is missing (during reviews)
  o Lack of non-execution based testing (reviews, walkthroughs, inspections)
  o Failure to adequately test each level of the SDLC
References


