e-Journal of Practical Business Research

A Risk-based Approach to Testing Socio-Technical Systems
A Case Study at a Subsidiary of Daimler AG

Daniel Rheinbay

Erschienen im e-Journal of Practical Business Research
unter: http://www.e-journal-of-pbr.de

While concepts for software testing have been available for more than three decades, there is an ongoing discussion regarding the efficient resource allocation for software testing once these resources have become scarce. The following paper contributes to this discussion by suggesting a risk-based approach to software testing through the adoption of project management methods in the context of a real-world case study. After positioning the procedure of software testing within the context of IT service management, conventional ISO standards are consulted in order to identify which requirements need to be tested with the goal to safeguarding the transition of a closed-source software system. Additionally, corporate standards are incorporated before introducing applicable aspects of risk-based project management. As a result, a risk-enabled test object sheet is introduced which aims to facilitate putting the suggested approach into practice. Based on the experience gained during the case study following thereafter, recommendations for follow-up actions are generated.

Zitation: Rheinbay, Daniel (2009): A Risk-based Approach to Testing Socio-Technical Systems – A Case Study at a Subsidiary of Daimler AG. In: e-Journal of Practical Business Research, Ausgabe 8 (03/2009), DOI: 10.3206/0000000020

1. Introduction

Concepts to test the quality of software have been available for more than three decades (McCall et. al 1977). With the advent of Extreme Programming during the dot-com boom, test-driven approaches became a crucial part of agile software development patterns. Testing software is considered necessary not only to verify the implemented software complies with the given specifications, but also to validate that these specifications were appropriate in the first place. Though these integral parts of quality management assure the quality of the software from an engineer's point of view in great detail, they fail to address the fact that software itself cannot unfold its economic value added since they treat the object of investigation as an isolated entity. However, to put software into effect, it needs to be employed in a socio-technical system in which humans use the provided hard- and software to achieve their goals: In his studies, MacCormack (2003) found out that 50 to 70 percent of the total cost of ownership (TCO) of IT systems consists of the personnel required to make use of the software.

One of the major reasons causing these shortcomings is that software testing is typically done under the assumption that all aspects of the software are equally important (Fisher, M 2007).

Traditional project management methodologies, on the other hand, do address the interaction of different systems, and also take into account the human factor. To plan and verify that a pre-defined progress is made at a given date, for example, mile stones are commonly used. By evaluating the achievements made regarding certain mile stones, it is possible to estimate the level of over-all functionality that the projected system provides at any point in time. Some research has been done regarding the use of risk management to safeguard software projects by enabling project managers to react adequately (DeMarco, T & Lister, T 2003). However, these approaches are described on an abstract level, and mainly address the interests of those who develop software.

A recent survey showed that 45 percent of the polled companies in Germany do not have a strategy of what to test first when they are running out of time or budget (*Bei Softwarequalität hapert's*, 2008). Therefore, the goal of this paper is to contribute to the discussion and decision making of what to test first by suggesting a risk-based approach.

2. Course of Action

The first part of this article will give you an introduction to the conventional software testing approaches. In the second part, an overview of risk-based project management will be given. Subsequently, a risk-enabled approach for testing socio-technical systems will be suggested in the third part, which applies certain concepts of risk-based project management to the testing of IT systems. This will be exemplified through a case study at a subsidiary of Daimler AG. The paper concludes in concrete recommendations for follow-up actions and a critical assessment of the results.

Within the scope of the case study, information will be gathered by employing the method of semi-structured expert interviews as described by Kornmeier (2007). While standardized interviews are especially useful as a method to gather data for quantitative evaluations, the qualitative method of semi-structured interviews offers the opportunity to pose further questions in response to the interviewee's statements. This is of great benefit in the context of expert interviews, as they are especially suited for demanding subject areas (Bortz, J & Döring, N 2006). For these interviews, an outline will be used which can be found attached to this paper. The results of the interviews will be documented using the risk-enabled test object sheets, which will be introduced in "Proposal of a Risk-enabled Test Object Sheet" on page 15. As the forms may be used interactively, they can be found on the CD-ROM attached to the hardcopy of this article.

3. The Traditional Approach to Software Testing

This chapter focuses on the conventional approaches to software testing. At first, its context in IT Management will be given. Subsequently, the requirements as stated in ISO 9126 will be explored. Their relevance for the testing of socio-technical systems will be discussed afterwards. This will result in a selection of aspects which are most important for the following case study.

3.1 Context in IT Management

In the corporate world, IT service management is commonly performed according to the IT Infrastructure Library (ITIL). As IT services usually rely on more than one IT component (called Configuration Items, CI, in ITIL), the compatibility of those CIs needs to be ensured whenever changes to the infrastructure are made. This task is performed as part of the *Release Management* process (Schieferer, H & Schitterer, E 2006). However, with the advent of the IT Infrastructure Library (ITIL), the focus of IT management has shifted away from the delivery of singular IT products to the delivery of IT services (Kresse, M 2005). Accordingly, the testing of individual software pieces is not considered a core component of ITIL (Brunnstein, J 2006). While testing the CIs is required by ITIL, other standards need to be consulted in order to compile concrete requirements for software testing.

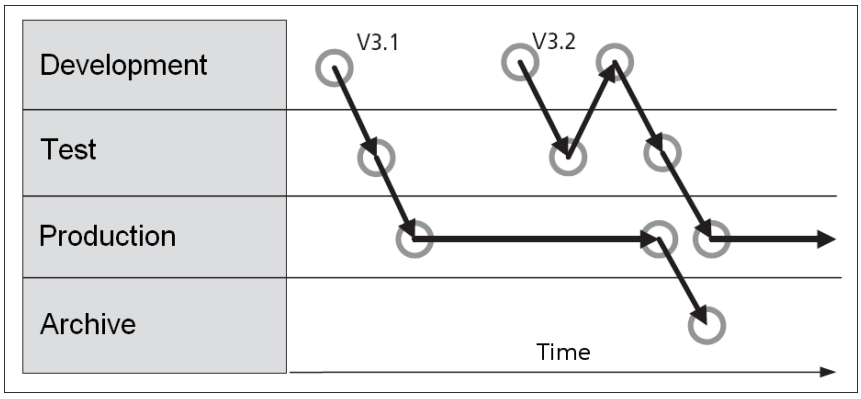

Figure 1: Testing is an essential mile stone during the release management process. Source: Following Bon, J 2004, p. 109.

3.2 ISO 9126 Standard

The ISO 9126 norm is an internationally accepted standard for the evaluation of software quality. A complementary standard focusing rather on the software creation processes than on the actual software development itself has been published as ISO 14598. However, the latter is beyond the scope of this paper.

The ISO 9126 standard divides requirements in *functional requirements* and *non-functional requirements*. However, no software will be developed in the following case study: Only *commercial, off-the-shelf software* (COTS) will be used. Thus, the goal of the following paragraphs is to decide which of the requirements as defined in ISO 9126 are actually applicable for the upcoming case study.

3.2.1 Functional Requirements

Functional requirements generally define *what* a system is required to be capable of. In ISO 9126, *functionality* of a system is made up of its *suitability*, *accurateness*, *interoperability*, *security* and *compliance*. To fulfill the criteria of suitability, a system's essential functionality has to conform to its technical specifications. While this accordance as well as the aspect of accurateness need to be ensured for custom-made systems, the suitability of COTS commonly depends on their parameterization to support the customer's (business) processes which are commonly not found in technical specifications (Kühnel, W 2006). Instead, the currently deployed version will be used as the reference to evaluate the suitability and accurateness of the new version. As the goal is to safeguard the changes made to a single system, testing its interoperability with the already existing environment is the main focus of this paper. Security is a concern at financial service providers (Assuring Compliance for Financial Services, 2006), thus this functionality aspect needs to be tested, too. It is noteworthy that compliance is considered a functional characteristic which can be attributed to interbranch government initiatives such as the Sarbanes-Oxley-Act (Lavi, I (n.d.)).

3.2.2 Non-Functional Requirements

Complementary to the functional requirements, non-functional requirements define *how* a system should perform its tasks. These requirements are necessary to quantify answers to rather subjective questions, e.g. "Do our customers get bored waiting for a result?" (Evans, I 2003, p. 3).

3.2.2.1 Usability

The *usability* of a system describes how much effort is needed to make use of its functionality. Examples for systems with a high usability are public transport ticket vending machines which enable the customer to purchase the most commonly bought ticket with the push of a single button (Krantz, P 2007). While the CEO of Axel Springer AG recently decided to switch to Apple computers because "they are easier to use than all the others" and the "most beautiful" ones (Fels, E 2008), pure usability and esthetics appears to be less of a concern for most companies.

3.2.2.2 Portability

Portability is the characteristic which describes how effortless the transition of a system from one environment to another can be accomplished (Franz, K 2007). The portability of an already existing system needs to be evaluated before it is put into production in order to protect the invested resources. While testing the portability of a system which is already in every-day use may yield relevant information for future management decisions, this appears to be less relevant for the purpose of safeguarding the actual transition of the system.

3.2.2.3 Maintainability

Maintainability aims to describe the amount of effort required to make changes to the system, which includes the correction of errors and the adjustment to changing requirements (Lavi, I (n.d.)). Like testing portability, testing the maintainability of a system which is already deployed in a production environment turns out to be a retrospective analysis of the IT management's decision to deploy the system in the first place. While reviewing such decisions may yield essential information for a sustainable management, it is beyond the scope of this paper.

3.2.2.4 Efficiency

It is often stated that taking into account the efficiency of a system is not necessary, as the ever-increasing performance of today's and tomorrow's computers makes inefficiency a negligible characteristic. Balzert (1999), however, states that the opposite is true: Efficiency needs to be monitored over the course of time, because as the computers' performance increase, so does the amount and complexity of the tasks which the systems shall cope with.

3.2.2.5 Reliability

Reliability describes the ability of a system to keep its performance on a given level under hostile circumstances, which is largely influenced by the degree of tolerance against invalid input data (Franz, K 2007). Especially for systems which are used by numerous individuals, the amount of effort required for a disaster recovery is of high priority (Limoncelli, T A, Hogan, C J & Chalup, S R, 2007).

3.2.3 Overview

Based on the above considerations, the following table sums up the quality requirements according to ISO 9126 which need to be tested to safeguard the transition of COTS-based systems:

Functional Requirements	Non-Functional Requirements
Suitability	~~Usability~~
Accurateness	~~Portability~~
Interoperability	~~Maintainability~~
Security	Efficiency
Compliance	Reliability

Table 1: Quality requirements as per ISO 9216 and their applicability for the transition of COTS-based systems. Source: Author's own.

3.3 Testing Strategies

Depending on the testing person's knowledge of the respective system's internals, they may choose different approaches to ensure the quality of software. Three common methods are described below, out of which one will be chosen as the most appropriate one for testing socio-technical systems.

3.3.1 Verification

In case of access to the development documentation as well as the source code of the software, a theoretical analysis of the implemented code following strict, formal rules can be performed to prove its correctness (Balzert, H 1999). This procedure is called *verification*. It aims at demonstrating that the implemented algorithms yield the correct output for any possible input. To prove the correctness of an entire system, every single one of its components has to pass the verification process in a bottom-up approach, suggesting the correctness of the entire system. Therefore, Brause (2005) considers verification an impractical scheme, being immensely heavy on financial and time resources (Listing, F 2008).

3.3.2 Glass-box testing

Knuth's (1977, p. 5) well-known quotation "Beware of bugs in the above code; I have only proved it correct, not tried it." demonstrates that verification itself does not ensure functionality at runtime. Therefore, complementary to the verification of the static source code, the goal of *glass-box testing* is to ensure that the software fulfills the documented requirements at runtime. This is done by testing the respective components using well-chosen test data, which is also referred to as *unit testing*. Schneider (2007) defines testing as "the execution of a program with the intention to find errors". It is therefore important to realize that only a fraction of the possible input values are used for testing purposes. Thus, unlike verification, glass-box testing can only prove the presence of errors; not their absence (Jézéquel, J-M & Meyer, B 1997). Since glass-box testing aims to cover as much of the system's source code as possible, an in-depth knowledge of the system's internals is required to create the test cases and to choose appropriate testing data. Therefore, it should be carried out by developers (Sneed, H M, Baumgartner, M & Seidl, R 2007).

Figure 2: Much like this transparent car body made of acrylic glass, access to the source code of software as provided by FLOSS allows inspecting the system's internals. Source: Shorey, B 2003.

3.3.3 Black-box Testing

Contrary to glass-box testing, *black-box testing* requires far less knowledge of the system's internals: Instead of verifying its correctness, only the behavior of the system's exposed interfaces are tested. Though it is unlikely that black-box testing will reveal such critical vulnerabilities as the infamous OpenSSL flaw from earlier this year (Debian Security Advisory 1571, 2008), IT management may choose a black-box testing strategy simply because verification and glass-box testing are not an option: While access to the source code is inherent to *Free/Libre Open Source Software* (FLOSS), neither design information (Zhu, H & Xudong H, 2005) nor the source code (Beydeda, S 2007) are

commonly available to the customers of most COTS-products. Though some Fortune 500 already make intensive use of FLOSS (PSA Peugeot Citroën Chooses SuSE Linux Enterprise Desktop from Novell, 2007), Hansen and Neumann (2007) give two predominant reasons why IT management may refrain from using FLOSS:

- Lack of knowledge for customization and maintenance of COTS can be compensated by outsourcing such activities to the software producer.
- By shifting responsibility towards the software producer in case a project fails, the decision makers feel less attackable.

Under these circumstances, a black-box testing strategy appears to be appropriate to safeguard the transition of an already existing COTS-based system which is in production use.

3.4 Choosing the Relevant Testing Stages

According to the testing guidelines of the Daimler subsidiary, a *testing stage* consists of several tests which focus on the same object of investigation. Once the software may be executed, testing stages are independent from each other as they have individual purposes. *Unit testing* is the stage that needs to be completed first to allow any other testing (Sneed, H M, Baumgartner, M & Seidl, R 2007), and is therefore not applicable when using COTS. *Performance testing* aims at ensuring the efficiency of a system. As mentioned above, efficiency should be monitored over the course of time even when using COTS and is therefore in the scope of this paper. Also, the new version of the system must not miss functions which were available in the previous version, so *regression tests* are applicable when transitioning COTS as well (Hoffmann, D W 2008). Finally, a disaster recovery test performed before the newly installed system goes into production use should be performed to make sure the system can be recovered within a given amount of time (Limoncelli, T A, Hogan, C J & Chalup, S R, 2007).

3.5 Choosing the Relevant Testing Phases

Testing Phases are periods of testing which are characterized by chronological interdependencies. Thus, they may only overlap under certain circumstances. The *technical single-system test* aims at ensuring the respective component works as expected in an isolated environment, whereas the functionality of the component in its environment is in the focus of the *technical integration test*. Complementary to the technical tests, the

functional single-system and *functional integration test* aim at ensuring that the implemented system supports the respective business processes as expected. Since IT should not be an end in itself, these testing stages are important especially when introducing new COTS (Kühnel, W 2006). Yet, as the regression tests mentioned above already cover the case that functionality from the previous version is missing in the new version, the functional single-system and functional integration test are beyond the scope of this paper.

3.6 Preparing the Test Concept

Based on the considerations above, the conventional testing approach includes the creation of a test concept. The testing guidelines at the Daimler subsidiary state that this should include the identification of so-called test objects: These are functional objects (e.g. “task”) or functional processes (e.g. “release management life cycle”) which shall be tested. Subsequently, test cases for the respective test objects shall be specified, including their target and the requirements that have to be met to pass the test, as well as its prerequisites. Additionally, a test concept should include the stakeholders and a testing schedule. This is where risk management comes into play to allocate the available resources adequately.

4. The Risk-based Approach to Project Management

This chapter will give you an introduction to DeMarco’s risk-based approach to managing software projects, which will serve as the basis for the risk-based component in the next chapter. Fisher (2007) defines *risk* using the following illustration:

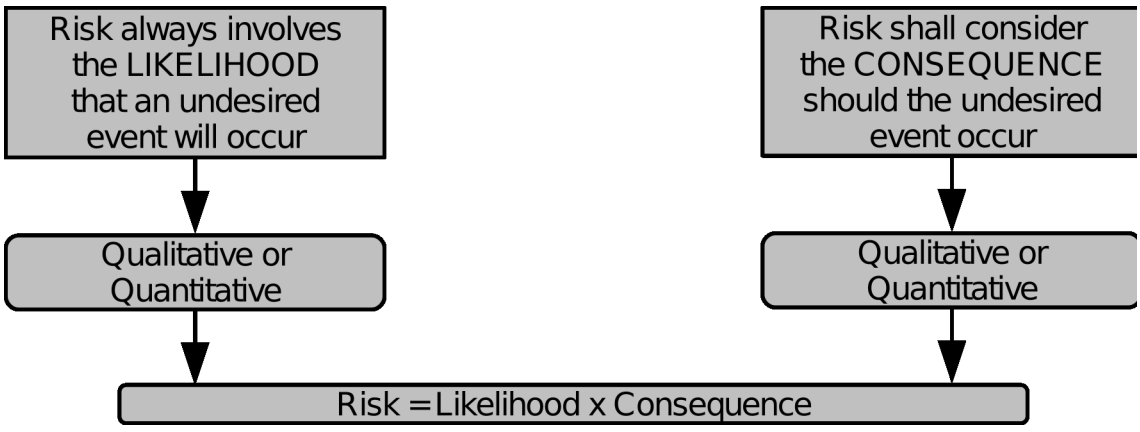

Figure 3: A risk involves the likelihood that an undesired event will occur and the consequence of the undesired event. Source: Following Fisher, M S 2007, p. 69.

4.1 Identifying the Risks

According to DeMarco and Lister (2003), the first step in risk-based project management is the identification of risks. While the top management will be able to come up with worst-case scenarios based on their experience from previous projects (Schneider, H & Marti, A 2006), it is important to offer non-executive employees the opportunity to submit further potentially undesired events which are usually compensated further down the hierarchy. Ahrents and Marton (2008) pull together several reasons why decision makers possess only a fraction of the available and required knowledge:

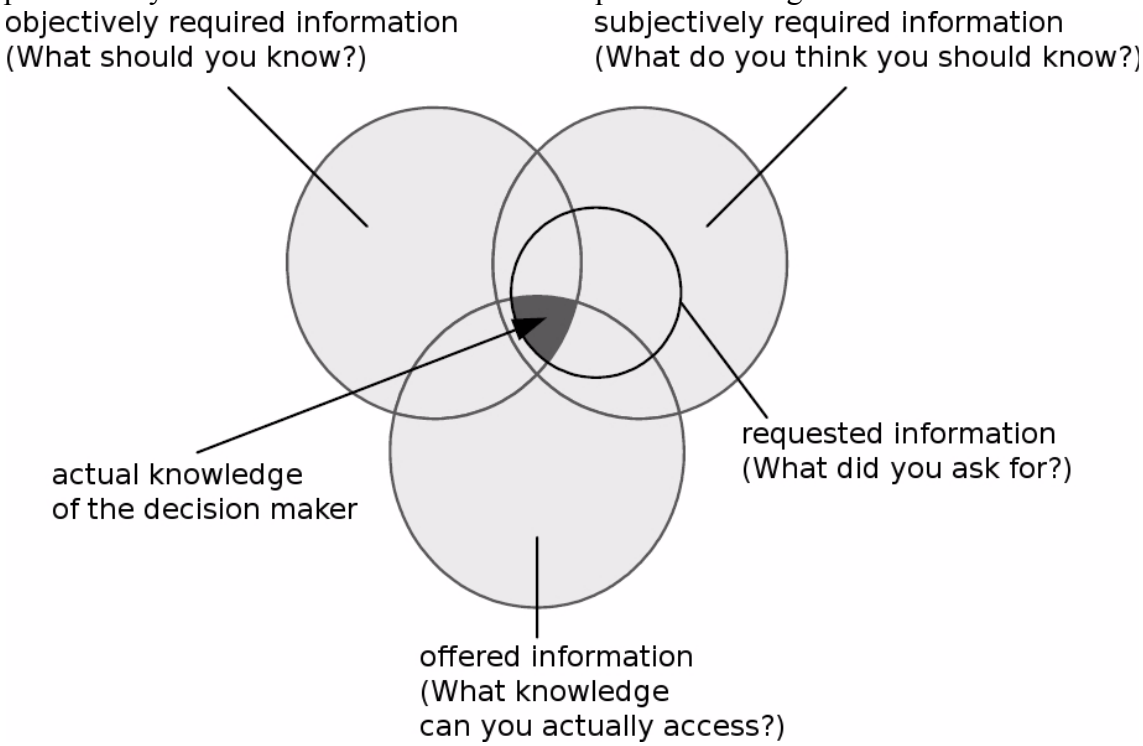

Figure 4: For a variety of reasons, decision makers possess only a fraction of the available and required knowledge. Source: Following Ahrendts, F. & Marton, A 2008, p. 62.

However, due to the habit of positive thinking which is predominantly expected in the unwritten commandments of the corporate world, the detailed depiction of worst-case scenarios is commonly a challenge (DeMarco, T & Lister, T 2003): Employees may fear they might be held responsible for the reasons which may cause the undesired events to materialize and thus may prefer to remain silent. While some companies offer anonymous email systems to offer their employees a way to letting the management know about possible risks (DeMarco, T 2001), other suggestions include moderated team events using brainstorming (Köhler, P 2005) or brainwriting methods such as the *6-3-5 method* (Kienpointer, M 1997).

4.2 Quantification

Once the risks have been identified, the second step is to quantify them. While Fisher's definition of risks allows for qualitative declarations of both the probability and the consequence, Schneider and Marti (2006) strongly encourage the quantitative approximation of the risks: Otherwise, nobody will feel responsible to ensure that the system's value-added actually materializes, yet the planned expenses will be made, resulting in a (economic) failure of the project (DeMarco, T & Lister, T 2003).

DeMarco and Lister (2003) assume that project managers are able to quantify the risks of their projects once they have been identified, and therefore do not give concrete recommendations of how to do so – besides pointing out that the value added and the risks shall be quantified with an equal precision. However, they do provide the following equation (ibid):

$$\text{Capital value} = \frac{\text{value added} - \text{expenses}}{\text{expenses}}$$

Plugging in concrete values results in an interest rate which allows the comparison of one project with different projects (and also with other investment opportunities).

4.3 Contingency Planning

In practical project management, analyzing the *critical path* of a project is an essential step during the preparation of the project activities. This analysis yields the information in what order the critical activities have to be executed to avoid running out of time (Kurbel, K E 2008).

To prevent a project from failing simply because one of the critical activities could not be executed (for whatever reason), DeMarco & Lister (2003) suggest *contingency planning*. This includes all activities related to the planning of those activities which are required to be executed as soon as a risk in fact materializes. These measures are required to be aimed at damage control.

4.4 Risk Mitigation

In addition to planning what to do in case a risk materializes, properly executed risk-based project management also defines measures which must be made to reduce the consequence of potentially undesired events. If unlimited resources were available, the same measures as defined for damage control could already be executed before the damage

occurs, thereby effectively mitigating the consequence of the undesired event to the extent that it will not have any impact on the project (DeMarco, T & Lister, T 2003). However, as risk-based project management is only applicable to situations with limited resources, this is not an option. Thus, additional activities have to be defined whose purpose is the mitigation of the consequences or the reduction of the probability of potentially undesired consequences, without wasting scarce resources. In order to be able to decide if a certain mitigation measure should be executed, indicators have to be defined and continuously monitored.

5. Applying the Risk-based Project Management to System Testing

In this chapter, certain concepts of the previous topic of risk-based project management will be applied to the testing of IT systems, thereby extending the conventional testing methods which were discussed in "The Traditional Approach to Software Testing" on page 3. Hence this chapter is structured similar to the previous one. The goal is to generate an approach to decide what to test first if resources are scarce.

5.1 Identifying the Risks

The conventional test concept as described in "Preparing the Test Concept" on page 9 called for the identification of test objects. As each object is related to functional requirements, this appears to be a vantage point from which to start the risk identification. Since assessing the risk related to a certain test object requires profound knowledge of the related processes and operational activities, quantitative surveys would result in less meaningful information. Therefore, it is suggested to take advantage of semi-structured expert interviews: While the general topic of the interviews is already predetermined based on the identified test objects, further questions maybe asked in case an expert deems it necessary to introduce further aspects in order to quantify the risks appropriately. In these interviews, the risk of each identified test object with regard to the quality requirements as described in "ISO 9126 Standard" on page 3 shall be assessed.

Before pushing a product to the market, producers of COTS are obliged to test its functionality in the environments which they certified to be compatible (Masak, D 2005). Thus, and in order to keep the additional work imposed on the experts as low as possible,

the functional requirements will be grouped to one aspect. So, during the interviews, the risk of functionality, efficiency and reliability shall be evaluated by the experts.

5.2 Quantification

As stated above, DeMarco and Lister (2003) assume that once a certain undesired event has been identified, its impact on the project is obvious and thus its quantification is simple. While this may be true for project management, this is not applicable to the quantification of the activities which are required to fix the code related to the respective testing object when using COTS, since the source code of COTS-systems is rarely available to the customer (Beydeda, S 2007).

Instead, the focus should rely on the value-added of the respective object, which the customer should be able to evaluate (Heinrich, B, Seifert, F & Wehrmann, A, 2006) based on their decision why they chose this specific system in the first place. If not, a fall-back option will be given at the end of the following paragraph.

5.3 Contingency Planning

Once the test objects have been identified, the next step is to identify their interdependencies. For the purpose of software testing, the method of identifying critical paths should be applied with respect to the proper execution of the business processes which the IT system shall support. This analysis yields the information which test object are critical to the successful execution of the respective business processes.

When the critical test objects of a system have been identified, the next step is to plan what to do in case one of these test objects fails to work as expected after the system has been introduced to the production environment. To do so, it is necessary to regard the respective system as a socio-technical entity. This means that, as stated in the introduction of this paper, humans are required to take advantage of the soft- and hardware. Thus, the following key question from IT landscape management may be called upon: Is it beneficial to further integrate certain IT systems to reduce the amount of human interaction, or do the expenses required for the integration exceed the cuttings in the cost of human labor (Masak, D 2005)? By looking closer at this question, it becomes obvious that the following assumption is at its bottom: If the technical component of the socio-technical system is not available, then human labor can compensate for that. Based on this suppo-

sition, the expenses required to hire human resources for this task are the equivalent value necessary for a potential damage control, should the test object not work in the production environment.

If no concrete value-added could be quantified in the previous paragraph, which is applicable to half the systems at financial services providers (El Hage, B & Bechmann, T 2002), then this amount may also be considered the value-added as it is not need to be spent if the test object works as expected. For the probability values, it is suggested to derive concrete figures based on the experience of previous migrations, taking into account the manufacturer's upgrade documentation and knowledge base articles.

5.4 Risk Mitigation

As testing aims to ensure the quality of a product, it is by definition an activity which mitigates risks. However, as almost three quarters of the IT projects polled by the Standish Group in 2004 were not finished on time (Heinrich, B, Seifert, F & Wehrmann, A, 2006), most projects appear to be on a tight schedule. As testing can hardly be performed before a certain degree of development has been done (Sneed, H M, Baumgartner, M & Seidl, R 2007), the schedules for testing are even tighter (Listing, F 2008). To avoid worsening this situation, no further delays should be introduced. With regard to system testing, these measures should especially focus on maintaining the required testing environments. This is not specific to certain testing objects, but generally applies to testing activities.

5.5 Risk Indicator Monitoring

If the release of the system to be tested is coordinated using a release management process (Bon, J 2004), the developers should not able to make any unexpected changes to the system which is being tested. Thus, monitoring indicators in order to be able to respond to a worsening condition of the test objects is not required. Instead, indicators for upcoming risks may be derived by

- tracking management decisions which affect the system being tested,
- staying in touch with the system's technical system owner (TSO) as she might know about upcoming changes which will need to be re-tested,
- following up on the development progress of the involved components.

5.6 Proposal of a Risk-enabled Test Object Sheet

Based on the considerations of this chapter, a risk-enabled test object sheet is suggested on the following page which aims at facilitating putting the suggested testing approach into practice. The purpose of this sheet is not to replace the conventional test concept, but to enhance it. As the test concept already includes the information regarding the test cases, test data etc., it should not be kept redundantly on the proposed sheet. The form consists of a Microsoft Word Document file, allowing a wide-spread use throughout the company. For each identified test object, its quantified values may be filled in. The form then calculates the associated risks based upon these values with the click of a button. Clicking the button triggers the execution of Visual Basic for Applications (VBA) code, which performs the actual calculation. The suggested form may also be used to keep track of re-evaluations of the associated risks. Taking advantage of this sheet will ease the decision whether further testing of a given object is appropriate or if the resources should rather be spent on testing a different object.

Risk-enabled Test Object Sheet

1. General Information

Unique name:	
Owner:	Day of Discovery:
Description:	

2. Risk Assessment

2.1 Functionality

Potential consequences:	Likelihood:
Indicators:	

2.2 Efficiency

Potential consequence:	Likelihood:
Indicators:	

2.3 Reliability

Potential consequence:	Likelihood:
Indicators:	

2.4 Re-evaluation protocol

Calculate Risk Total risk of this test object:

6. Putting the Concept into Practice

In the previous chapter, a risk-based approach to testing socio-technical systems has been suggested. The following paragraphs first give you an overview of the initial situa-

tion, followed by a portrayal of the nominal condition. Employing the previously introduced risk-enabled test object sheet, the third section of this chapter demonstrates how the concept of risk-based testing can be put into practice by giving examples of risk-enabled test objects and summing up the observations made during the case study.

6.1 Initial State

The case study was conducted at a subsidiary of Daimler AG. As Daimler AG is listed at the New York Stock Exchange, it has to comply with the Sarbanes-Oxley-Act (SOX) (Wiethoff, F 2008). SOX is an interbranch law introduced by the US Administration as a response to corporate and accounting scandals. SOX aims at recovering the public's confidence in securities markets by requiring corporations to implement extensive controls, including IT controls: Unauthorized changes of software are no longer an issue limited to the companies, but in fact affect the companies' compliance with applicable law (Marks, N 2008). Therefore, an effective change management system is required which tracks software changes in a revision-safe manner. For this purpose, the company opted for a COTS-based system from one of the market-leading software producers. The system consisted of a source code versioning system whose meta-data was stored in a database. On top of this versioning system, a web-based change management system coordinated the life cycle of each change request. This included capturing the information which is relevant for the respective change request, assigning tasks to developers who process the respective change request, and the activities necessary to deploy the changes to the test environments. The last step in the life cycle is to propagate the changes either to the upcoming release of the respective software or to the currently deployed production system, depending on the priority of the change.

As the software producer discontinued the support of the initially deployed version of the change management system, it was bound to become a legacy system. Running such legacy systems should be avoided whenever possible for economic reasons (Masak, D 2006), as the expenses for their maintenance increase disproportionately to their age. The hardware which the change management software was running on was equally outdated.

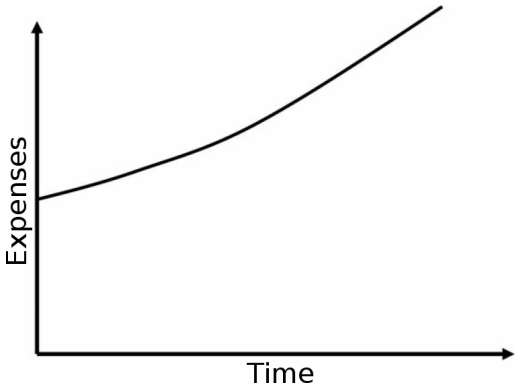

Figure 5: Growth of the expenses required for the maintenance of legacy systems in relation to their age.

More than 500 employees used the change management system in order to coordinate more than 12.000 changes per year. It was therefore a critical component for the success of the release management process.

6.2 Nominal Condition

The nominal condition required to run a supported, up-to-date and stable version of the change management system on recent hardware in order to prevent the change management system from becoming a legacy system. All data was required to be migrated, including the source-code objects, reports and queries which the users created. The deadline for the migrated system to go into production use was the end of July, as this was the point in time at which the release management started to prepare their upcoming fall release. As neither hiring additional testers nor releasing the experts who are in charge of the change management system from their regular work in the versioning & configuration team was an option, the resources which were available were effectively limited. Therefore, the situation required a way to prioritize what to test first.

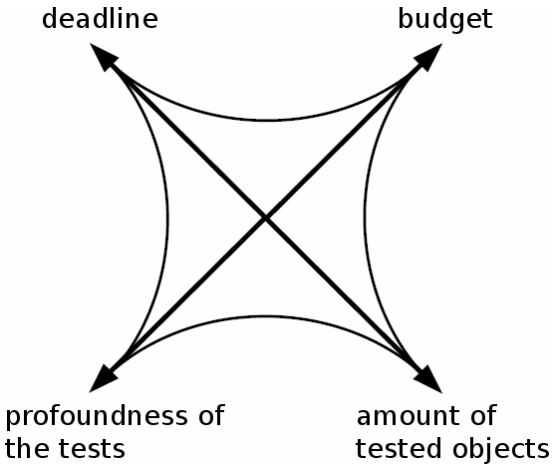

Figure 6: It is impossible to improve on any one of the four dimensions without worsening on at least another one.
Source: Following Ahrendts, F & Marton, A 2008, p. 35.

6.3 Examples of Risk-Enabled Test Objects

One of the more critical test objects was the interface to the change management system of a major German IT services provider. As this provider operates critical services, approximately 1000 change requests are communicated using this interface per year. Having been developed in-house, the compatibility of the interface with the new version of the change management system could not be assured by the software producer. The peculiar singularity of this test object was that it turned out the impact of a worsened reliability would have been three times higher than the activities required if the interface had not worked at all. Relying on emails in order to exchange the change requests between the two systems, the interface lacked a way for the receiving party to be automatically notified that they missed a change request if the email did not arrive. Had the interface not worked reliably, manual research would have been necessary in order to find out which change requests were not exchanged, outweighing the resources required to manually exchanging the tickets in the first place by a factor of three. So, while the automated exchange of the change requests eased the communication with the service provider a lot as long as it worked reliable, its existence in fact would have been counterproductive if it had not worked reliably.

Unlike the previous one, a test object whose risk regarding its reliability is considerably less is the process of resetting a user's password: While this activity was automated as well, it occasionally required one of the experts who were interviewed to engage in the process. However, as the user who asked for the password reset would have noticed if his request had not been put into action, it was unlikely that any damage would be caused without being noticed.

6.4 Evaluating the Filled Out Sheets

In contrast to the risk of the initially described test object, no more efforts beyond executing the automated tasks manually would have been required, had one of the interfaces to the project management system, the reporting system or the integrated development environments failed. This is because in contrast to the interface to the IT service provider, the reliability of the other interfaces was being monitored continuously.

It is noteworthy that in the context of this particular case-study, efficiency was not a single time considered a risk factor. The three interviewed experts pointed out concordantly

that while their every-day work was dependant on the reliability and functionality of the test objects which automated the exchange of data between different systems, the efficiency of these test objects was not critical in terms of time: Should a change request be required to be pushed to the production environment urgently, their usual course of action includes getting in close personal contact with all the stakeholders in order to ensure a quick passage of the change request through its life cycle.

During the interviews, it turned out that neither the value nor the risks related to central test objects which provide the core functionality of the system, such as the versioning of source code objects, can be quantified clearly: Doing so would amount to a complex valuation of a part of the IT landscape (Heinrich, B, Seifert, F & Wehrmann, A, 2006). The effort required for such an analysis is in no way justified, as it is up to the producer of the COTS component to ensure its core functionality (Masak, D 2005).

7. Recommendations for Follow-Up Actions

Based on the experience from the case study which was described in the previous chapter, it is recommended to implement risk-based testing to support the decision making of what to test first if only limited resources are available: The approach is well-founded on already established project management methods, and requires only little effort to be put into practice. In this context, the policies for the approximation of the consequences and likelihood of risks should be implemented company-wide to allow for the comparison of different projects. Therefore, further research is proposed to improve the precision of the risks related to the long-term operation of the respective systems, for example by combining the discounted cash-flow method which is commonly used in corporate finance (Weißmann, F 2005) with the Poisson distribution (Mosler, K & Schmid, F 2006).

As literature suggests (Köhler, P 2005), it proves to be beneficial to have the management's support during the introduction of risk-based testing, as this method heavily relies on contributions from co-employees regarding the identification and evaluation of possible risks.

Once additional approaches for the allocation of testing resources have been published, IT management should re-evaluate which one of them appears to be most applicable to their specific environment.

8. Critical Assessment

This article contributes to the current discussion of how companies may prioritize what to test first in case their resources are limited by suggesting an approach for risk-based testing. As the case study in the previous chapter demonstrated, the approach can be put into practice to safeguard the migration of COTS-based systems. However, its adaptability for large-scale projects has yet be proven, as this is beyond the scope of this article.

Before any advantages may materialize, at first additional work is imposed on the involved stakeholders. In this context, two aspects are noteworthy: Firstly, the same effect also applies to other processes of IT management, such as the entire release management process according to ITIL (Schieferer, H & Schitterer, E 2006). Secondly, if quantifying the risks or value-added of a given system appears to be sheer impossible, often times the company's resources should rather be spent on a different system (DeMarco, T & Lister, T 2003).

Reference List

Ahrendts, F. & Marton, A 2008, *IT-Risikomanagement leben!,* Springer-Verlag Berlin, Heidelberg; Berlin.

Balzert, H 1999, *Grundlagen der Informatik*, Spektrum Akademischer Verlag, Heidelberg; Berlin.

'Bei Softwarequalität hapert's', 2008, *Computer Zeitung*, vol. 2004, no. 28, p. 4.

Beydeda, S 2007, 'STECC: Selbsttestende Software-Komponenten', *Informatik - Forschung und Entwicklung*, vol. 21, no. 3-4, pp. 243-253.

Bon, J 2004, *IT Service Management eine Einführung basierend auf ITIL*, Van Haren Publishing, Zaltbommel.

Bortz, J & Döring, N 2006, *Forschungsmethoden und Evaluation für Human- und Sozialwissenschafftler*, Fourth Edition, Springer Medizin Verlag, Heidelberg.

Brause, R 2005, *Kompendium der Informationstechnologie*, Springer-Verlag Berlin Heidelberg, Berlin; Heidelberg; New York.

Brunnstein, J 2006, *ITIL Security Management realisieren*, Friedr.Vieweg & Sohn Verlag, Wiesbaden.

DeMarco, T & Lister, T 2003, *Bärentango: Mit Risikomanagement Projekte zum Erfolg führen*, Carl Hanser Verlag, München; Wien.

Fisher, M S 2007, *Software Verification and Validation: An Engineering and Scientific Approach*, Springer Science+Business Media, USA.

Franz, K 2007, *Handbuch zum Testen von Web-Applikationen*, Springer-Verlag Berlin Heidelberg, Berlin; Heidelberg; New York.

Hansen, R & Neumann, G. 2005, *Wirtschaftsinformatik 1*, Ninth Edition, Lucius & Lucius, Stuttgart.

Heinrich, B, Seifert, F & Wehrmann, A, 2006, 'Quantitatives IT-Portfoliomanagement, Risiken von IT-Investitionen wertorientiert steuern', *Wirtschaftsinformatik*, vol. 48, no. 4, pp. 234-245.

Hoffmann, D W 2008, *Software-Qualität*, Springer-Verlag, Berlin; Heidelberg.

Jézéquel, J-M & Meyer, B 1997, 'Put it in the Contract: The Lessons of Ariane', *IEEE Computer*, vol. 30, no. 7, pp. 129-130.

Kienpointer, M 1997, 'On the Art of Finding Arguments: What Ancient and Modern Masters of Invention Have to Tell Us About the "Ars Inveniendi"', *Argumentation*, vol. 11, no. 2, pp. 225-236.

Knuth, D 1977, *Notes on the van Emde Boas construction of priority deques: An instructive use of recursion*, Stanford University, Stanford.

Köhler, P 2005, *PRINCE2*, Springer-Verlag Berlin Heidelberg, Berlin; Heidelberg; New York.

Kornmeier, M 2007, *Wissenschaftstheorie und wissenschaftliches Arbeiten*, Physica-Verlag, Heidelberg.

Kühnel, W 2006, 'Quo vadis Pflichtenheft?', *Das Logistik-Magazin*, vol. 2006, no. 5, pp. 50-51.

Kresse, M et al. 2005, *ITSM Advanced Pocket Book*, Serview GmbH, Bad Homburg.

Kurbel, K E 2008, *The Making of Information Systems*, Springer-Verlag Berlin, Berlin; Heidelberg.

Limoncelli, T A, Hogan, C J & Chalup, S R 2007, *The Practice of System and Network Administration*, Second Edition, Addison-Wesley, Boston.

Listing, F 2008, 'Was den Code stark macht', *ElektronikPraxis*, vol 2008, no 5, pp. 38-40.

Masak, D 2005, *Moderne Enterprise Architekturen*, Springer Berlin Heidelberg, Berlin; Heidelberg; New York.

Masak, D 2006, *Legacysoftware*, Springer-Verlag Berlin Heidelberg, Berlin; Heidelberg; New York.

Mosler, K & Schmid, F 2006, *Wahrscheinlichkeitsrechnung und schließende Statistik*, Second Edition, Springer-Verlag Berlin Heidelberg, Berlin; Heidelberg; New York.

Schieferer, H & Schitterer, E 2006, *Prozesse optimieren mit ITIL*, Friedr. Vieweg & Sohn Verlag, Wiesbaden.

Schneider, K 2007, *Abenteuer Softwarequalität*, dpunkt.verlag, Heidelberg.

Schneider, H & Marti, A 2006, *Krisen vermeiden in IT-Projekten*, Springer-Verlag Berlin Heidelberg, Berlin; Heidelberg; New York.

Sneed, H M, Baumgartner, M & Seidl, R 2007, *Der Systemtest*, Carl Hanser Verlag, München; Wien.

Weissmann, F 2005, *Unternehmen steuern mit Controlling*, Springer-Verlag Berlin Heidelberg, Berlin; Heidelberg; New York.

Zhu, H & Xudong, H 2005, 'A Methodology of Component Integration Testing', in Beydeda, S & Gruhn, V (eds) *Testing Commercial-off-the-Shelf Components and Systems*, Part II, Springer-Verlag Berlin Heidelberg, Berlin; Heidelberg; New York.

List of Internet Sources

Assuring Compliance for Financial Services, 2006, NetIQ Corporation, viewed July 21 2008, <http://download.netiq.com/CMS/WHITEPAPER/NetIQIndustryWP_FSI.pdf>

Debian Security Advisory 1571, 2008, Software in the Public Interest, Inc., viewed June 29 2008, <http://www.debian.org/security/2008/dsa-1571>

DeMarco, T 2001, e-Talk: DeMarco, Tom, 22 February 2001, viewed July 3 2008, <http://www.stickyminds.com/sitewide.asp?Function=edetail&Object-Type=ART&ObjectId=2384>

Evans, I 2003, *Common Sense Quality,* IE Testing Consultancy, viewed June 20 2008, <http://www.fmnet.info/fortest/documents/Evans_sheffield.pdf>

Fels, E 2008, *Axel Springer AG stellt Arbeitsplatzsysteme konzernweit auf Apple um*, Axel Springer AG, viewed July 6th, <http://www.axelspringer.de/presse/Axel-Springer-AG-stellt-Arbeitsplatzsysteme-konzernweit-auf-Apple-um_154973.html>

Krantz, P 2008, *Interacting With a Stockholm Public Transport Ticket Vending Machine*, viewed June 30 2008, <http://www.peterkrantz.com/2007/man-machine-interface/>

Lavi, I (n.d.), *ISO 9126 Software Quality Model*, Jerusalem College of Technology, viewed July 12 2008, <http://sukka.jct.ac.il/~itzhakl/Additional%20Material/ISO%209126%20Software%20Quality%20Model.doc>

MacCormack, A 2003, *The True Costs of Software*, Computerworld, viewed July 8 2008, <http://www.computerworld.com/softwaretopics/os/windows/story/0,10801,81590,00.html>

Marcs, N 2008, *Sarbanes-Oxley Section 404: A Guide for Managment by Internal Controls Practitioners*, The Institute of Internal Auditors, viewed July 5 2008, <http://www.theiia.org/download.cfm?file=31866>

PSA Peugeot Citroën Chooses SuSE Linux Enterprise Desktop from Novell, 2007, Novell, Inc., viewed June 25 2008, <http://www.novell.com/news/press/psa_peugeot_citro_eumln_chooses_suse_linux_enterprise_desktop_from_novell>

Shoey, B 2003, *1940 pontiac transparent car 03*, viewed June 27 2008, <http://www.shorey.net/Auto/American/GM/Pontiac/1940%20pontiac%20transparent%20car-03.jpg>

Slonim, Y 2005, *The Software Quality life cycle*, Dr. Dobb's Journal, viewed June 29 2008, <http://www.ddj.com/article/printableArticle.jhtml?articleID=184407853&dept_url=/architect/>

Wiethoff, F 2008, *Sarbanes-Oxley Act (SOX)*, KPMG Deutsche Treuhand-Gesellschaft AG, viewed June 30 2008, <http://www.kpmg.de/Themen/1439.htm>

Appendix

Interview Outline

General Topic: Identification of risk objects.

Key questions:

- What are the interviewee's main activities with regard to the change management system?
- What test cases are appropriate to ensure that these activities can be performed after the migration of the system?
- How can these test cases be grouped to functional test objects?
- Have any of the interviewee's previous or current activities been automated?
- How much time is required to manually compensate for an undesired consequence if either the functionality, the efficiency or the reliability is not provided as expeced?
- How often has this been necessary in the past?
- Which of the test objects related to the automation of activities are prone to fail during migrations of the change management system, based on the experience from previous upgrades?