A Risk Assessment Framework for Software Testing

Michael Felderer, Christian Haisjackl, Viktor Pekar, and Ruth Breu

Institute of Computer Science, University of Innsbruck, Austria
{michael.felderer,christian.haisjackl,viktor.pekar,ruth.breu}@uibk.ac.at

Abstract. In industry, testing has to be performed under severe pressure due to limited resources. Risk-based testing which uses risks to guide the test process is applied to allocate resources and to reduce product risks. Risk assessment, i.e., risk identification, analysis and evaluation, determines the significance of the risk values assigned to tests and therefore the quality of the overall risk-based test process. In this paper we provide a risk assessment model and its integration into an established test process. This framework is derived on the basis of best practices extracted from published risk-based testing approaches and applied to an industrial test process.

Keywords: Risk Assessment, Risk Identification, Risk Analysis, Risk Evaluation, Risk-Based Testing, Risk Management, Software Testing.

1 Introduction

Risk-based testing (RBT) is a pragmatic and well-known approach to address the problem of ever limited testing resources that recently gained much attention [1]. It is based on the intuitive idea to focus test activities on those scenarios that trigger the most critical situations for a software system [2]. Its appropriate application may then have several benefits. RBT optimizes the allocation of resources (budget, time, persons), is a means for mitigating risks, helps to early identify critical areas, and provides decision support for the management. Risk-based testing involves the identification, analysis and evaluation of product risks, which are together referred to as risk assessment, and the use of risks to guide the test process [3].

Because risk identification, analysis, and evaluation determine the significance of the risk values assigned to tests and therefore the quality of the overall test process, they are core activities in every risk-based test process. Although several RBT approaches are available [4], and the upcoming international standard ISO/IEC 29119 [5] on testing techniques, processes, and documentation even requires the consideration of risks as an integral part of the test planning process, a framework on how to integrate risk assessment in a test process has not been proposed. But such a framework provides guidelines and supports test and process managers to establish a risk-based test process on the basis of an existing test process.

T. Margaria and B. Steffen (Eds.): ISoLA 2014, Part II, LNCS 8803, pp. 292–308, 2014.

The objective of this paper is to provide a framework for integrating risk assessment, i.e., risk identification, analysis, and evaluation, into an established test process. The framework contains a risk assessment model which configures the risk-based test process. It is derived on the basis of best practices extracted from published RBT approaches and applied to an industrial test process.

The remainder of this paper is structured as follows. Section 2 discusses background on risk-based testing and related work. Section 3 defines a risk assessment framework for testing purposes. Section 4 shows how this model is applied in industrial projects. Finally, Section 5 concludes the paper and presents future work.

2 Background on Risk-Based Testing

Risk-based Testing (RBT) is a type of software testing that considers risks assigned to risk items for testing activities [6,7]. In risk-based testing, testing activities are supported by risk management activities. It therefore integrates a risk management process into a test process. In this section we discuss background on the concept of risk (Section 2.1), test and risk management processes (Section 2.2) as well as RBT approaches (Section 2.3).

2.1 Concept of Risk

A risk is the chance of injury, damage or loss and typically determined by the probability of its occurrence and its impact [8]. As it is the chance of something happening that will have an impact on objectives [9], the standard risk formalization [3] is based on the two factors probability (P), determining the likelihood that a failure assigned to a risk occurs, and impact (I), determining the cost or severity of a failure if it occurs in operation. Mathematically, the risk exposure R of an arbitrary asset a, i.e., something to which a party assigns value, is determined based on the probability P and the impact I in the following way:

$$R(a) = P(a) \circ I(a)$$

In the context of testing, assets are arbitrary testable artifacts also called risk items. For instance, requirements, components, security risks or failures are typical risk items to which risk exposure values R as well as tests are assigned. Within testing, a risk item is assigned to test cases which are typically associated with risk exposure values themselves derived from the risk items' risk exposure values. Risk exposure is sometimes also called risk coefficient, risk value or not distinguished from the risk itself. The depicted operation $\circ$ represents a multiplication of two numbers or a cross product of two numbers or letters (and can principally be an arbitrary mathematical operation used to determine risk). The factors P and I may be determined directly via suitable metrics or indirectly via intermediate criteria based on the Factor-Criteria-Metric model [10]. The probability typically considers technical criteria like complexity of components assigned to the risk item and the impact considers business criteria like monetary loss. The metrics can be measured

automatically, semi-automatically or manually. For instance, the complexity of a component can be estimated automatically by the McCabe complexity and the monetary loss can be estimated manually by a customer. Based on the determined metrics, risk exposure values are computed on the basis of a calculation procedure. Finally, risk exposure values are assigned to risk levels. A risk level [3] indicates the criticality of risk items and serves the purpose to compare risk items as well as to determine the use of resources, e.g., for testing. Risk levels are often defined via risk matrices combining probability and impact of a risk. An example for a risk matrix is shown in Fig. 1.

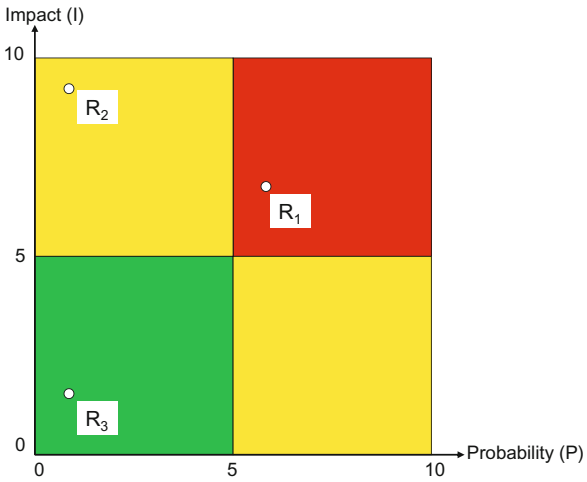

Fig. 1. Risk Matrix Example

The 2x2 risk matrix of Fig. 1. Probability and impact range from 0 to 10 and are shown on the x-axis and y-axis, respectively. Items in the lower left cell ($[0..5] \times [0..5]$) have low risk, items in the upper right cell ($[5..10] \times [5..10]$) have high risk, and items in the remaining cells ($[0..5] \times [5..10]$ and $[5..10] \times [0..5]$) have medium risk. For instance, risk R_1 in Figure 1 with value 6×7 is high, R_2 with value 1×9 is medium, and R_3 with value 1×2 is low.

2.2 Basic Concepts of Test and Risk Management Processes

A *test process* contains the core activities *test planning*, *test design*, *test implementation*, *test execution* as well as *test evaluation* [3]. Test planning is the activity of establishing or updating a test plan. A test plan is a document describing the scope, approach, resources, and schedule of intended test activities [3]. During the test design phase the general testing objectives defined in the test plan are transformed into tangible test conditions and test cases. Tests are then implemented which contains remaining tasks like preparing test harnesses and test data,

or writing automated test scripts which are necessary to enable the execution of the implementation-level test cases. The tests are then executed and all relevant details of the execution are recorded in a test log. During the test evaluation and reporting phase, the exit criteria are evaluated and the logged test results are summarized in a test report. Development projects typically contain several test cycles and therefore all or some phases of the test process are performed iteratively.

A *risk management process* contains the core activities *risk identification*, *risk analysis*, *risk evaluation*, *risk treatment*, and *risk monitoring* [9]. In the risk identification phase risk items are identified. In the risk analysis phase the probability and impact of risk items and hence their risk exposure values are estimated. In the risk evaluation phase, the significance of risk is assessed based on the estimated risk exposure values. As a consequence, risk items may be assigned to risk levels defining a risk classification and a prioritization. In the risk treatment phase actions for obtaining a satisfactory situation are determined and implemented. In case of risk-based testing, testing is applied as a measure to treat risks. In the risk monitoring phase risks are tracked over time and their status is reported. In addition, the effect of the implemented actions is determined. The activities risk identification, risk analysis, and risk evaluation are often collectively referred to as *risk assessment*, while the activities risk treatment and risk monitoring are referred to as *risk control*. As in the context of RBT, testing is per definition applied for risk control, only risk assessment, i.e., risk identification, analysis, and evaluation, has to be integrated into the test process as a separate activity.

2.3 Risk-Based Testing Approaches

The overall purpose of RBT approaches is to test in an efficient and effective way driven by risks. As mentioned before, every available risk-based testing approach therefore integrates testing and risk assessment activities. Several RBT approaches have been proposed in scientific conferences and journals. We systematically extracted these approaches from comprehensive related work sections of four recently published journal articles on risk-based testing [11,4,12,13] to get a broad and representative overview of RBT approaches. We considered all RBT approaches defined in the journal articles themselves as well as all RBT approaches cited in at least one related work section of the four journal articles. To guarantee evidence of the approaches and enough details to extract relevant information, we considered only RBT approaches reported in papers with a length of at least four pages published in a scientific journal or in conference proceedings. Table 1 lists all collected RBT approaches ordered by the date of their first publication. Some approaches, i.e., Redmill, Stallbaum, Souza, as well as Felderer and Ramler are covered by more than one cited publication (see entries with identifiers 03, 04, 05 and 13 in Table 1). Most listed approaches are cited by more than one journal article which is an additional indicator for the relevance of the RBT approaches collected in Table 1.

Table 1. Overview of Identified Risk-based Testing Approaches

ID	Approach	Description
01	Amland [6]	The approach defines a process which consists of the steps (1) planning, (2) identification of risk indicators, (3) identification of cost of a fault, (4) identification of critical elements, (5) test execution as well as (6) estimation to complete. In addition, it is presented how the approach was carried out in a large project.
02	Chen et al. [14]	The approach defines a specification-based regression test selection with risk analysis. Each test case is a path through an activity diagram (its elements represent requirements attributes) and has an assigned cost and severity probability. The test selection consists of the steps (1) assessment of the cost, (2) derivation of severity probability, and (3) calculation of risk exposure for each test case as well as (4) selection of safety tests. The risk exposure of test cases grouped to scenarios is summed up until one runs out of time and resources. The approach is evaluated by comparing it to manual regression testing.
03	Redmill [15,16]	The approach reflects on the role of risk for testing in general and proposes two types of risk analysis, i.e., single-factor analysis based on impact or probability as well as two-factor analysis based on both factors.
04	Stallbaum et al. [17,18]	The approach is model-based. Risk is measured on the basis of the Factor-Criteria-Metrics model and annotated to UML use case and activity diagrams from which test cases are derived.
05	Souza et al. [19,20]	The approach defines a risk-based test process including the activities (1) risk identification, (2) risk analysis, (3) test planning, (4) test design, (5) test execution, as well as (6) test evaluation and risk control. In addition, metrics to measure and control RBT activities are given. The approach is evaluated in a case study.
06	Zimmermann et al. [21]	The approach is model-based and statistical using Markov chains to describe stimulation and usage profile. Test cases are then generated automatically taking the criticality of transitions into account. The approach focuses on safety-critical systems and its application is illustrated by examples.
07	Kloos et al. [22]	The approach is model-based. It uses Fault Tree Analysis during the construction of test models represented as state machine, such that test cases can be derived, selected and prioritized according to the severity of the identified risks and the basic events that cause it. The focus of the approach are safety-critical systems and its application is illustrated by an example.
08	Yoon and Choi [23]	The approach defines a test case prioritization strategy for sequencing test cases. Each test case is prioritized on the basis of the product of risk exposure value manually determined by domain experts and the correlation between test cases and risks determined by mutation analysis. The effectiveness is shown by comparing the number and severity of faults detected to the approach of Chen et al.
09	Zech [24]	The approach is model-based and derives a risk model from a system model and a vulnerability knowledge base. On this basis a misuse case model is derived and test code generated from this model is executed. The approach is intended to be applied for testing cloud systems.
10	Bai et al. [11]	The approach addresses risk-based testing of service-based systems taking the service semantics which is expressed by an OWL ontology into account. For estimating probability and impact dependencies in the ontology are considered. The approach considers the continuous adjustment of software and test case measurement as well as of rules for test case selection, prioritization and service evaluation. The approach is evaluated by comparing its cost and efficiency to random testing.

Table 1. *(continued)*

11	Felderer et al. [7]	The approach defines a generic risk-based test process containing the steps (1) risk identification, (2) test planning, (3) risk analysis, (4) test design as well as (5) evaluation. Steps (2) and (3) can be executed in parallel. For this test process a risk assessment model based on the Factor-Criteria-Metrics model is defined. The metrics in this model can be determined automatically, semi-automatically or manually. The approach is illustrated by an example.
12	Wendland et al. [2]	The approach is model-based. It formalizes requirements as integrated behavior trees and augments the integrated behavior tree with risk information. Then for each risk an appropriate test directive is identified, and finally both the risk-augmented integrated behavior tree and the test directive definition are passed into a test generator.
13	Felderer and Ramler [12,25]	The approach defines a process to stepwise introducing risk-based testing into an established test process. On this basis four stages of risk-based test integration are defined, i.e., (1) initial risk-based testing including design and execution of test cases on the basis of a risk assessment, (2) risk-based test results evaluation, (3) risk-based test planning, as well as (4) optimization of risk-based testing. The approach is evaluated in a case study.
14	Ray and Mohapatra [13]	The approach defines a risk analysis procedure to guide testing. It is based on sequence diagrams and state machines. First one estimates the risk for various states of a component within a scenario and then, the risk for the whole scenario is estimated. The key data needed for risk assessment are complexity and severity. For estimating complexity inter-component state-dependence graphs are introduced. The severity for a component within a scenario is decided based on three hazard techniques: Functional Failure Analysis, Software Failure Mode and Effect Analysis and Software Fault Tree Analysis. The efficiency of the approach is evaluated compared to another risk analysis approach.

3 Risk Assessment Framework

In this section we present a risk assessment framework for risk-based testing purposes. This framework is shown in Fig. 3. It contains a *risk assessment model* which configures the *risk-based test process*. The execution of the test process provides feedback to continuously refine and improve the risk assessment model. As mentioned in the previous section, the risk-based test process integrates risk assessment into the test process and uses risks to support all phases of the test process, i.e., test planning, design, implementation, execution, and evaluation. The framework is based on the risk-based test process which is configured by and provides feedback for the risk assessment model and explained as background in Section 2.

The risk assessment model and its elements therefore determine the overall risk-based test process and are the main component of our risk assessment framework for testing purposes. The risk assessment model defines the test scope, the risk identification method, a risk model and the tooling for risk assessment. In the following, we explain these elements in more detail illustrated by examples from the RBT approaches collected in Section 2.3. Each mentioned approach is referred to by its name and identifier. For the often cited approach of Amland [6] we discuss all aspects of risk assessment model definition.

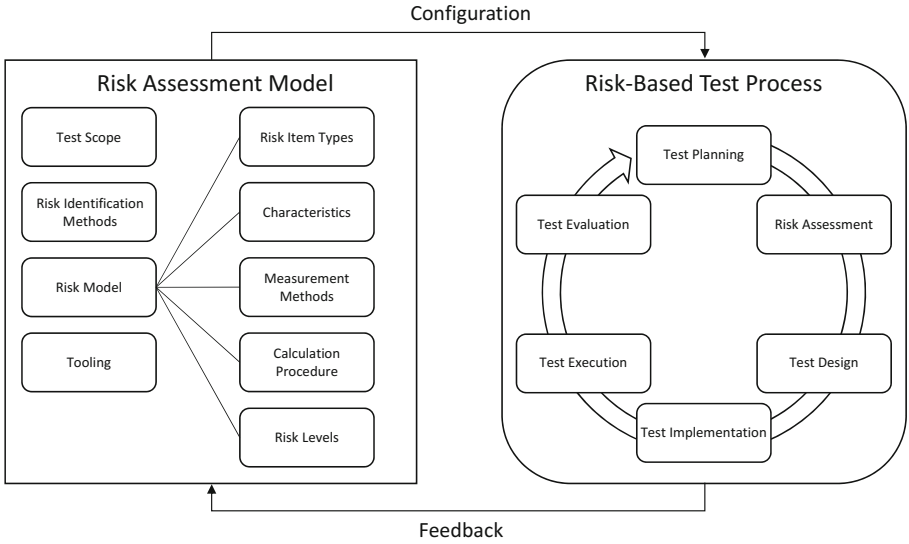

Fig. 2. Risk Assessment Framework

3.1 Test Scope

The test scope determines whether and how risk assessment is performed. It provides the overall testing context and typically considers the test object, test resources and test strategy. The test object defines the component or system to be tested and therefore influences the risk identification method and risk model. Limited test resources, i.e., personnel, time or budget, are typically the main driver for performing risk-based testing. Thus, the available resources determine whether a risk-based testing approach is required or not. The test strategy is a high-level description of the test levels to be performed and the testing within those levels determining risk-based testing as well.

All listed RBT approaches implicitly presume that prerequisites for the application of risk-based testing like limited resources are fulfilled and that the test objects are defined. Amland (01), for instance, states, "As for all projects, time and resources were limited." (cf. [6], page 2). Furthermore, all approaches consider system or integration testing for components, services or complete systems.

3.2 Risk Identification Methods

Risk identification methods are techniques to identify risks items. There are several risk identification methods such as brainstorming, risk checklists, and failure history available [3,26] which can be applied and tailored to a specific RBT context to define a risk model. Different roles of the software engineering process like product managers, business analysts, software architects, testers or developers as well as different artifact like requirements specifications, documentation,

defect databases or source code can be considered in specific risk identification methods.

Most RBT approaches do not explicitly mention the underlying risk identification methods but only present the resulting risk model and its application. Amland (01), for instance, explains the step 'identification of risk indicators' in which risk criteria are selected in a group meeting to guarantee that the used criteria are meaningful to those participating in the process of assessment. Souza et al. (05) use a taxonomy-based questionnaire answered by the project members, followed by a brainstorming meeting to identify technical risks.

3.3 Risk Model

The risk model is based on the concept of risk (see Section 2.1 the core artifact of the risk assessment model. It determines how the risk assessment is conducted in the risk-based test process. As Fig. 3 shows, the risk model consists of risk item types, characteristics, measurement methods, a calculation procedure, as well as risk levels which together define how risks are assessed. In the following, we explain these parts of the risk model in more detail.

Risk Item Types. The risk items type determines the risk items, i.e., the elements to which risk exposure values and tests are assigned, and their representation. For instance, Amland (01) assigns risks to system functions like 'Close Account' collected in a list. Furthermore, Chen et al. (02) assigns risks to test cases represented as path through an activity diagram, Zimmermann et al. (06) to critical functions represented as transitions in Markov chains, Kloos et al. (07) to safety risks represented as fault trees and state machines, Bai et al. (10) to web services represented as semantic models in OWL-S, and Wendland et al. (12) to requirements represented in behavior trees. Yoon and Choi (08) consider abstract sources of risk and assign the number of faults lying within the scope of a given risk and the test cases covering these faults to it. Felderer and Ramler (13) discuss different viewpoints for risk assessment, i.e., functional, architectural as well as development viewpoint, and conclude that the architectural viewpoint based on the components provides the most comprehensive structure in the considered project. Finally, Ray and Mohapatra (14) assigns risks to components represented as state machines, state dependence graphs and fault trees.

Characteristics. Characteristics define factors and their relationship to determine the risk. As such they define the applied risk concept. Typically, at least factors for probability and impact are considered which may be further refined based on the Factor-Criteria-Metrics model [27] defining a tree of factors with concrete measurable metrics at its leaves. Sometimes there are no defined characteristics and the risk is measured directly.

Amland (01) defines the factors probability and cost. For probability the criteria 'new functionality', 'design quality', 'size', and 'complexity' are distinguished, and for cost the criteria 'cost for customer' and 'fault occurrence'. Furthermore,

Redmill (03) distinguishes between single-factor analysis based on impact or probability as well as two-factor analysis based on both factors. Stallbaum et al. (04) as well as Felderer et al. (11) define characteristics explicitly on the Factor-Criteria-Metrics model. Both distinguish the factors probability and impact and state that probability is mainly determined by technical criteria and metrics of software development activities but impact mainly by business criteria and metrics of domain analysis. Finally, Ray and Mohaptra (14) take the factors severity and complexity of components into account.

Measurement Methods. A measurement method defines how values are directly assigned to factors. The measurement can be performed manually or automatically. If the measurement is performed manually, the role performing the estimation and the procedure how the estimation is performed (e.g., a consensus meeting if several persons perform the estimation) have to be defined. If it is performed automatically, the measurement object and the measurement tool, e.g., a static analysis tool, have to be defined. Each measured factor requires a scale of arbitrary range for its assigned value. For manual measurement, a Likert scale is typically used where selection items are assigned to values. Automatically measured values are used directly or they are mapped to another scale.

Amland (01) applies a three-point Likert scale with low (1), medium (2) and high (3) or all criteria. The numeric values for the probability criteria were determined in a consensus meeting where the roles developer, designer, product specialist, quality manage, development manager, project manager, sales manager and corporate management were present. The cost criteria were determined by the customer and the supplier, respectively. Measurement methods similar to Amland (01) are applied in many industrial settings [12]. More advanced approaches are presented by Zech (09) who defines an automated approach to measure security risk values based on a system model and a vulnerability knowledge base.

Calculation Procedure. The calculation procedure defines how risk exposure values are calculated on the basis of other risk exposure values, characteristics, measured values and testing information. It determines how to aggregate values, i.e., which aggregation function to apply, how to scale values and how to weight different factors.

Amland (01) weights the values for the probability factors and computes the probability value as their weighted sum. The cost value is the average of the two cost factor values. The risk exposure value is then the product of the probability and cost value. Stallbaum et al. (04) determines the risk exposure value for each action (modeled in an activity diagram) by the product of the probability that an entity contains a fault and he total damage caused by this fault which are both measured on a five-level scale from 1 to 5. The risk exposure value of a test case, which is a sequence of actions, is then the sum of the actions' risk exposure values.

Risk Levels. Risk levels indicate the criticality of risk items and serve the purpose to compare risk items as well as to configure testing activities. Risk

levels can be expressed either qualitatively or quantitatively [2]. For instance, numeric risk exposure values can be directly used as quantitative risk levels. Although, there is no restriction on the number of risk levels, a frequently used qualitatively scale for risk levels is low, medium, and high. Risk levels are often defined by the two dimensions probability and impact (each with levels low, medium and high) which are visualized in a risk matrix. Different areas in the risk matrix may then mapped to risk levels low, medium and high. If risk levels are measured qualitatively, factors are either directly measured qualitatively, e.g., with levels low, medium, and high, or their numeric values are mapped to these levels.

If risk levels are defined explicitly in the listed approaches, then qualitative two-dimensional risk levels are applied. Amland (01) and Wendland (12) map risk items to a 3x3 risk matrix with the two dimensions probability and impact with levels low, medium and high. The three cells at the lower left corner have low risk, the three cells at the upper right have high risk, and the remaining three cells have medium risk. Felderer et al. (11) apply a 2x2 risk matrix to determine the risk level and map their risk items with probability and impact values (each measured on a scale from 0 to 9) into this matrix. In the 2x2 risk matrix the lower left cell shows low risk, the upper right cell high risk, and the remaining two cells medium risk.

3.4 Tooling

If the risk assessment is not done ad-hoc, it requires tool support to be performed efficiently. The tooling may include printed forms as well as software tool support to perform the computations in an automatic way. Software tools supporting risk assessment for testing may be spreadsheets, specific risk assessment or management tools [28], or test or project management tools. The tooling is often fixed already before the risk model is defined, e.g., because a specific test management tool has to be used, and influences the definition of the risk model.

Amland (01) uses a spreadsheet for risk assessment. Souza et al. (05) use a specific risk assessment tool called RBTTool. Felderer et al. (11) use forms to conduct a risk assessment workshop. Finally, in Section 4 we integrate risk assessment into a project management tool.

Figure 3 summarizes which risk-based testing approach (RBT Approach) explicitly addresses which aspect of the risk assessment model. We skip the test scope as all listed RBT approaches implicitly presume that the prerequisites for the application of risk-based testing are fulfilled and defined.

The risk identification method is covered explicitly only be approaches 01 and 05. The risk item type is explicitly addressed by all RBT approaches. The remaining aspects are covered by most approaches. Only approach 01, i.e., Amland, covers all listed aspects explicitly.

	RBT Approach													
	01	02	03	04	05	06	07	08	09	10	11	12	13	14
Risk Identification Methods	x				x									
Risk Item Type	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Characteristics	x	x	x	x	x	x	x		x	x	x	x		x
Calculation Procedure	x	x	x	x			x	x	x	x	x	x		x
Measurement Methods	x	x	x	x	x	x			x	x	x		x	x
Risk Level	x	x	x	x	x	x		x	x	x	x	x	x	x
Tooling	x	x		x	x	x			x		x	x		x

Fig. 3. Elements of Risk Assessment Framework Covered by RBT Approaches

4 Application of Risk Assessment Model in an Industrial Test Process

In this section, we show how the risk assessment model is applied in the test process of a company in the telecommunication domain. The company follows a structured development and test process on the basis of a clearly-defined generic system and test model shown in Fig. 4. Further details on the company and its test process can be found in [29].

In this model, so called features are the central concept to plan and control implementation and testing. A *feature* has a concise and complete description of its functionality, along with non-functional aspects like performance or security. Features are on the one hand assigned to requirements and on the other hand to components. A *requirement* describes a certain functional or non-functional property of the system and is implemented by a set of features. A *component* is an installable artifact that provides the functionality of several features. Components are defined hierarchically in a tree. The root component represents the *system* and the leaves are *units*. As features are the tested artifacts, test cases are assigned to them. Differing from components, *testable objects* are executable units composed of one or more component and a test environment. Testable objects are assigned to the system or a component and have an attached test plan. A *test plan* contains test cases grouped either by features or components. Each test case contains a description, test steps and expected results.

Development and testing follow the V-model. First, a customer solution manager collects the requirements in a user requirements specification. Then, the features are defined and the system architecture is derived by a technical solution manager. The features are assigned to requirements in the technical requirements specification. The system design is then further refined to concrete components with assigned units, which are implemented and tested by a developer. As soon as feature definitions are available, test planning is started. First, testable objects are defined and a test plan, which is based on formerly defined requirements acceptance criteria. It contains test cases grouped either by features or components

and has a test end criterion. In the test design phase, executable test cases are defined by testers according to the test plan. Test cases are also adopted from existing components or features. New or changed test cases are reviewed and corrected if necessary. After the respective testable object (including its test environment) has been implemented, the test cases are executed. Each *test run* contains a *test result* for each of its executed test cases. Depending on the test results, a problem ticket is created. As soon as the test end criterion is reached, a test report is provided.

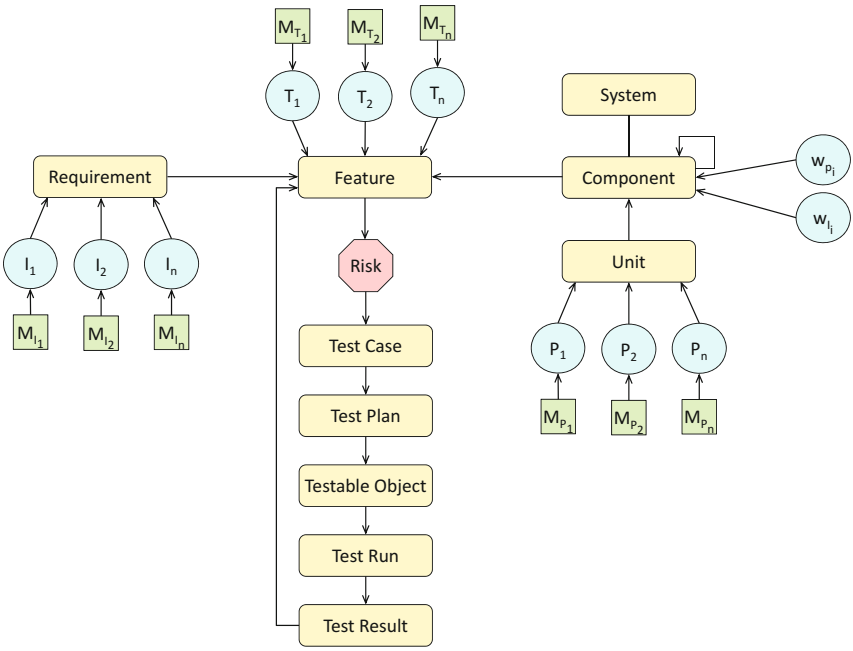

Fig. 4. System, Test and Risk Assessment Artifacts in Development and Test Process [29]

Test Scope. The expected benefits of risk-based testing are mainly decision support on resource allocation. The time resources for testing are limited as solutions have to be provided at fixed dates. Therefore, test cases should be prioritized for execution based on their risk level to mitigate highest risks in the limited test window. A test end criterion which considers the risk levels of features should terminate testing.

Risk Identification Methods. Due to the established development and test process, features were more or less already fixed as risk items. Risk identification put its focus on the identification of factors determining the risk assigned to features. For this purpose a list of factors is prepared from which suitable factors are selected by test managers in a separate workshop.

Risk Item Types. As test cases are linked to features, they are used as risk items to which risk exposure values are assigned as well. Features are traceable to requirements and components and therefore allow integrating a technical view on risk based on the components as well as a business view based on the requirements.

Characteristics. Risk is defined on the basis of the Factor-Criteria-Metrics model [27]. The definition considers the factors probability P and impact I as well as the additional factor time T. Probability reflects the technical view, impact the business view and time the system evolution. The factor values are determined by weighted criteria. The probability factor is composed of the weighted technical criteria code complexity, data complexity, functional complexity, visibility and third-party software. The impact factor is composed of the user and business-oriented weighted criteria usage, availability, importance and performance. The time factor is composed of the criteria bug tracking, change history, new technologies and project progress. Each factor is determined by a specific metric.

Measurement Methods. Probability, impact and time are explicitly defined on the basis of several criteria. For each criterion metrics are defined to determine the value in an objective way. The metrics can be determined manually by a suitable stakeholder or even automatically. For instance, the importance is measured manually on a five-point Likert scale and the code complexity is measured automatically by the McCabe complexity [30]. For the automatic measurement of source code metrics the source code quality management tool Sonar [31] is applied.

Calculation Procedure. Due to the traceability between requirements, components and units via features, the probability criteria are measured for units, the impact factors for requirements, and the weights are assigned to components (from which only a few exist). Time criteria are directly assigned to features, for which the risk exposure values are calculated (see Fig. 4). The probability P and the impact I are evaluated by several weighted criteria. For a risk item a, the probability P is for instance determined by the formula $P(a) = (\sum_{j=0}^{m} p_j \cdot w_j) \div (\sum_{j=0}^{m} w_j)$, where p_j are values for probability criteria and w_j are weight values for the criteria. The range of the criteria values are natural numbers between 0 and 9, and of the weights real numbers between 0 and 1 (so the weight can be naturally interpreted as scaling factor). The time factor, which scales the probability, has a range between 0 and 1, and is the mean of the time criteria values. The risk exposure value of a feature can be calculated via the formula, $R = (P \cdot T) \times I$, where P denotes the aggregated probability factor, I the aggregated impact factor, and T the aggregated time factor which reduces the value of P over time. Figure 6 shows the computed risk coefficients for seven features based values for the time criteria bug tracking, change history, new technologies and project progress. For each feature, the mean of the time

criteria values is multiplied with the probability value and then combined with the impact value. For instance, all time criteria values of Feature 003 in Fig. 6 are 1. Therefore, the time factor T, i.e., the mean of the time criteria values, is 1 as well. With a probability factor P of 5 and an impact factor I of 6.75, the resulting risk coefficient R is $(5 \cdot 1) \times 6.75$, which corresponds to the value 33.75.

Risk Levels. The scaled probability and impact value defining the risk value are mapped to a 2x2 risk matrix to determine the risk level. Risk items, i.e., features with assigned risk values, mapped to the lower left cell have low risk, to the upper right cell high risk, and to the remaining two cells medium risk. Figure 5 shows the applied risk matrix. In this risk matrix, Feature 003 with risk coefficient $(5 \cdot 1) \times 6.75$ is of high risk.

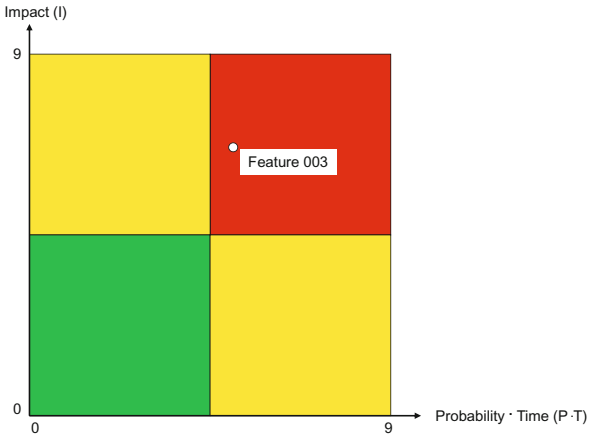

Fig. 5. Risk Matrix in Industrial Case

Tooling Risk assessment as well as the overall risk-based test process with all its artifacts and process steps is supported in the project management tool in-Step [32] which is already established in the company. Figure 6 shows a screenshot of the specifically developed risk assessment view of in-Step where the measures for criteria are entered and processed to calculate risk exposure values. In addition, for the automatic measurement of source code metrics the source code quality management tool Sonar [31] is used.

Risk-Based Test Process In the test process risks are explicitly considered in the test planning, test design and test execution phase. In the test planning phase, first testable objects, which are executable units composed of one or more components and a test environment, are defined. Then a test plan is created on the basis of formally defined test end criteria, features with attached risk exposure values and components. A typical test end criterion defines that all features with

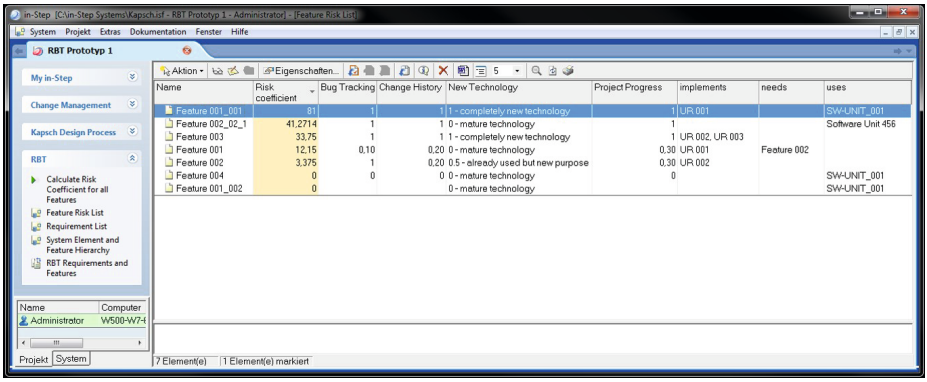

Fig. 6. Risk Assessment in Project Management Tool in-Step [29]

high risk have to be tested, features with medium risk are optional candidates to be tested in order to reach the required test coverage, and features with low risk are only tested if all others have been tested and resources are available. In the test design phase, executable test cases are defined by testers according to the test plan and get assigned the risk exposure value of the feature they are designed for. Already existing test cases which are applicable are selected from similar previous components or features. New or changed test cases are reviewed and corrected if necessary. After the respective testable object (including its test environment) is available for the test, the test cases are executed ordered by their risk level and risk exposure values (inherited from the assigned features). Each test run contains a test result for each of its executed test cases. Depending on the test results, a problem ticket is created. As soon as the test end criterion is reached, a test report is created. But the test report itself does not explicitly take risk information into account.

5 Summary and Future Work

In this paper we presented a risk assessment framework for testing purposes. The framework is based on the risk-based test process which is configured by and provides feedback for a risk assessment model. This model is the main component of our framework and defines the test scope, the risk identification method, a risk model as well as the tooling for risk assessment. The risk assessment framework is derived on the basis of best practices extracted from published risk-based testing approaches and applied to an industrial test process where it guides the definition and application of the RBT approach.

In future, we intend to provide a catalog with concrete guidelines on how to configure the risk assessment model to additionally support practitioners. In addition, we will perform empirical case studies to further evaluate and improve the risk assessment model.

Acknowledgment. This research was partially funded by the research projects MOBSTECO (FWF P 26194-N15) and QE LaB - Living Models for Open Systems (FFG 822740).

References

1. Felderer, M., Schieferdecker, I.: A taxonomy of risk-based testing. STTT (2014)
2. Wendland, M.F., Kranz, M., Schieferdecker, I.: A systematic approach to risk-based testing using risk-annotated requirements models. In: ICSEA 2012, The Seventh International Conference on Software Engineering Advances, pp. 636–642 (2012)
3. ISTQB: Standard glossary of terms used in software testing, version 2.2. Technical report, ISTQB (2012)
4. Alam, M.M., Khan, A.I.: Risk-based testing techniques: A perspective study. International Journal of Computer Applications 65(1) (2013)
5. ISO: ISO/IEC 29119 Software Testing, Draft (2013)
6. Amland, S.: Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study. Journal of Systems and Software 53(3), 287–295 (2000)
7. Felderer, M., Haisjackl, C., Breu, R., Motz, J.: Integrating manual and automatic risk assessment for risk-based testing. In: Biffl, S., Winkler, D., Bergsmann, J. (eds.) SWQD 2012. LNBIP, vol. 94, pp. 159–180. Springer, Heidelberg (2012)
8. Merriam-Webster: Merriam-Webster Online Dictionary (2009), `http://www.merriam-webster.com/dictionary/risk` (accessed: April 04, 2013)
9. Standards Australia/New Zealand: Risk Management AS/NZS 4360:2004 (2004)
10. McCall, J., Richards, P., Walters, G.: Factors in software quality. Technical report, NTIS, vol. 1, 2 and 3 (1997)
11. Bai, X., Kenett, R.S., Yu, W.: Risk assessment and adaptive group testing of semantic web services. International Journal of Software Engineering and Knowledge Engineering 22(05), 595–620 (2012)
12. Felderer, M., Ramler, R.: Integrating risk-based testing in industrial test processes. Software Quality Journal, 1–33 (2013) (online first)
13. Ray, M., Mohapatra, D.P.: Risk analysis: a guiding force in the improvement of testing. IET Software 7(1), 29–46 (2013)
14. Chen, Y., Probert, R.L., Sims, D.P.: Specification-based regression test selection with risk analysis. In: Proceedings of the 2002 Conference of the Centre for Advanced Studies on Collaborative Research, p. 1. IBM Press (2002)
15. Redmill, F.: Exploring risk-based testing and its implications. Software Testing, Verification and Reliability 14(1), 3–15 (2004)
16. Redmill, F.: Theory and practice of risk-based testing. Software Testing, Verification and Reliability 15(1), 3–20 (2005)
17. Stallbaum, H., Metzger, A.: Employing requirements metrics for automating early risk assessment. In: Proc. of MeReP 2007, Palma de Mallorca, Spain, pp. 1–12 (2007)
18. Stallbaum, H., Metzger, A., Pohl, K.: An automated technique for risk-based test case generation and prioritization. In: Proceedings of the 3rd International Workshop on Automation of Software Test, pp. 67–70. ACM (2008)
19. Souza, E., Gusmao, C., Alves, K., Venancio, J., Melo, R.: Measurement and control for risk-based test cases and activities. In: 10th Latin American Test Workshop, pp. 1–6. IEEE (2009)

20. Souza, E., Gusmão, C., Venâncio, J.: Risk-based testing: A case study. In: 2010 Seventh International Conference on Information Technology: New Generations (ITNG), pp. 1032–1037. IEEE (2010)
21. Zimmermann, F., Eschbach, R., Kloos, J., Bauer, T., et al.: 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 2009 (2009)
22. Kloos, J., Hussain, T., Eschbach, R.: Risk-based testing of safety-critical embedded systems driven by fault tree analysis. In: 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 26–33. IEEE (2011)
23. Yoon, H., Choi, B.: A test case prioritization based on degree of risk exposure and its empirical study. International Journal of Software Engineering and Knowledge Engineering 21(02), 191–209 (2011)
24. Zech, P.: Risk-based security testing in cloud computing environments. In: 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation (ICST), pp. 411–414. IEEE (2011)
25. Felderer, M., Ramler, R.: Experiences and challenges of introducing risk-based testing in an industrial project. In: Winkler, D., Biffl, S., Bergsmann, J. (eds.) SWQD 2013. LNBIP, vol. 133, pp. 10–29. Springer, Heidelberg (2013)
26. Pandian, C.R.: Applied software risk management: a guide for software project managers. CRC Press (2006)
27. Cavano, J., McCall, J.: A framework for the measurement of software quality. ACM SIGMETRICS Performance Evaluation Review 7(3-4), 133–139 (1978)
28. Haisjackl, C., Felderer, M., Breu, R.: Riscal–a risk estimation tool for software engineering purposes. In: 2013 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 292–299. IEEE (2013)
29. Felderer, M., Ramler, R.: A multiple case study on risk-based testing in industry. STTT (2014)
30. McCabe, T.: A complexity measure. IEEE Transactions on Software Engineering, 308–320 (1976)
31. SonarSource: Sonar (2013), `http://www.sonarsource.org/` (accessed: March 12, 2013)
32. microtool: in-Step (2013), `http://www.microtool.de/inStep` (accessed: November 30, 2013)