ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/271737308

# A taxonomy of risk-based testing

**Article** *in* International Journal on Software Tools for Technology Transfer · October 2014

DOI: 10.1007/s10009-014-0332-3

CITATIONS
109

READS
1,917

**2 authors:**

Michael Felderer
German Aerospace Center (DLR)
**398** PUBLICATIONS **7,085** CITATIONS

SEE PROFILE

Ina Kathrin Schieferdecker
Technische Universität Berlin
**271** PUBLICATIONS **2,716** CITATIONS

SEE PROFILE

All content following this page was uploaded by Michael Felderer on 06 February 2015.

The user has requested enhancement of the downloaded file.

IPR2025-01397
CrowdStrike EX1035 Page 1

Int J Softw Tools Technol Transfer
DOI 10.1007/s10009-014-0332-3

INTRODUCTION

# A taxonomy of risk-based testing

**Michael Felderer · Ina Schieferdecker**

© Springer-Verlag Berlin Heidelberg 2014

**Abstract** Software testing has often to be done under severe pressure due to limited resources and a challenging time schedule facing the demand to assure the fulfillment of the software requirements. In addition, testing should unveil those software defects that harm the mission-critical functions of the software. Risk-based testing uses risk (re-)assessments to steer all phases of the test process to optimize testing efforts and limit risks of the software-based system. Due to its importance and high practical relevance, several risk-based testing approaches were proposed in academia and industry. This paper presents a taxonomy of risk-based testing providing a framework to understand, categorize, assess, and compare risk-based testing approaches to support their selection and tailoring for specific purposes. The taxonomy is aligned with the consideration of risks in all phases of the test process and consists of the top-level classes risk drivers, risk assessment, and risk-based test process. The taxonomy of risk-based testing has been developed by analyzing the work presented in available publications on risk-based testing. Afterwards, it has been applied to the work on risk-based testing presented in this special section of the International Journal on Software Tools for Technology Transfer.

**Keywords** Risk-based testing · Risk management · Risk analysis · Software testing · Classification · Taxonomy

M. Felderer (✉)
University of Innsbruck, Innsbruck, Austria
e-mail: michael.felderer@uibk.ac.at

I. Schieferdecker
Fraunhofer Institute FOKUS and Freie Universität Berlin,
Berlin, Germany
e-mail: ina.schieferdecker@fokus.fraunhofer.de

## 1 Introduction

Testing of safety-critical, security-critical, or business-critical software faces the problem of determining those tests that assure the essential properties of the software and have the ability to unveil those software defects that harm the mission-critical functions of the software. However, also for normal, less critical software a comparable problem exists: Usually testing has to be done under severe pressure due to limited resources and tight time constraints with the consequence that testing efforts have to be focused.

Both decision problems can adequately be addressed by risk-based testing approaches which consider risks of the software product as the guiding factor to steer all phases of the test process, i.e., test planning, design, implementation, execution, and evaluation [15,18,38]. Risk-based testing is a pragmatic, in industry, widely used approach to address the problem of tests for mission-critical software or to cope with ever limited testing resources. Risk-based testing uses the straightforward idea to focus test activities on those scenarios that trigger the most critical situations for a software system [47].

Due to its importance and high practical relevance, several risk-based testing approaches were proposed in academia, e.g., [5,9,12,15,41,47,50,52,53], and industry, e.g., [1,3,18,35,37,46]. Recently, the international standard ISO/IEC/IEEE 29119 Software Testing [25] on testing techniques, processes, and documentation even explicitly considers risks as an integral part of the test planning process. Because of the growing number of available risk-based testing approaches and its increasing dissemination in industrial test processes [13], support to categorize, assess, compare, and select risk-based testing approaches is required.

In this paper, we present a taxonomy of risk-based testing providing a framework to understand, categorize, assess,

Published online: 23 July 2014



IPR2025-01397
CrowdStrike EX1035 Page 2

and compare risk-based testing approaches to support their selection and tailoring for specific purposes. In general, a *taxonomy* or classification (scheme) defines a hierarchy of classes (also referred to as categories, dimensions, criteria, or characteristics) to categorize things or concepts. It describes a tree structure whose leaves define concrete values to characterize instances. The proposed taxonomy is aligned with the consideration of risks in all phases of the test process and consists of the top-level classes *risk drivers* (with subclasses functionality, safety, and security), *risk assessment* (with subclasses risk item type, factors, estimation, and degree of automation), and *risk-based test process* (with subclasses risk-based test planning, design, implementation, execution, and evaluation). The taxonomy of risk-based testing has been developed by analyzing the work presented in [1,3,5,9,12,14,15,21,29,34–37,39–42,46,47,50–53]. Afterwards, it has been applied to the work on risk-based testing [7,10,16,32] presented in this special section of the International Journal on Software Tools for Technology Transfer (STTT, see Sect. 4).

The remainder of this paper is structured as follows. Section 2 presents basic concepts of risk-based testing. Section 3 introduces the taxonomy of risk-based testing. Section 4 presents the articles of the STTT Special Section on Risk-Based Testing and discusses them in the context of the taxonomy. Finally, Sect. 5 concludes this paper.

## 2 Basic concepts of risk-based testing

*Testing* is the evaluation of software by observing its execution [2]. The executed software-based system is called *system under test* (SUT). *Risk-based testing* (RBT) is a testing approach which considers risks of the software product as the guiding factor to support decisions in all phases of the test process [15,18,38]. A *risk* is a factor that could result in future negative consequences and is usually expressed by its likelihood and impact [26]. In software testing, the *likelihood* is typically determined by the probability that a failure assigned to a risk occurs, and the *impact* is determined by the cost or severity of a failure if it occurs in operation. The resulting *risk value* or *risk exposure* is assigned to a *risk item*. In the context of testing, a risk item is anything of value (i.e., an asset) under test, for instance, a requirement, a component, or a fault.

RBT is a testing-based approach to risk management which can only deliver its full potential if a test process is in place and if risk assessment is integrated appropriately into it. A *test process* comprises the core activities: test planning, test design, test implementation, test execution, and test evaluation [26]. In the following, we explain the particular activities and associated concepts in more detail.

According to [23] and [26], *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. It identifies, amongst others, objectives, the features to be tested, the test design techniques, and exit criteria to be used and the rationale of their choice. *Test objectives* are reason or purpose for designing and executing a test. The reason is either to check the functional behavior of the system or its nonfunctional properties. *Functional testing* is concerned with assessing the functional behavior of an SUT, whereas *nonfunctional testing* aims at assessing nonfunctional requirements such as security, safety, reliability, or performance. The scope of the features to be tested can be components, integration, or system. At the scope of *component testing* (also referred to as unit testing), the smallest testable component, e.g., a class, is tested in isolation. *Integration testing* combines components with each other and tests those as a subsystem, that is, not yet a complete system. In *system testing*, the complete system, including all subsystems, is tested. *Regression testing* is the selective retesting of a system or its components to verify that modifications have not caused unintended effects and that the system or the components still comply with the specified requirements [33]. *Exit criteria* are conditions for permitting a process to be officially completed. They are used to report against and to plan when to stop testing. Coverage criteria aligned with the tested feature types and the applied test design techniques are typical exit criteria. Once the test plan has been established, test control begins. It is an ongoing activity in which the actual progress is compared against the plan which often results in concrete measures.

During the *test design* phase, the general testing objectives defined in the test plan are transformed into tangible test conditions and abstract test cases. *Test implementation* comprises tasks to make the abstract test cases executable. This includes tasks like preparing test harnesses and test data, providing logging support, or writing test scripts which are necessary to enable the automated execution of test cases. In the *test execution* phase, the test cases are then executed and all relevant details of the execution are logged and monitored. Finally, in the *test evaluation* phase the exit criteria are evaluated and the logged test results are summarized in a test report.

*Risk management* comprises the core activities: *risk identification*, *risk analysis*, *risk treatment*, and *risk monitoring* [44]. In the risk identification phase, risk items are identified. In the risk analysis phase, the likelihood and impact of risk items and, hence, the risk exposure is estimated. Based on the risk exposure values, the risk items may be prioritized and assigned to risk levels defining a risk classification. In the risk treatment phase, the actions for obtaining a satisfactory situation are determined and implemented. In the risk monitoring phase, the risks are tracked over time and their



IPR2025-01397
CrowdStrike EX1035 Page 3

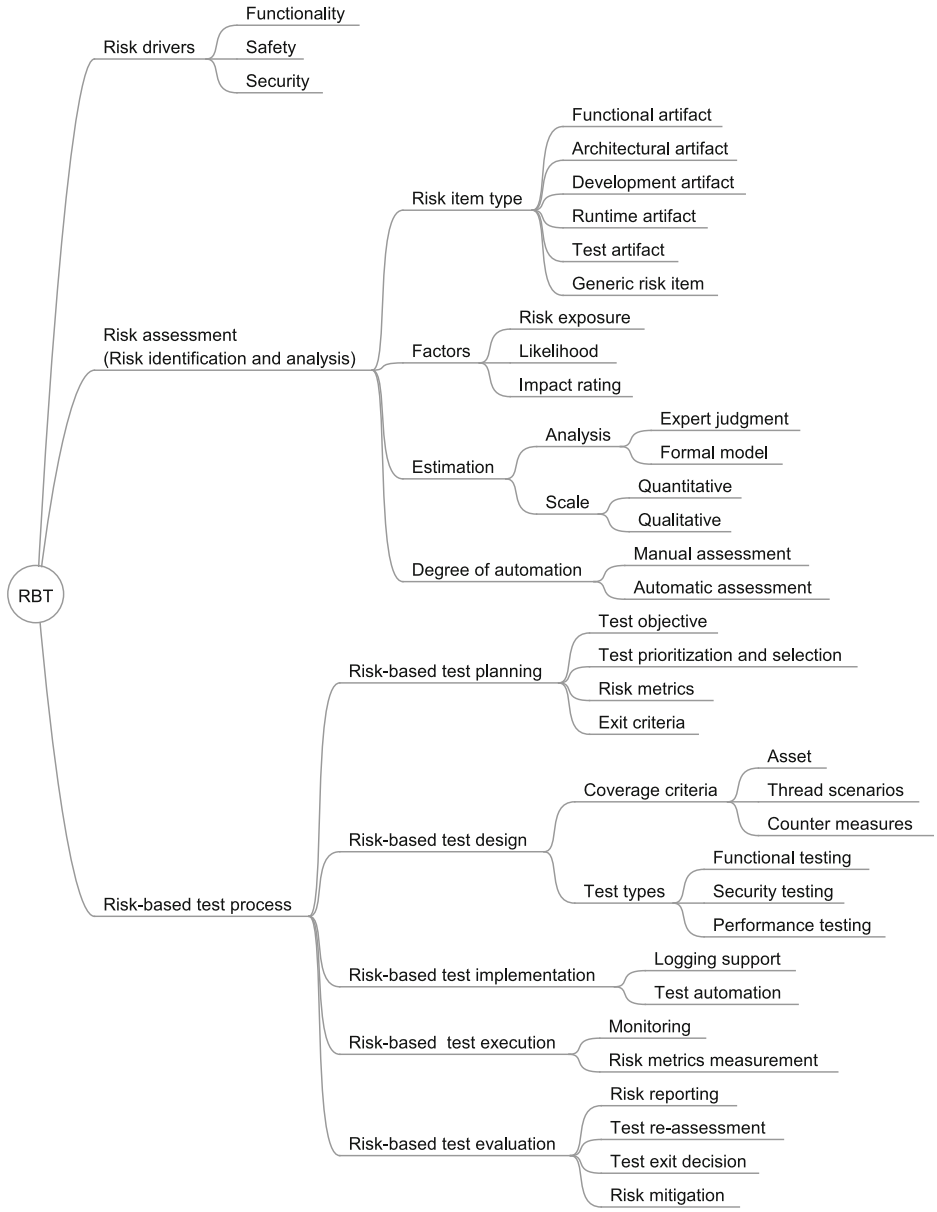


**Fig. 1** Risk-based testing taxonomy

status is reported. In addition, the effect of the implemented actions is determined. The activities risk identification and risk analysis are often collectively referred to as *risk assessment*, while the activities risk treatment and risk monitoring are referred to as *risk control*.

## 3 Taxonomy of risk-based testing

The taxonomy of risk-based testing is shown in Fig. 1. It contains the top-level classes *risk drivers*, *risk assessment* as well as *risk-based test process* and is aligned with the consideration of risks in all phases of the test process. In this section, we explain these classes, their subclasses, and concrete values for each class of the risk-based testing taxonomy in depth.

### 3.1 Risk drivers

As stated in [24], risks result from hazards. Hazards in software-based systems stem from software vulnerabilities and from defects in software functionalities, which are critical to safety-related aspects or to the business cases of the software. One needs to test that a software-based system is

- Reliable, i.e., able to deliver services as specified.
- Available, i.e., able to deliver services when requested.



IPR2025-01397
CrowdStrike EX1035 Page 4

- Safe, i.e., able to operate without harmful states.
- Secure, i.e., able to remain protected against accidental or deliberate attacks.
- Resilient, i.e., able to recover timely from unexpected events.

Since the respective test methods, i.e., functional testing, security testing, and performance testing differ, the considered risk drivers determine which testing is appropriate and has to be chosen. Therefore, the taxonomy begins with the *risk drivers* to be a first differentiating element of risk-based testing approaches. We consider *functionality*, *security*, and *safety* to be the dominant risk drivers for software. They together form the reliability, availability, safety, security, and resilience of a software-based system and hence constitute the options for the risk drivers in the RBT taxonomy.

## 3.2 Risk assessment

The second differentiating element of RBT approaches is the way risks are being determined. According to [26], *risk assessment* is the process of *identifying* and subsequently *analyzing* the identified risk to determine its level of risk, typically by assigning likelihood and impact ratings. Risk assessment itself has multiple aspects, so that one needs to differentiate further into the *risk item type*, to which the risk relate to, the *factors* influencing risks, the *risk estimation* technique used to estimate and/or evaluate the risk, and the *degree of automation* for risk assessment.

### *3.2.1 Risk item type*

The risk item type determines the elements to which risk exposures and tests are assigned [14]. Risk items can be of type *generic risk*, i.e., risk items independent of a particular artifact like security risks or specific faults, *test case* [50], i.e., directly test cases themselves as in regression testing scenarios, *runtime artifact* like deployed services, *functional artifact* like requirements or features, *architectural artifact* like component, or *development artifact* like source code file. The risk item type is determined by the test level. For instance, functional or architectural artifacts are often used for system testing, and generic risks for security testing. In addition, we use the term *artifact* to openly refer to other risk items used in requirements capturing, design, development, testing, deployment, and/or operation and maintenance, which all might relate to the identified risks.

### *3.2.2 Factors*

The risk factors quantify identified risks [5]: *Risk exposure* is the quantified potential for loss. It is calculated by the likelihood of risk occurrence multiplied by the potential loss, also called the impact. The risk exposure considers typically aspects like liability issues, property loss or damage, and product demand shifts. RBT approaches might also consider the specific aspect of *likelihood* of occurrence, e.g., for test prioritization or selection or the specific aspect of *impact rating* to determine test efforts needed to analyze the countermeasures in the software.

### *3.2.3 Estimation*

The estimation technique determines how the risk exposure is actually estimated and can be *expert judgment* or *formal model* [27]. The essential difference between formal-model-based and expert-judgment-based effort estimation is the quantification step—that is, the final step that transforms the input into the risk estimate. Formal risk estimation models are based on a mechanical quantification step such as a formula or a test model. On the other hand, judgment-based estimation methods are based on a judgment-based quantification step—for example, what the expert believes is most risky. Judgment-based estimation processes range from pure gut feelings to structured, historical data including failure history and checklist-based estimation processes.

In addition, any risk estimation uses a scale to determine the risk "level". This risk scale can be *quantitative* or *qualitative*. Quantitative risk values are numeric and allow computations, qualitative risk values can only be sorted and compared. An often used qualitative scale for risk levels is low, medium, and high [47].

### *3.2.4 Degree of automation*

Risk assessment can be supported by automated methods and tools. For example, risk-oriented metrics can be measured *manually* or *automatically*. The manual measurement is often supported by strict guidelines and the automatic measurement is often performed via static analysis tools. Other examples for automated risk assessment include the derivation of risk exposures from formal risk models, see, for instance [17].

## 3.3 Risk-based test process

Based on the risks being determined and characterized, RBT follows the fundamental test process [26] or variations thereof. All activities and phases in a test process are impacted by the risk perspective taken in RBT. RBT specifics in test processes are outlined in the following subsections.

### *3.3.1 Risk-based test planning*

*Test planning* establishes or updates the scope, approach, resources, and schedule of intended test activities. Amongst



IPR2025-01397
CrowdStrike EX1035 Page 5

others, *test objectives*, *test prioritization and selection*, *risk metrics*, and *exit criteria*, which impact risk-based testing [36], are determined.

*Test objectives* RBT requires focusing the testing activities and efforts based on the risk assessment of the particular product or of the project, in which it is developed. In simple words: if there is high risk, then there will be serious testing. If there is no risk, then there will be rather little testing. For example, products with high complexity, new technologies, many changes, many defects found earlier, developed by personnel with less experiences or lower qualification, or developed along new or renewed development processes may have a higher probability to fail and need to be tested more thoroughly. The reason to design or execute a test, i.e., a *test objective*, can be related to the risk item to be tested, to the thread scenarios of a risk item, or to the counter measures established to secure that risk item, see also Sect. 3.3.2.

*Test prioritization and selection* To optimize the costs of testing and/or the quality and fault detection capability of testing, techniques for prioritizing, selecting, and minimizing tests as well as combinations thereof have been developed and are widely in use [49]. In the ranges of intolerable risk and "As Low As Reasonably Practicable (ALARP)"[1] risks, these techniques are used to identify tests for the risk-related test objectives determined before. For example, design-based approaches for test selection [6] and coverage-based approaches [1] for test prioritization are wellsuited for RBT.

*Risk metrics* Metrics are used to quantify different aspects in testing such as the minimum level of testing, extra testing needed because of high number of faults found, the quality of the tests and the test process. They are used to manage the RBT process and optimize it with respect to time, efforts, and quality [1].

*Exit criteria* Typical exit criteria for testing that are used to report against and to plan when to stop testing, include all test ran successfully, all issues have been retested and signed off, or all acceptance criteria have been met. Specific RBT-related exit criteria [1] add criteria on the residual risk in the product and coverage-related criteria: all risk items, their threat scenarios, and/or counter measures being covered.

[1] The ALARP principle is typically used for safety-critical, but also for mission-critical systems. It says that the residual risk shall be as low as reasonably practical.

### 3.3.2 Risk-based test design

*Test design* is the process of transforming test objectives into test cases. This transformation is guided by the coverage criteria, which are used to quantitatively characterize the test cases and often used for exit criteria. Furthermore, the technique of transformation depend on the test types needed to realize a test objective. RBT uses coverage criteria specific to the risk artifacts and test types specific to the risk drivers on functionality, security, and safety.

*Coverage criteria* In RBT, the classical code-oriented and model-based coverage criteria like path coverage, condition-oriented coverage criteria like modified condition decision coverage, requirements-oriented coverage criteria like requirements or use case coverage are extended with coverage criteria to cover selected or all assets, thread scenarios, and counter measures [43]. While asset coverage rather belongs to requirements-oriented coverage [47], thread scenario, and counter measure coverage can be addressed by code-oriented, model-based, and/or condition-oriented coverage criteria [21].

*Test types* As reported by different computer emergency response teams such as GovCERT-UK, software defects continue to be a major, if not the main source of incidents caused by software-based systems. Therefore, functional testing is likewise a major test type in RBT to analyze reliability and safety aspects, see, e.g., [1]. In addition, security testing including penetration testing, fuzz testing, and/or randomized testing is key in RBT [51] to analyze security and resilience aspects. Furthermore, performance and scalability testing focusing on normal load, maximal load, and overload scenarios to analyze availability and resilience aspects, see, e.g., [1].

### 3.3.3 Risk-based test implementation

*Test implementation* comprises tasks like preparing test harnesses and test data, providing logging support, or writing automated test scripts to enable the automated execution of test cases [26]. Risk aspects are especially essential for providing *logging support* and for *test automation*.

*Logging support* Logging is the process of recording information about tests executed into a test log. Especially for risk-based testing, it is important to document the test progress via test logging [1]. This may require adaptations of the logging support to meet the special requirements of risk-based testing, for instance, on risk items.

*Test automation* Test automation is the use of special software (separate from the software under test) to control the execution of tests and the comparison of actual outcomes with pre-



IPR2025-01397
CrowdStrike EX1035 Page 6

dicted outcomes [22]. Experiences from test automation [20] show possible benefits like improved regression testing or a positive return on investment, but also caveats like high initial investments or difficulties in test maintenance. Risks may, therefore, be beneficial to guide decisions where and to what degree testing should be automated.

*3.3.4 Risk-based test execution*

*Test execution* is the process of running test cases. In this phase, risk-based testing is supported by *monitoring* and *risk metrics measurement*.

*Monitoring* Monitoring is run concurrently with a system under test and supervises, records, or analyzes the behavior of the running system [26,33]. Differing from software testing, which actively stimulates the system under test, monitoring only passively observes a running system. For risk-based testing purposes, monitoring enables additional complex analysis, e.g., of the internal state of a system for security testing, as well as tracking the project's progress toward resolving its risks and taking corrective action where appropriate.

*Risk metrics measurement* Risk metrics measurement determines risk metrics defined in the test planning phase. A measured risk metrics could be the number of observed critical failures for risk items where failure has high impact [11].

*3.3.5 Risk-based test evaluation*

*Test evaluation* comprises decisions on the basis of exit criteria and logged test results compiled in a test report. In this respect, risks are *mitigated* and may require a *re-assessment*. Furthermore, risks may guide *test exit decisions* and *reporting*.

*Risk reporting* Test reports are documents summarizing testing activities and results [26] that communicate risks and alternatives requiring a decision. They typically report progress of testing activities against a baseline (such as the original test plan) or test results against exit criteria. In *risk reporting*, assessed risks which are monitored during the test process, are explicitly reported in relation to other test artifacts. Risk reports can be descriptive summarizing relationships of the data or predictive using data and analytical techniques to determine the probable future risk. Typical descriptive risk reporting techniques are risk burn down charts which visualize the development of the overall risk per iteration, as well as traffic light reports providing a high level view on risks using colors: red for high risks, yellow for medium risks, and green for low risks. A typical predictive risk reporting technique is residual risk estimation, for instance, based on software reliability growth models [19].

*Risk re-assessment* The *re-assessment of risks* after test execution may be planned in the process or triggered by a comparison of test results against the assessed risks. This may reveal deviations between the assessed and the actual risk level and require a re-assessment to adjust them. Test results can explicitly be integrated into a formal risk analysis model [41] or just trigger the re-assessment in an informal way.

*Test exit decision* The *test exit decision* determines if and when to stop testing [14], but may also trigger further risk mitigation measures. This decision may be taken on the basis of a test report matching test results and exit criteria or ad hoc, for instance, solely on the basis of the observed test results.

*Risk mitigation* Risk mitigation covers efforts taken to reduce either the likelihood or impact of a risk [45]. In the context of risk-based testing, the assessed risks and their relationship to test results and exit criteria (which may be outlined in the test report), may trigger additional measures to reduce either the likelihood or impact of a risk to occur in the field. Such measures may be bug fixing, re-design of test cases, or re-execution of test cases.

## 4 Articles of this special section

This special section on risk-based testing comprises four articles. Two of them are primary studies on risk-based testing [7, 32] presenting novel risk-based testing approaches, and two are secondary studies on risk-based testing [10,16] empirically evaluating existing risk-based testing approaches. In this section, we provide an overview of each article and discuss each of them in context of the proposed risk-based testing taxonomy. For this purpose, primary studies on risk-based testing are classified according to the taxonomy and secondary studies are used to additionally check its adequacy.

The article "Risk-based testing via active continuous quality control" by Neubauer et al. [32] shows how Active Continuous Quality Control (ACQC) [48], which employs learning technology to automatically maintain test models during system evolution, can be extended by risk assessment to support risk-based (regression) testing. Key to this enhancement is the tailoring of ACQC's characteristic model extraction based on automata learning to prioritize critical aspects. Technically, so called risk analysts are provided with an abstract modeling level tailored to design the required learning alphabets that encompass data flow constraints reflecting risk profiles at the user level. The resulting alphabet models steer the ACQC process in a fashion that increases risk coverage while it at the same time reduces the regression testing effort. This is illustrated by the application of the approach to Springer's Online Conference Service in two system evo-



IPR2025-01397
CrowdStrike EX1035 Page 7

lution scenarios, i.e., system migration and functional extension.

According to the risk-based testing taxonomy, the risk driver of the presented approach to risk-based testing via ACQC is fault-prone functionality expressed by symbols which reflect single critical actions of the SUT from the user perspective and together form an alphabet. Risk assessment itself is not directly addressed in the article. But it is stated that, "a risk analyst uses his knowledge to select, parametrize (instantiate), and combine generic alphabet symbols to build tailored risk-based alphabet models", and further, that, these alphabet models "encompass data flow constraints reflecting risk profiles at the user level". From these statements, it can be concluded that risks are assigned to functional artifacts, i.e., user actions, probability and impact factors are not defined, and risk exposure is measured automatically on a qualitative scale based on expert judgment. The risk-based alphabet models steer the ACQC process and are considered for test planning and design. The approach determines the generated test model which influences test selection and prioritization in the test planning phase, as well as coverage of risk items, i.e., critical actions.

The article "Dynamic test planning: a study in an industrial context" by Carrozza et al. [7] presents a method to dynamically allocate testing resources to software components minimizing residual risks in terms of the estimated number of defects or estimated residual defect density. The method is based on software reliability growth models [19], used at component level to monitor the testing progress of each component. From these models, an estimate of the quality achievable for a component in relation to the testing effort devoted to it is obtained. By iteratively solving an optimization problem, the testing effort is consecutively directed toward the component contributing the most to reduce the number of residual defects or their density in the overall system. The method is implemented in a tool and applied to a real-world critical system in the homeland security domain aiming at the management of port, maritime and coastal surveillance. Results of the application show improved test process outcome measured in terms of detected defects compared to uniform or size-based allocation of testing resource given a predefined amount of testing resources.

With regard to the taxonomy, the presentation of the method in the article focuses on risk assessment to dynamically allocate testing resources, but its integration into the test process is only addressed on the side. The considered risk driver is again fault-prone functionality. Risk is assessed for architectural artifacts, i.e., software components, taking probability based on defects into account. Estimation is based on a formal analysis model using software reliability growth models to quantitatively measure risks on the basis of the estimated number of residual defects and their density. The measurement is automated by a tool presented in the article. The assessed risk of components is applied in the test planning phase as a metrics to distribute testing resources to components. During test execution, risk in terms of observed defects is monitored and taken into account to re-assess and report risks in the test evaluation phase.

The article "A multiple case study on risk-based testing in industry" by Felderer and Ramler [16] presents a case study on currently applied risk-based testing approaches grounded on three industry cases from different backgrounds, i.e., a test project in context of the extension of a large information system, product testing of a measurement and diagnostic equipment for the electrical power industry as well as a test process of a system integrator of telecommunications solutions. The main analysis across the three cases was conducted qualitatively by exploring documents, tools, and interview protocols. The study revealed that all cases follow a common definition of risk relating its likelihood and impact. However, the definition may remain implicit and the degree of formality depends on the application scope, i.e., it increases from project to product, and, furthermore, to process. In the three cases, risk is taken into consideration in virtually all testing activities. Risks are calculated from risk information associated to testable entities for identifying critical areas of the software system to focus testing effort on. Risk-based testing relies on the established standard tool infrastructure for testing which is partially supported by tools for collecting and analyzing measurement data. Finally, central benefits of risk-based testing using risk information to increase the range of testing for detecting additional defects, and to target the most critical defects from the very beginning, as well as the incorporation of risks for informed decision-making in testing are identified.

The findings from the multiple case study on risk-based testing substantiate the defined taxonomy. The observation that risks are taken into account in almost all phases of the test process justifies the alignment of the taxonomy with the consideration of risks in the test process phases. In addition, the categories risk item type, factors, estimation and degree of automation, defined for the class risk assessment as well as their values, are confirmed by the findings following findings (already mentioned): (1) all cases follow a common definition of risk relating its likelihood and impact, (2) risks are calculated from risk information associated to testable entities for identifying critical areas to focus testing effort on, (3) the degree of formality of the risk definition depends on the application scope, and (4) risk-based testing is partially supported by tools for collecting and analyzing measurement data.

The article "Approaches for the combined use of risk analysis and testing: a systematic literature review" by Erdogan et al. [10] provides a systematic literature review on risk-based testing approaches, which use risk analysis to support testing, as well as test-based risk analysis approaches,



IPR2025-01397
CrowdStrike EX1035 Page 8

which use testing to support risk analysis. In this article, the term "risk analysis" is used in the sense of risk assessment. The authors follow the guidelines of Kitchenham and Charters [28] for conducting a systematic literature review. From the search engines Google Scholar, IEEE Xplore Digital Library, SpringerLink, ScienceDirect, and ACM Digital Library a total of 32 peer-reviewed papers reporting approaches for the combined use of risk assessment and testing were selected for inclusion in the survey and grouped by the first author into 24 approaches which were considered for further analysis with the following results: Only two approaches address test-based risk assessment, while the remaining 22 focus solely on various aspects of risk-based testing. From the 22 risk-based testing approaches, four address the combination of risk assessment and testing at a general level, two address model-based risk estimation, five focus on test case generation, three focus on test case analysis, one addresses source code analysis, and one aims at measurement. Besides these six generic types of RBT approaches, specific RBT approaches were identified, i.e., two addressing specific programming paradigms and four targeting specific applications. The analysis further shows that recurring goals of the RBT approaches are to improve effectiveness of testing and to reduce time and costs. Safety and security are in particular of special concern in test case generation approaches. Most approaches are general and not intended to be used in a specific context. For the approaches targeting specific applications, web services/applications and cloud computing are the most common types. The level of formality varies from purely qualitative risk levels without initial risk identification and analysis to rigorous mathematical models. Tool support is rarely reported as only one approach presents a complete tool for automatically generating test cases and four approaches are supported by tool prototypes.

The systematic literature review provides evidence for the values of the risk assessment categories estimation and degree of automation in the RBT taxonomy. Furthermore, each of the generic types of RBT approaches identified in the systematic literature review can be aligned with the taxonomy:

- Approaches addressing the combination of risk assessment and testing at a general level directly follow the top-level classification principle of the taxonomy combining risk assessment with the integration of risk into the test process.
- Approaches with main focus on model-based risk estimation address a formal risk analysis model.
- Approaches with main focus on test case generation address risk-based test design and implementation.
- Approaches with main focus on test case analysis focus on selection and prioritization of generated test cases.
- Approaches based on automatic source code analysis address automatic risk assessment of the factor probability for development artifacts by quantitative and formal estimation.
- Approaches aiming at measurement address the classes risk assessment and risk-based test evaluation in the taxonomy.

Finally, all specific RBT approaches take risk assessment for specific programming paradigms or applications into account, and, besides that, mainly address test selection and prioritization [4,30,31] or test design [8,51]. Only Rosenberg [37] addresses solely risk assessment (aiming at risk-based testing) for a specific programming paradigm, i.e., object-oriented systems.

## 5 Conclusion

In this paper, we presented a taxonomy of risk-based testing. It is aligned with the consideration of risks in all phases of the test process and consists of three top-level classes, i.e., risk drivers, risk assessment, and risk-based test process. Risk drivers can be functionality, safety, and security. Risk assessment comprises the subclasses risk item type, factors, estimation, and degree of automation. The risk-based test process then takes the assessed risks into account to guide the activities of the test process providing risk-based test planning, design, implementation, execution, and evaluation. The taxonomy provides a framework to understand, categorize, assess, and compare risk-based testing approaches to support their selection and tailoring for specific purposes. We further presented the four articles of the STTT Special Section on Risk-Based Testing and discussed them in the context of the taxonomy. Two articles of the special Section [7,32] present novel risk-based testing approaches and were classified according to the taxonomy. The two other articles [10,16] comprise secondary studies on risk-based testing. These studies empirically evaluate existing risk-based testing approaches and were used to additionally check the presented taxonomy of risk-based testing.

**Acknowledgments** This research was partially funded by the research projects MOBSTECO (FWF P 26194-N15), QE LaB - Living Models for Open Systems (FFG 822740), ITEA2 DIAMONDS (Development and Industrial Application of Multi-Domain-Security Testing Technologies), and EU RASEN (Compositional Risk Assessment and Security Testing of Networked Systems).

## References

1. Amland, S.: Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study. J. Syst. Softw. **53**(3), 287–295 (2000)



IPR2025-01397
CrowdStrike EX1035 Page 9

2. Ammann, P., Offutt, J.: Introduction to Software Testing. Cambridge University Press, Cambridge (2008)
3. Bach, J.: Heuristic risk-based testing. Softw. Test. Qual. Eng. Mag. **11**, 99 (1999)
4. Bai, X., Kenett, R.S.: Risk-based adaptive group testing of semantic web services. In: 33rd Annual IEEE international computer software and applications conference (COMPSAC'09). vol. 2, pp. 485–490. IEEE (2009)
5. Bai, X., Kenett, R.S., Yu, W.: Risk assessment and adaptive group testing of semantic web services. Int. J. Softw. Eng. Knowl. Eng. **22**(05), 595–620 (2012)
6. Briand, L.C., Labiche, Y., He, S.: Automating regression test selection based on UML designs. Inf. Softw. Technol. **51**(1), 16–30 (2009)
7. Carrozza, G., Pietrantuono, R., Russo, S.: Dynamic test planning: a study into an industrial context. STTT in this volume (2014)
8. Casado, R., Tuya, J., Younas, M.: Testing long-lived web services transactions using a risk-based approach. In: 10th international conference on quality software. pp. 337–340. IEEE (2010)
9. 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)
10. Erdogan, G., Li, Y., Runde, R.K., Seehusen, F., Stølen, K.: Approaches for the combined use of risk analysis and testing: a systematic literature review. STTT in this volume (2014)
11. Felderer, M., Beer, A.: Using defect taxonomies to improve the maturity of the system test process: results from an industrial case study. In: software quality. Increasing value in software and systems development, pp. 125–146. Springer (2013)
12. Felderer, M., Haisjackl, C., Breu, R., Motz, J.: Integrating manual and automatic risk assessment for risk-based testing, pp. 159–180. Software quality. Process automation in software, development (2012)
13. Felderer, M., Haisjackl, C., Pekar, V., Breu, R.: A risk assessment framework for software testing. In: ISoLA 2014. Springer (2014)
14. Felderer, M., Ramler, R.: Experiences and challenges of introducing risk-based testing in an industrial project. In: Software quality. Increasing value in software and systems development, pp. 10–29. Springer (2013)
15. Felderer, M., Ramler, R.: Integrating risk-based testing in industrial test processes. Softw. Qual. J. **22**(3), 543–575 (2014)
16. Felderer, M., Ramler, R.: A multiple case study on risk-based testing in industry. STTT in this volume (2014)
17. Fredriksen, R., Kristiansen, M., Gran, B.A., Stølen, K., Opperud, T.A., Dimitrakos, T.: The coras framework for a model-based risk management process. In: Anderson, S., Bologna, S., Felici, M. (eds.) SAFECOMP. Lecture Notes in Computer Science, vol. 2434, pp. 94–105. Springer (2002)
18. Gerrard, P., Thompson, N.: Risk-based e-business testing. Artech House Publishers, (2002)
19. Goel, A.L.: Software reliability models: assumptions, limitations, and applicability. IEEE Trans. Softw. Eng. **11**(12), 1411–1423 (Dec 1985)
20. Graham, D., Fewster, M.: Experiences of test automation: case studies of software test automation. Addison-Wesley Professional, (2012)
21. Hosseingholizadeh, A.: A source-based risk analysis approach for software test optimization. In: Computer Engineering and Technology (ICCET), 2010 2nd international conference on. vol. 2, pp. V2601–V2604. IEEE (2010)
22. Huizinga, D., Kolawa, A.: Automated defect prevention: best practices in software management. Wiley (2007)
23. IEEE: IEEE Standard for Software and System Test Documentation. IEEE Std 829–2008 (2008)
24. ISO: ISO 14971: medical devices—application of risk management to medical devices. ISO (2000)
25. ISO: ISO/IEC/IEEE 29119 Software Testing. http://www.softwaretestingstandard.org/ (2013). Accessed 6 May 2014
26. ISTQB: Standard glossary of terms used in software testing. version 2.2. Tech. rep., ISTQB (2012)
27. Jorgensen, M., Boehm, B., Rifkin, S.: Software development effort estimation: formal models or expert judgment? IEEE Softw. **26**(2), 14–19 (2009)
28. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Tech. rep., Technical report, EBSE Technical Report EBSE-2007-01 (2007)
29. Kloos, J., Hussain, T., Eschbach, R.: Risk-based testing of safety-critical embedded systems driven by fault tree analysis. In: ICSTW 2011. pp. 26–33. IEEE (2011)
30. Kumar, N., Sosale, D., Konuganti, S.N., Rathi, A.: Enabling the adoption of aspects-testing aspects: a risk model, fault model and patterns. In: proceedings of the 8th ACM international conference on Aspect-oriented software development. pp. 197–206. ACM (2009)
31. Murthy, K.K., Thakkar, K.R., Laxminarayan, S.: Leveraging risk based testing in enterprise systems security validation. In: first international conference on emerging network intelligence. pp. 111–116. IEEE (2009)
32. Neubauer, J., Windmüller, S., Steffen, B.: Risk-based testing via active continuous quality control. STTT in this volume (2014)
33. Radatz, J., Geraci, A., Katki, F.: IEEE standard glossary of software engineering terminology. IEEE Std. **610121990**, 121990 (1990)
34. Ray, M., Mohapatra, D.P.: Risk analysis: a guiding force in the improvement of testing. IET Softw. **7**(1), 29–46 (2013)
35. Redmill, F.: Exploring risk-based testing and its implications. Softw. Test. Verif. Reliab. **14**(1), 3–15 (2004)
36. Redmill, F.: Theory and practice of risk-based testing. Softw. Test. Verif. Reliab. **15**(1), 3–20 (2005)
37. Rosenberg, L., Stapko, R., Gallo, A.: Risk-based object oriented testing. Proceedings of 13th international software/internet quality week-QW 2 (2000)
38. Schieferdecker, I., Grossmann, J., Schneider, M.: Model-based security testing. Proceedings 7th workshop on model-based testing (2012)
39. 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)
40. Souza, E., Gusmão, C., Venâncio, J.: Risk-based testing: A case study. In: information technology: new generations (ITNG), 2010 seventh international conference on. pp. 1032–1037. IEEE (2010)
41. Stallbaum, H., Metzger, A.: Employing requirements metrics for automating early risk assessment. Proceedings of MeReP07, Palma de Mallorca, Spain. pp. 1–12 (2007)
42. 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)
43. 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. AST '08, ACM, New York, NY, USA (2008)
44. Standards Australia/New Zealand: risk management AS/NZS 4360:2004 (2004)
45. Tran, V., Liu, D.B.: A risk-mitigating model for the development of reliable and maintainable large-scale commercial-off-the-shelf integrated software systems. In: reliability and maintainability symposium. 1997 proceedings, annual. pp. 361–367 (1997)
46. van Veenendaal, E.: Practical risk-based testing—The PRISMA Approach. UTN Publishers (2012)




IPR2025-01397
CrowdStrike EX1035 Page 10

47. Wendland, M.F., Kranz, M., Schieferdecker, I.: A systematic approach to risk-based testing using risk-annotated requirements models. ICSEA **2012**, 636–642 (2012)
48. Windmüller, S., Neubauer, J., Steffen, B., Howar, F., Bauer, O.: Active continuous quality control. In: proceedings of the 16th international ACM sigsoft symposium on component-based software engineering. pp. 111–120. ACM (2013)
49. Yoo, S., Harman, M.: Regression testing minimization, selection and prioritization: a survey. Softw. Test. Verif. Reliab. **22**(2), 67–120 (Mar 2012)
50. Yoon, H., Choi, B.: A test case prioritization based on degree of risk exposure and its empirical study. Int. J. Softw. Eng. Know. Eng. **21**(02), 191–209 (2011)
51. Zech, P.: Risk-based security testing in cloud computing environments. In: ICST 2011. pp. 411–414. IEEE (2011)
52. Zech, P., Felderer, M., Breu, R.: Towards risk-driven security testing of service centric systems. In: QSIC. pp. 140–143 (2012)
53. 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: EWDC 2009 (2009)



View publication stats

IPR2025-01397
CrowdStrike EX1035 Page 11