1

Recent Advances in Classifying Risk-Based Testing Approaches

Michael Felderer, Jürgen Großmann, and Ina Schieferdecker

Synopsis

In order to optimize the usage of testing efforts and to assess risks of software-based systems, risk-based testing uses risk (re-)assessments to steer all phases in a test process. Several risk-based testing approaches have been proposed in academia and/or applied in industry, so that the determination of principal concepts and methods in risk-based testing is needed to enable a comparison of the weaknesses and strengths of different risk-based testing approaches. In this chapter we provide an (updated) taxonomy of risk-based testing aligned with risk considerations in all phases of a test process. It consists of three top-level classes: contextual setup, risk assessment, and risk-based test strategy. This taxonomy provides a framework to understand, categorize, assess, and compare risk-based testing approaches to support their selection and tailoring for specific purposes. Furthermore, we position four recent risk-based testing approaches into the taxonomy in order to demonstrate its application and alignment with available risk-based testing approaches.

1.1 Introduction

Testing of safety-critical, security-critical or mission-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 failures that harm the critical functions of the software. However, 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 and be driven by business risks.

Both decision problems can be adequately addressed by risk-based testing, which considers the risks of the software product as the guiding factor to steer all the phases of a test process, i.e., test planning, design, implementation, execution, and evaluation (Gerrard and Thompson, 2002; Felderer and Ramler, 2014a; Felderer and Schieferdecker, 2014). Risk-based testing is a pragmatic approach widely used in companies of all sizes (Felderer and Ramler, 2014b, 2016) which uses the straightforward idea of focusing test activities on those scenarios that trigger the most critical situations of a software system (Wendland et al., 2012).
The recent international standard ISO/IEC/IEEE 29119 Software Testing (ISO, 2013) on testing techniques, processes, and documentation even explicitly specifies risk considerations to be 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 (Felderer et al., 2014), methodological support to categorize, assess, compare, and select risk-based testing approaches is required.

In this paper, we present an (updated) taxonomy of risk-based testing that provides a framework for understanding, categorizing, assessing, and comparing risk-based testing approaches and that supports the selection and tailoring of risk-based testing approaches for specific purposes. To demonstrate the application of the taxonomy and its alignment with available risk-based testing approaches, we position four recent risk-based testing approaches, the RASEN approach (Großmann et al., 2015), the SmartTesting approach (Ramler and Felderer, 2015), risk-based test case prioritization based on the notion of risk exposure (Yoon and Choi, 2011), and risk-based testing of open source software (Yahav et al., 2014a), in the taxonomy.

A taxonomy defines a hierarchy of classes (also referred to as categories, dimensions, criteria, or characteristics) to categorize things and concepts. It describes a tree structure whose leaves define concrete values to characterize instances in the taxonomy. The proposed taxonomy is aligned with the consideration of risks in all phases of the test process and consists of the top-level classes context (with subclasses risk driver, quality property, and risk item), risk assessment (with subclasses factor, estimation technique, scale, and degree of automation), and risk-based test strategy (with subclasses risk-based test planning, risk-based test design and implementation, and risk-based test execution and evaluation). The taxonomy presented in this chapter extends and refines our previous taxonomy of risk-based testing (Felderer and Schieferdecker, 2014).

The remainder of this chapter is structured as follows. Section 1.2 presents background on software testing and risk management. Section 1.3 introduces the taxonomy of risk-based testing. Section 1.4 presents the four selected recent risk-based testing approaches and discusses them in the context of the taxonomy. Finally, Section 1.5 summarizes this chapter.

1.2 Background on Software Testing and Risk Management

1.2.1 Software Testing

Software testing (ISTQB, 2012) is the process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation, and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose, and to detect defects. According to this definition it comprises static activities like reviews but also dynamic activities like classic black- or white-box testing. The tested software-based system is called the system under test (SUT). As highlighted before, risk-based testing (RBT) is a testing approach which considers the risks of the software product as the guiding factor to support decisions in all phases of the test process (Gerrard and Thompson, 2002; Felderer and Ramler, 2014a; Felderer and Schieferdecker, 2014). A risk is a factor that could result in future negative consequences and is usually expressed by its likelihood and
Classifying Risk-Based Testing Approaches

Figure 1.1 Core test process steps.

- Test planning
- Test design
- Test implementation
- Test execution
- Test evaluation

impact (ISTQB, 2012). 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.

Risk-based testing is a testing-based approach to risk management that 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* consists of the core activities test planning, test design, test implementation, test execution, and test evaluation (ISTQB, 2012) – see Figure 1.1. In the following, we explain the particular activities and associated concepts in more detail.

According to ISO (2013) and ISTQB (2012), *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 the reason or purpose for designing and executing a test. The reason is either to check the functional behavior of the system or its non-functional properties. *Functional testing* is concerned with assessing the functional behavior of an SUT, whereas *non-functional testing* aims at assessing non-functional 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 (Radatz et al., 1990). *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.

### 1.2.2 Risk Management

Risk management comprises the core activities risk identification, risk analysis, risk treatment, and risk monitoring (Standards Australia/New Zealand, 2004; ISO, 2009). 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 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.

### 1.3 Taxonomy of Risk-Based Testing

The taxonomy of risk-based testing is shown in Figure 1.2. It contains the top-level classes contextual setup, risk assessment, and 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.

#### 1.3.1 Context

The context characterizes the overall context of the risk assessment and testing processes. It includes the subclasses risk driver, quality property, and risk item to characterize the drivers that determine the major assets, the overall quality objectives that need to be fulfilled, and the items that are subject to evaluation by risk assessment and testing.

##### 1.3.1.1 Risk Driver

A risk driver is the first differentiating element of risk-based testing approaches. It characterizes the area of origin for the major assets and thus determines the overall quality requirements and the direction and general setup of the risk-based testing process. Business-related assets are required for a successful business practice and thus often directly
relate to software quality properties like functionality, availability, security, and reliability. **Safety** relates to the inviolability of human health and life and thus requires software to be failsafe, robust, and resilient. **Security** addresses the resilience of information technology systems against threats that jeopardize confidentiality, integrity, and availability of digital information and related services. Finally, **compliance** relates to assets that are directly derived from rules and regulations, whether applicable laws, standards, or other forms of governing settlements. Protection of these assets often, but not exclusively, relates to quality properties like security, reliability, and compatibility.
1.3.1.2 Quality Property

A quality property is a distinct quality attribute (ISO, 2011) which contributes to the protection of assets, and thus is subject to risk assessment and testing. As stated in ISO (2000), risks result from hazards. Hazards related to software-based systems stem from software vulnerabilities and from defects in software functionalities that are critical to business cases, safety-related aspects, security of systems, or applicable rules and regulations.

One needs to test that a software-based system is

- functionally suitable, i.e., able to deliver services as requested;
- reliable, i.e., able to deliver services as specified over a period of time;
- usable, i.e., satisfies the user expectation;
- performant and efficient, i.e., able to react appropriately with respect to stated resources and time;
- secure, i.e., able to remain protected against accidental or deliberate attacks;
- resilient, i.e., able to recover in a timely manner from unexpected events;
- safe, i.e., able to operate without harmful states.

The quality properties considered determine which testing is appropriate and has to be chosen. We consider functionality, security, and reliability to be the dominant quality properties that are addressed for software. Together they 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.

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. The quality properties determine the test types and test techniques that are applied in a test process to find software defects or systematically provide belief in the absence of such defects. Functional testing is likewise a major test type in RBT to analyze reliability and safety aspects – see, e.g., Amland (2000). In addition, security testing including penetration testing, fuzz testing, and/or randomized testing is key in RBT (Zech, 2011; ETSI, 2015b) to analyze security and resilience aspects. Furthermore, performance and scalability testing focusing on normal load, maximal load, and overload scenarios analyze availability and resilience aspects – see, e.g., Amland (2000).

1.3.1.3 Risk Item

The risk item characterizes and determines the elements under evaluation. These risk items are the elements to which risk exposures and tests are assigned (Felderer and Ramler, 2013). Risk items can be of type test case (Yoon and Choi, 2011), 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 a 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.
1.3.2 Risk Assessment

The second differentiating element of RBT approaches is the way risks are determined. According to ISTQB (2012), 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 the factors influencing risks, the risk estimation technique used to estimate and/or evaluate the risk, the scale type that is used to characterize the risk exposure, and the degree of automation for risk assessment.

1.3.2.1 Factor
The risk factors quantify identified risks (Bai et al., 2012). 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 typically considers aspects like liability issues, property loss or damage, and product demand shifts. Risk-based testing 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.

1.3.2.2 Estimation Technique
The estimation technique determines how the risk exposure is actually estimated and can be list based or a formal model (Jørgensen et al., 2009). The essential difference between formal-model- and list-based 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 complex, multivalued quantification step such as a formula or a test model. On the other hand, list-based estimation methods are based on a simple quantification step – for example, what the expert believes is riskiest. List-based estimation processes range from pure “gut feelings” to structured, historical data including failure history and checklist-based estimation processes.

1.3.2.3 Scale
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. A qualitative scale often used for risk levels is low, medium, and high (Wendland et al., 2012).

1.3.2.4 Degree of Automation
Risk assessment can be supported by automated methods and tools. For example, risk-oriented metrics can be measured manually or automatically. Manual measurement is often supported by strict guidelines, and 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, Fredriksen et al. (2002).

1.3.3 Risk-Based Testing Strategy

Based on the risks being determined and characterized, RBT follows the fundamental test process (ISTQB, 2012) or variations thereof. The notion of risk can be used to
optimize already existing testing activities by introducing risk-based strategies for prioritization, automation, selection, resource planning, etc. Depending on the approach, nearly all activities and phases in a test process may be affected by taking a risk-based perspective. This taxonomy aims to highlight and characterize the RBT specifics by relating them to the major phases of a normal test process. For the sake of brevity, we have focused on the phases risk-based test planning, risk-based test design and implementation, and risk-based test execution and evaluation, which are outlined in the following subsections.

1.3.3.1 Risk-Based Test Planning
The main outcome of test planning is a test strategy and a plan that depicts the staffing, the required resources, and a schedule for the individual testing activities. Test planning establishes or updates the scope, approach, resources, and schedule of intended test activities. Amongst other things, test objectives, test techniques, and test completion criteria that impact risk-based testing (Redmill, 2005) are determined.

**Test Objective and Technique** Test objectives and techniques are relevant parts of a test strategy. They determine what to test and how to test a test item. The reason for designing or executing a test, i.e., a test objective, can be related to the risk item to be tested, to the threat scenarios of a risk item, or to the countermeasures established to secure that risk item; see also Section 1.3.3.2. The selection of adequate test techniques can be done on the basis of the quality properties as well as from information related to defects, vulnerabilities, and threat scenarios coming from risk assessment.

**Test Completion Criterion** Typical exit criteria for testing that are used to report against and to plan when to stop testing include all tests running successfully, all issues having been retested and signed off, or all acceptance criteria having been met. Specific RBT-related exit criteria (Amland, 2000) add criteria on the residual risk in the product- and coverage-related criteria: all risk items, their threat scenarios, and/or countermeasures being covered. Risk-based metrics are used to quantify different aspects in testing such as the minimum level of testing, extra testing needed because of a high number of faults found, or the quality of the tests and the test process. They are used to manage the RBT process and optimize it with respect to time, effort, and quality (Amland, 2000).

**Resource Planning and Scheduling** Risk-based testing 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 a minimum of 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 of failing and need to be tested more thoroughly. Within this context, information from risk assessment can be used to roughly identify high-risk areas or features of the SUT and thus determine and optimize the respective test effort, the required personnel and their qualifications, and the scheduling and prioritization of the activities in a test process.
1.3.3.2 Risk-Based Test Design and Implementation

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 depends on the test types needed to realize a test objective. These test types directly relate to the quality property defined in Section 1.3.1. 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 (ISTQB, 2012). Risk aspects are especially essential for providing logging support and for test automation.

Coverage Item Determination Risk-based testing uses coverage criteria specific to the risk artifacts, and test types specific to the risk drivers on functionality, security, and safety. The classical code-oriented and model-based coverage criteria like path coverage, condition-oriented coverage criteria like modified condition decision coverage, and requirements-oriented coverage criteria like requirements or use case coverage are extended with coverage criteria to cover selected or all assets, threat scenarios, and countermeasures (Stallbaum et al., 2008). While asset coverage rather belongs to requirements-oriented coverage (Wendland et al., 2012), threat scenario and vulnerability coverage and countermeasure coverage can be addressed by code-oriented, model-based, and/or condition-oriented coverage criteria (Hosseingholizadeh, 2010).

Test or Feature Prioritization Selection In order 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 used (Yoo and Harman, 2012). Within 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 (Briand et al., 2009) and coverage-based approaches (Amland, 2000) for test prioritization are well suited to RBT. Depending on the approach, prioritization and selection can take place during different phases of the test process. Risk-based feature or requirement prioritization and selection selects the requirements or features to be tested. This activity is usually started during test planning and continued during test design. Test case prioritization and selection requires existing test specifications or test cases. It is thus either carried out before test implementation, to determine the test case to be implemented, or in the preparation of test execution or regression testing, to determine the optimal test sets to be executed.

Test Case Derivation/Generation Risk assessment often comprises information about threat scenarios, faults and vulnerabilities that can be used to derive the test data, the test actions, probably the expected results, and other testing artifacts. Especially when addressing publicly known threat scenarios, these scenarios can be used to directly refer to predefined and reusable test specification fragments, i.e., so-called test pattern. These test patterns already contain test actions and test data that are directly applicable to either test specification, test implementation, or test execution (Botella et al., 2014).

---

1 The ALARP principle is typically used for safety-critical systems, but also for mission-critical systems. It says that the residual risk shall be as low as reasonably practical.
**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 predicted outcomes (Huizinga and Kolawa, 2007). Experiences from test automation (Graham and Fewster, 2012) 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 in guiding decisions as to where and to what degree testing should be automated.

1.3.3.3 Risk-Based Test Execution and Evaluation

*Test execution* is the process of running test cases. In this phase, risk-based testing is supported by *monitoring* and *risk metric measurement*. *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 *reassessment*. Furthermore, risks may guide *test exit decisions* and *reporting*.

**Monitoring and Risk Metric Measurement**  Monitoring is run concurrently with an SUT and supervises, records, or analyzes the behavior of the running system (Radatz et al., 1990; ISTQB, 2012). 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 metric measurement* determines risk metrics defined in the test planning phase. A measured risk metric could be the number of observed critical failures for risk items where failure has a high impact (Felderer and Beer, 2013).

**Risk Reporting**  Test reports are documents summarizing testing activities and results (ISTQB, 2012) 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 that 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, which provide a high level view on risks using the 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 (Goel, 1985).

**Test and Risk Reassessment**  The *reassessment 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 reassessment to adjust them. Test results can be explicitly integrated into a formal risk analysis model (Stallbaum and Metzger, 2007), or just trigger the reassessment in an informal way.
Test Exit Decision  The test exit decision determines if and when to stop testing (Felderer and Ramler, 2013), 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 (Tran and Liu, 1997). 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 occurring in the field. Such measures may be bug fixing, redesign of test cases, or re-execution of test cases.

1.4 Classification of Recent Risk-Based Testing Approaches

In this section, we present four recent risk-based testing approaches: the RASEN approach (Section 1.4.1), the SmartTesting approach (Section 1.4.2), risk-based test case prioritization based on the notion of risk exposure (Section 1.4.3), and risk-based testing of open source software (Section 1.4.4); we position each in the risk-based testing taxonomy presented in the previous section.

1.4.1 The RASEN Approach

1.4.1.1 Description of the Approach
The RASEN project (www.rasen-project.eu) has developed a process for combining compliance assessment, security risk assessment, and security testing based on existing standards like ISO 31000 and ISO 29119. The approach is currently extended in the PREVENT project (www.prevent-project.org) to cover business-driven security risk and compliance management for critical banking infrastructure. Figure 1.3 shows an overview of the RASEN process.

The process covers three distinguishable workstreams that each consist of a combination of typical compliance assessment, security risk assessment activities, and/or security testing activities, emphasizing the interplay and synergies between these formerly independent assessment approaches.

1) The test-based security risk assessment workstream starts like a typical risk assessment workstream and uses testing results to guide and improve the risk assessment. Security testing is used to provide feedback on actually existing vulnerabilities that have not been covered during risk assessment, or allows risk values to be adjusted on the basis of tangible measurements like test results. Security testing should provide a concise feedback as to whether the properties of the target under assessment have really been met by the risk assessment.

2) The risk-based compliance assessment workstream targets the identification and treatment of compliance issues. It relies on security risk assessment results to identify compliance risk and thus systematize the identification of compliance issues. Moreover, legal risk assessment may be used to prioritize the treatment of security issues.
3) The risk-based security testing workstream starts like a typical testing workstream and uses risk assessment results to guide and focus the testing. Such a workstream starts by identifying the areas of risk within the target’s business processes, and building and prioritizing the testing program around these risks. In this setting, risks help focus the testing resources on the areas that are most likely to cause concern, or support the selection of test techniques dedicated to already identified threat scenarios.

According ISO 31000, all workstreams start with a preparatory phase called *Establishing the Context* that includes preparatory activities like understanding the business and regulatory environment as well as the requirements and processes. During this first phase, the high level security objectives are identified and documented, and the overall process planning is done. Moreover, the process shows additional support activities like *Communication and Consult* and *Monitoring and Review* that are meant to set up the management perspective, thus to continuously control, react, and improve all relevant information and the results of the process. From a process point of view, these activities are meant to provide the contextual and management-related framework. The individual activities covered in these phases might differ in detail depending on whether the risk assessment or testing activities are the guiding activities. The main phase, namely the *Security Assessment* phase, covers the definition of the integrated compliance assessment, risk assessment, and security testing workstreams.

**The Risk Assessment Workstream**  The overall risk assessment workstream is decomposed into the three main activities *Risk Identification, Risk Estimation*, and *Risk Evaluation.*
RASEN has extended the risk identification and risk estimation activities with security testing activities in order to improve the accuracy and efficiency of the overall workstream.

Risk identification is the process of finding, recognizing, and describing risks. This consists of identifying sources of risk (e.g., threats and vulnerabilities), areas of impacts (e.g., the assets), malicious events, their causes, and their potential impact on assets. In this context, security testing is used to obtain information that eases and supports the identification of threats and threat scenarios. Appropriate are testing and analysis techniques that yield information about the interfaces and entry points (i.e., the attack surface) like automated security testing, network discovery, web crawling, and fuzz testing.

Following risk identification, risk estimation is the process of expressing the likelihood, intensity, and magnitude of the identified risks. In many cases, the relevant information on potential threats is often imprecise and insufficient, so that estimation often relies on expert judgment only. This, amongst others, might result in a high degree of uncertainty related to the correctness of the estimates. Testing or test-based risk estimation may increase the amount of information on the target of evaluation. Testing might in particular provide feedback regarding the resilience of systems, i.e., it can support the estimation of the likelihood that an attack will be successful if initiated. Information from testing on the presence or absence of potential vulnerabilities has direct impact on the likelihood values of the associated threat scenarios. Similar to test-based risk identification, penetrating testing tools, model-based security testing tools, static and dynamic code analysis tools, and vulnerability scanners are useful for obtaining this kind of information.

The Compliance Assessment Workstream The risk-based compliance assessment workstream consists of three major steps. The compliance risk identification step provides a systematic and template-based approach to identify and select compliance requirements that imply risk. These requirements are transformed into obligations and prohibitions that are the basis for further threat and risk modeling using the CORAS tool. The second step, compliance risk estimation, is dedicated to understanding and documenting the uncertainty that originates from compliance requirement interpretation. Uncertainty may arise from unclear compliance requirements or from uncertainty about the consequences in case of non-compliance. During compliance risk evaluation, compliance requirements are evaluated and prioritized based on their level of risk so that during treatment compliance resources may be allocated efficiently based on their level of risk. In summary, combining security risk assessment and compliance assessment helps to prioritize compliance measures based on risks, and helps to identify and deal with compliance requirements that directly imply risk.

The Security Testing Workstream The risk-based security testing workstream is structured like a typical security testing process. It starts with a test planning phase, followed by a test design and implementation phase, and ends with test execution, analysis, and summary. The result of the risk assessment, i.e., the identified vulnerabilities, threat scenarios, and unwanted incidents, are used to guide the test planning and test identification, and may complement requirements engineering results with systematic information concerning the threats and vulnerabilities of a system.
Factors like probabilities and consequences can additionally be used to weight threat scenarios and thus help identify which threat scenarios are more relevant are thus the ones that need to be treated and tested more carefully. From a process point of view, the interaction between risk assessment and testing could be best described following the phases of a typical testing process.

1) Risk-based security test planning deals with the integration of security risk assessment in the test planning process.
2) Risk-based security test design and implementation deals with the integration of security risk assessment in the test design and implementation process.
3) Risk-based test execution, analysis, and summary deals with risk-based test execution and with the systematic analysis and summary of test results.

1.4.1.2 Positioning in the Risk-Based Testing Taxonomy

Context The overall process (ETSI, 2015a; Großmann and Seehusen, 2015) is directly derived from ISO 31000 and slightly extended to highlight the integration with security testing and compliance assessment. The approach explicitly addresses compliance, but also business and in a limited way safety, as major risk drivers. It is defined independently of any application domain and independently of the level, target, or depth of the security assessment itself. It could be applied to any kind of technical assessment process with the potential to target the full number of quality properties that are defined in Section 1.3.1.2. Moreover, it addresses legal and compliance issues related to data protection and security regulations. Looking at risk-based security testing, the approach emphasizes executable risk items, i.e., runtime artifacts. Considering risk-based compliance assessment, the approach also addresses the other risk items mentioned in the taxonomy.

Risk Assessment The test-based risk assessment workstream uses test results as explicit input to various risk assessment activities. Risk assessment in RASEN has been carried out on the basis of the CORAS method and language. Thus, risk estimation is based on formal models that support the definition of likelihood values for events and impact values to describe the effect of incidents on assets. Both likelihood and impact values are used to calculate the overall risk exposure for unwanted incidents, i.e., the events that directly harm assets. CORAS is flexible with respect to the calculation scheme and to the scale for defining risk factors. It generally supports values with qualitative scale as well as with quantitative scale.

Risk-Based Test Strategy Security is not a functional property and thus requires dedicated information that addresses the (security) context of the system. While functional testing is more or less guided directly by the system specification (i.e., features, requirements, architecture), security testing often is not. The RASEN approach to risk-based security test planning especially addresses the risk-based selection of test objectives and test techniques as well as risk-based resource planning and scheduling. Security risk assessment serves this purpose and can be used to roughly identify high-risk areas or features of the SUT and thus determine and optimize the respective test effort. Moreover, a first assessment of the identified vulnerabilities and threat scenarios may help to select test strategies and techniques that are dedicated to deal with the most critical security risks.
Classifying Risk-Based Testing Approaches

Considering security test design and implementation, especially the selection and prioritization of the feature to test, the concrete test designs and the determination of test coverage items are critical. A recourse to security risks, potential threat scenarios, and potential vulnerabilities provide good guidance for improving item prioritization and selection. Security-risk-related information supports the selection of features and test conditions that require testing. It helps in identifying which coverage items should be covered to what depth, and how individual test cases and test procedures should look. The RASEN approach to risk-based security test design and implementation uses information on expected threats and potential vulnerabilities to systematically determine and identify coverage items (besides others, asset coverage and threat scenario and vulnerabilities coverage), test conditions (testable aspects of a system), and test purposes. Moreover, the security risk assessment provides quantitative estimations on the risks, i.e., the product of frequencies or probabilities and estimated consequences. This information is used to select and prioritize either the test conditions or the actual tests when they are assembled into test sets. Risks as well as their probabilities and consequence values are used to set priorities for the test selection, test case generation, and for the order of test execution expressed by risk-optimized test procedures. Risk-based test execution allows the prioritization of already existing test cases, test sets, or test procedures during regression testing. Risk-based security test evaluation aims to improve risk reporting and the test exit decision by introducing the notion of risk coverage and remaining risks on the basis of the intermediate test results as well as on the basis of the errors, vulnerabilities, or flaws that have been found during testing. In summary, we have identified the three activities that are supported through results from security risk assessment.

1.4.2 The SmartTesting Approach

1.4.2.1 Description of the Approach

Figure 1.4 provides an overview of the overall process. It consists of different steps, which are either directly related to risk-based test strategy development (shown in bold font) or which are used to establish the preconditions (shown in normal font) for the process by linking test strategy development to the related processes (drawn with dashed lines) of defect management, requirements management, and quality management. The different steps are described in detail in the following subsections.

Definition of Risk Items In the first step, the risk items are identified and defined. The risk items are the basic elements of a software product that can be associated with risks. Risk items are typically derived from the functional structure of the software system, but they can also represent non-functional aspects or system properties. In the context of testing it should be taken into account that the risk items need to be mapped to test objects (ISTQB, 2012), i.e., testable objects such as subsystems, features, components, modules, or functional as well as non-functional requirements.

Probability Estimation In this step the probability values (for which an appropriate scale has to be defined) are estimated for each risk item. In the context of testing, the probability value expresses the likelihood of defectiveness of a risk item, i.e., the likelihood that a fault exists in a specific product component due to an error in a previous development phase that may lead to a failure. There are several ways to estimate or predict
the likelihood of a component’s defectiveness. Most of these approaches rely on historical defect data collected from previous releases or related projects. Therefore, defect prediction approaches are well suited to supporting probability estimation (Ramler and Felderer, 2016).

**Impact Estimation** In this step the impact values are estimated for each risk item. The impact values express the consequences of risk items being defective, i.e., the negative effect that a defect in a specific component has on the user or customer and, ultimately, on the company’s business success. The impact is often associated with the cost of failures. The impact is closely related to the expected value of the components for the user or customer. The value is usually determined in requirements engineering when eliciting and prioritizing the system’s requirements. Thus, requirements management may be identified as the main source of data for impact estimation.

---

**Figure 1.4** SmartTesting process (Ramler and Felderer, 2015).
Computation of Risk Values
In this step risk values are computed from the estimated probability and impact values. Risk values can be computed according to the definition of risk as $R = P \times I$, where $P$ is the probability value and $I$ is the impact value. Aggregating the available information to a single risk value per risk item allows the prioritization of the risk items according to their associated risk values or ranks. Furthermore, the computed risk values can be used to group risk items, for example according to high, medium, and low risk. Nevertheless, for identifying risk levels it is recommended to consider probability and impact as two separate dimensions of risk.

Determination of Risk Levels
In this step the spectrum of risk values is partitioned into risk levels. Risk levels are a further level of aggregation. The purpose of distinguishing different risk levels is to define classes of risks such that all risk items associated with a particular class are considered equally risky. As a consequence, all risk items of the same class are subject to the same intensity of quality assurance and test measures.

Definition of Test Strategy
In this step the test strategy is defined on the basis of the different risk levels. For each risk level the test strategy describes how testing is organized and performed. Distinguishing different levels allows testing to be performed with differing levels of rigor in order to adequately address the expected risks. This can be achieved either by applying specific testing techniques (e.g., unit testing, use case testing, beta testing, reviews) or by applying these techniques with more or less intensity according to different coverage criteria (e.g., unit testing at the level of 100% branch coverage or use case testing for basic flows and/or alternative flows).

Refinement of Test Strategy
In the last step the test strategy is refined to match the characteristics of the individual components of the software system (i.e., risk items). The testing techniques and criteria that have been specified in the testing strategy for a particular risk level can be directly mapped to the components associated with that risk level. However, the test strategy is usually rather generic. It does not describe the technical and organizational details that are necessary for applying the specified techniques to a concrete software component. For each component, thus, a test approach has to be developed that clarifies how the test strategy should be implemented.

1.4.2.2 Positioning in the Risk-Based Testing Taxonomy

Context
SmartTesting provides a lightweight process for the development and refinement of a risk-based test strategy. It does not explicitly address risk drivers, but – as in every risk-based testing process – it is implicitly assumed that a risk driver and a quality property to be improved are available. The risk drivers of the broad range of companies involved in the accompanying study (Ramler and Felderer, 2015) cover all types, i.e., business, safety, and compliance. Also, different quality properties of interest are covered, mainly as impact factors. For instance, the companies involved considered performance and security besides functionality as impact factors.

Risk Assessment
SmartTesting explicitly contains a step to define risk items, which can in principle be of any type from the taxonomy. For the companies involved, risk items were typically derived from the system's component structure. Via the process step computation of risk values, SmartTesting explicitly considers risk exposure, which is qualitatively
estimated by a mapping of risk values to risk levels in the process step determination of risk levels. The risk value itself is measured based on a formal model in the process step computation of risk values, which combines values from probability and impact estimation. Probability estimation takes defect data into account, and impact estimation is based on impact factors, which are typically assessed manually.

**Risk-Based Test Strategy** The process steps definition and refinement of the test strategy comprise risk-based test planning resulting in the assignment of concrete techniques, resource planning and scheduling, prioritization and selection strategies, metrics as well as exit criteria to the risk levels and further to particular risk items.

### 1.4.3 Risk-Based Test Case Prioritization Based on the Notion of Risk Exposure

#### 1.4.3.1 Description of the Approach

Choi et al. present different test case prioritization strategies based on the notion of risk exposure. In Yoon and Choi (2011), test case prioritization is described as an activity intended “to find the most important defects as early as possible against the lowest costs” (Redmill, 2005). Choi et al. claim that their risk-based approach to test case prioritization performs well against this background. They empirically evaluate their approach in a setting where various versions of a traffic conflict avoidance system (TCAS) are tested, and show how their approach performs well compared to the prioritization approach of others. In Hettiarachchi et al. (2016) the approach is extended using an improved prioritization algorithm and towards an automated risk estimation process using fuzzy expert systems. A fuzzy expert system is an expert system that uses fuzzy logic instead of Boolean logic to reason about data. Conducting risk estimation with this kind of expert system, Choi et al. aim to replace the human actor during risk estimation and thus avoid subjective estimation results. The second approach has been evaluated by prioritizing test cases for two software products, the electronic health record software iTrust, an open source product, and an industrial software application called Capstone.

#### 1.4.3.2 Positioning in the Risk-Based Testing Taxonomy

**Context** Neither approach explicitly mentions one of the risk drivers from Section 1.3.1.1, nor do they provide exhaustive information on the quality properties addressed. However, in Yoon and Choi (2011) the authors evaluate their approach in the context of a safety critical application. Moreover, the authors emphasize that they refer to risks that are identified and measured during the product risk assessment phase. Such a phase is typically prescribed for safety critical systems. Both facts indicate that safety seems to be the major risk driver, and the safety-relevant attributes like functionality, reliability, and performance are the major quality properties that are addressed by testing. In contrast, the evaluation in Hettiarachchi et al. (2016) is carried out with business critical software, considering quality properties like functionality and security. Both approaches have in common that they do not address compliance as a risk driver.

**Risk Assessment** The risk assessment process for both approaches aims to calculate risk exposure. The authors define risk exposure as a value with a quantitative scale that
expresses the magnitude of a given risk. While in Yoon and Choi (2011) the authors explicitly state that they are intentionally not using their own testing-related equivalent for expressing risk exposure but directly refer to risk values coming from a pre-existing risk assessment, risk estimation in Hettiarachchi et al. (2016) is done automatically and tailored towards testing. The authors calculate risks on the basis of a number of indicators that are harvested from development artifacts like requirements. They use properties like requirements modification status and frequency as well as requirements complexity and size to determine the risk likelihood and risk impact for each requirement. In addition, indicators on potential security threats are used to address and consider the notion of security. In contrast to Yoon and Choi (2011), Hettiarachchi et al. (2016) address the automation of the risk estimation process using an expert system that is able to aggregate the risk indicators and thus to automatically compute the overall risk exposure. While Yoon and Choi (2011) do not explicitly state whether the initial risk assessment relies on formal models or not, the approach in Hettiarachchi et al. (2016) is completely formal. However, since Yoon and Choi (2011) refer to safety critical systems, we can assume that the assessment is not just a list-based assessment.

Risk-Based Test Strategy With respect to testing, both approaches aim for an efficient test prioritization and selection algorithm. Thus, they are mainly applicable in situations where test cases, or at least test case specifications, are already available. This first of all addresses regression testing, and also decision problems during test management, e.g., when test cases are already specified and the prioritization of test implementation efforts is required.

To obtain an efficient test prioritization strategy, both approaches aim to derive risk-related weights for individual test cases. In Yoon and Choi (2011), the authors propose two different strategies. The first strategy aims for simple risk coverage. Test cases that cover a given risk obtain a weight that directly relates to the risk exposure for that risk. If a test case covers multiple risks, the risk exposure values are summed. The second strategy additionally tries to consider the fault-revealing capabilities of the test cases. Thus, the risk-related weight for a test case is calculated by means of the risk exposure for a given risk correlated with the number of risk-related faults that are detectable by that test case, so that test cases with higher fault-revealing capabilities are rated higher. The fault-revealing capabilities of test cases are derived through mutation analysis; i.e., this strategy requires that the test cases already exist and that they are executable.

In Hettiarachchi et al. (2016), test cases are prioritized on the basis of their relationship to risk-rated requirements. Risk rating for requirements is determined by an automated risk rating conducted by the fuzzy expert system, and additional analysis of fault classes and their relation to the individual requirements. In short, a fault class is considered to have more impact if it relates to requirements with a higher risk exposure. In addition, a fault of a given fault class is considered to occur more often if that fault class relates to a larger number of requirements. Both values determine the overall risk rating for the individual requirements and thus provide the prioritization criteria for requirements. Finally, test cases are ordered by means of their relationship to the prioritized requirements. During the evaluation of the approach, the authors obtained the relationship between test cases and requirements from existing traceability information.
While both approaches provide strong support for risk-based item selection, they do not support other activities during risk-based test design and implementation, nor do they establish dedicated activities in the area of risk-based test execution and evaluation.

1.4.4 Risk-Based Testing of Open Source Software

1.4.4.1 Description of the Approach
Yahav et al. (2014a,b) provide an approach to risk-based testing of open source software (OSS) to select and schedule dynamic testing based on software risk analysis. Risk levels of open source components or projects are computed based on communication between developers and users in the open source software community. Communication channels usually include mail, chats, blogs, and repositories of bugs and fixes. The provided data-driven testing approach therefore builds on three repositories, i.e., a social repository, which stores the social network data from the mined OSS community; a bug repository, which links the community behavior and OSS quality; and a test repository, which traces tests (and/or test scripts) to OSS projects. As a preprocessing step, OSS community analytics is performed to construct a social network of communication between developers and users. In a concrete case study (Yahav et al., 2014b), the approach predicts the expected number of defects for a specific project with logistic regression based on the email communication and the time since the last bug.

1.4.4.2 Positioning in the Risk-Based Testing Taxonomy

Context The approach does not explicitly mention one of the risk drivers from Section 1.3.1.1, nor the quality properties. However, the authors state that the purpose of risk-based testing is mitigation of the significant failures experienced in product quality, timeliness, and delivery cost when adapting OSS components in commercial software packages (Yahav et al., 2014a). Therefore, risk drivers may be business to guarantee the success of a system, where the tested OSS component is integrated, or even safety, if the OSS component were to be integrated into a safety critical system. Due to the testing context, i.e., selection or prioritization of available tests of OSS components, the main quality property is supposed to be functionality. The risk item type is OSS components (developed in OSS projects), i.e., architectural artifacts.

Risk Assessment The risk assessment approach quantifies the risk likelihood. For this purpose, a formal model is created to predict the number of bugs based on the communication in different communities and the time since the last bug. The scale is therefore quantitative as the approach tries to predict the actual number of bugs. The approach implements an automatic assessment as it uses monitors to automatically store data in repositories and then applies machine learning approaches, i.e., logistic regression, to predict the risk level.

Risk-Based Test Strategy The approach explicitly supports risk-based test planning in terms of test prioritization and selection and resource planning and scheduling. The approach mainly addresses the allocation of available test scripts for dynamic testing and highlights that exhaustive testing is infeasible and that therefore selective testing techniques are needed to allocate test resources to the most critical components. Therefore,
risk-based test design is not explicitly addressed. As risk metrics, the number of communication metrics as well as the time since the last defect are computed. The approach uses specific logging support to log and trace community and defect data. For risk reporting confusion matrices are used, which contrast the actual and predicted number of defects.

### 1.5 Summary

In this chapter, 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: contextual setup, risk assessment, and risk-based test strategy. The contextual setup is defined by risk drivers, quality properties, and risk items. Risk assessment comprises the subclasses factors, estimation technique, scale, and degree of automation. The risk-based test strategy then takes the assessed risks into account to guide test planning, test design and implementation, and test 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. To demonstrate its application and alignment with available risk-based testing approaches, we positioned within the taxonomy four recent risk-based testing approaches: the RASEN approach, the SmartTesting approach, risk-based test case prioritization based on the notion of risk exposure, and risk-based testing of open source software.

### References


Classifying Risk-Based Testing Approaches


