Int J Softw Tools Technol Transfer
DOI 10.1007/s10009-014-0330-5

RBT

Approaches for the combined use of risk analysis and testing: a systematic literature review

Gencer Erdogan · Yan Li · Ragnhild Kobro Runde · Fredrik Seehusen · Ketil Stølen

Abstract Risk analysis and testing are conducted for different purposes. Risk analysis and testing nevertheless involve processes that may be combined to the benefit of both. We may use testing to support risk analysis and risk analysis to support testing. This paper surveys literature on the combined use of risk analysis and testing. First, the existing approaches are identified through a systematic literature review. The identified approaches are then classified and discussed with respect to main goal, context of use and maturity level. The survey highlights the need for more structure and rigor in the definition and presentation of approaches. Evaluations are missing in most cases. The paper may serve as a basis for examining approaches for the combined use of risk analysis and testing, or as a resource for identifying the adequate approach to use.

Keywords Risk-based testing · Test-based risk analysis · Literature survey

1 Introduction

This paper presents results from a systematic literature review addressing the combined use of risk analysis and testing. Risk analysis and testing involve separate processes and are conducted for different purposes. In the case of risk analysis, the aim is to fulfill the risk acceptance criteria; in the case of testing, the aim is to fulfill the test objectives. While the risk acceptance criteria specify the level of risk that can be tolerated, the test objectives specify the goals of the testing to be conducted. There are nevertheless many dependencies between risk analysis and testing. Identified risks may for example motivate a new test project to check whether certain potentially risky scenarios really may occur. On the other hand, an already finished test project may provide valuable input to a risk analysis.

There are basically two main strategies for the combined use of risk analysis and testing:

- The use of testing to support the risk analysis process.
- The use of risk analysis to support the testing process.

We refer to the first strategy as test-based risk analysis and to the second strategy as risk-based testing.

Seven of the 24 approaches in our survey are also included in a recent study by Alam and Khan [1]. Focusing on risk-based testing, they describe each of the approaches in great detail, also including example tables and figures from some of them. In contrast, the main contribution of this paper is a comparison of the approaches with respect to a set of predefined criteria. Using a systematic search process, our survey includes papers on risk-based testing not discussed in [1], while at the same time excluding approaches not published in peer-reviewed journals/proceedings. In addition, we also include papers on test-based risk analysis.

In our survey, we only consider approaches combining risk analysis and testing. Using a search process similar to the one described in this paper, Sulaman et al. [30] review methods for risk analysis of IT systems. Their study may be seen as complementary to the work presented here, as the sets of papers are disjoint.

The remainder of this paper is organized as follows: in Sect. 2, we introduce the basic risk and test terminology; in particular, we define the notions of risk analysis and testing

G. Erdogan · Y. Li · F. Seehusen · K. Stølen
SINTEF ICT, Oslo, Norway

G. Erdogan · Y. Li · R. K. Runde (✉) · K. Stølen
Department of Informatics, University of Oslo, Oslo, Norway
e-mail: ragnhild.runde@ifi.uio.no

Published online: 03 August 2014

and clarify the distinction between test-based risk analysis and risk-based testing. In Sect. 3, we describe our research process; i.e., the various steps and tasks of the literature review. In Sect. 4, we classify the identified approaches and provide a short description of each of the approaches, while Sect. 5 gives an overview of their maturity levels. In Sect. 6, we look at the relationships between the papers based on their list of references. In Sect. 7, we summarize our findings with respect to the research questions defined in Sect. 3, before concluding in Sect. 8.

2 Terminology

In the following, we define the technical terms within risk analysis and testing that we make use of in the literature survey. The first step is of course to explain what we mean by risk analysis and testing.

Risk analysis is a collective term denoting the five-step process consisting of:

- *Establishing the context* describing the target to be analyzed, the assumptions on which to build, the scope of the analysis as well as the risk acceptance criteria.
- *Risk identification* finding, recognizing and describing risks.
- *Risk estimation* estimating the likelihood and consequence of risk.
- *Risk evaluation* comparing the results of the risk estimation with the risk acceptance matrix and prioritizing risks.
- *Risk treatment* identifying and evaluating means to mitigate, eliminate and prevent risks.

This corresponds to the five processes referred to as establishing the context, risk identification, risk analysis, risk evaluation and risk treatment in ISO 31000 [15]. Hence, we use the term risk analysis in a more general manner than ISO 31000. There are two reasons for this: firstly, we think that this better reflects how the term is used in practice. Secondly, ISO 31000 does not offer a collective term for these five processes; they are just five out of the seven processes that make up what ISO 31000 refers to as the risk management process.

Testing is the process of exercising the system to verify that it satisfies specified requirements and to detect errors (adapted from ISO 29119 [16]). A typical testing process may also be divided into five steps:

- *Test planning* identifying test objectives and developing the test plan.
- *Test design and implementation* deriving the test cases and test procedures.
- *Test environment set-up and maintenance* establishing and maintaining the environment in which tests are executed.
- *Test execution* running the test procedure resulting from the test design and implementation on the test environment established by the previous step.
- *Test incident reporting* managing the test incidents.

We define test-based risk analysis (TR for short) as risk analysis that makes use of testing to identify potential risks, validate risk analysis results, or both. On the other hand, risk-based testing (RT for short) is testing that makes use of risk analysis to identify tests, select between tests and testing approaches, or both. To decide whether an approach to the combined use of testing and risk analysis is of type TR or RT, we employ the following criteria:

- A TR approach aims to fulfill the risk acceptance criteria. The testing is used to identify and validate the risk analysis results, and the risk analysis results are updated based on the results from the testing.
- An RT approach aims to fulfill the test objectives. The overall purpose is to test and the risk analysis is used only for test identification and prioritization at one or more points during the testing process.

3 Research process

The systematic literature review process conducted in this study consists of the following six steps: (1) define the objective of the study; (2) define research questions; (3) define the search process including inclusion and exclusion criteria; (4) perform the search process; (5) extract data from relevant full texts; (6) analyze data and provide answers for the research questions. This process has been constructed based on the guidelines given by Kitchenham and Charters [17].

- *Objective* The objective is to review and to bring forth a state of art of the current approaches for the combined use of risk analysis and testing, based on publications related to this topic.
- *Research questions* We defined the following research questions (RQ) to fulfill our objectives:

RQ1 What approaches exist for performing risk-based testing (RT) and test-based risk analysis (TR)?
RQ2 For each of the identified approaches, what is the main goal, and what strategies are used to achieve that goal?
RQ3 Are there contexts in which TR and RT are considered to be particularly useful?
RQ4 How mature are the approaches, considering degree of formalization, empirical evaluation, and tool support?
RQ5 What are the relationships between the approaches, considering citations between the publications?

– *Inclusion and exclusion criteria* Papers should be included if they fulfill both of the following criteria:

  – The paper is written in English, published in a peer-reviewed journal/magazine (including online first publication), or as part of the proceedings from a conference/workshop.
  – The paper describes an approach (e.g., method, process, or guidelines) combining risk analysis and testing, specifically related to computer science.

Papers should be excluded if they fulfill one or more of the following criteria:

  – The paper is written in a language other than English.
  – The paper is an unpublished paper, white paper, technical report, thesis, patent, general web page, presentation, or a book chapter. Book chapters are excluded from the survey as scientific books should be based on papers already published in journals/magazines or conference/workshop proceedings, and such papers are already included in the survey.
  – The paper describes an approach not related to computer science.
  – The paper consists only of informal discussions of combining risk analysis and testing, without research questions aimed at combining the two, or without presenting a concrete approach.
  – The paper considers risk analysis and testing as separate activities, or considers only one of them.

– *Search process* The search process was performed manually using the following search engines:

  – Google Scholar (http://scholar.google.com/)
  – IEEE Explore (http://ieeexplore.ieee.org/Xplore/)
  – SpringerLink (http://link.springer.com/)
  – Science Direct (http://www.sciencedirect.com/)
  – ACM (http://dl.acm.org/)

Step 1 The responsibility for each of the five search engines was divided among three of the researchers. Based on an initial literature search, 23 search strings were identified (see Appendix).

Step 2 Each of the 23 search strings was executed in all five search engines. For each search, potentially relevant papers were selected based on the titles and abstracts only, until finding 50 consecutive irrelevant results.

Step 3 For all the papers selected, their title and abstract were read by all three researchers before they voted for or against the inclusion of the paper in the study. Then, based on the individual voting, the researchers voted on the papers collectively by discussing their individual votes. The reasons for rejecting a paper were documented. Additionally, all duplicates were excluded from the literature collection.

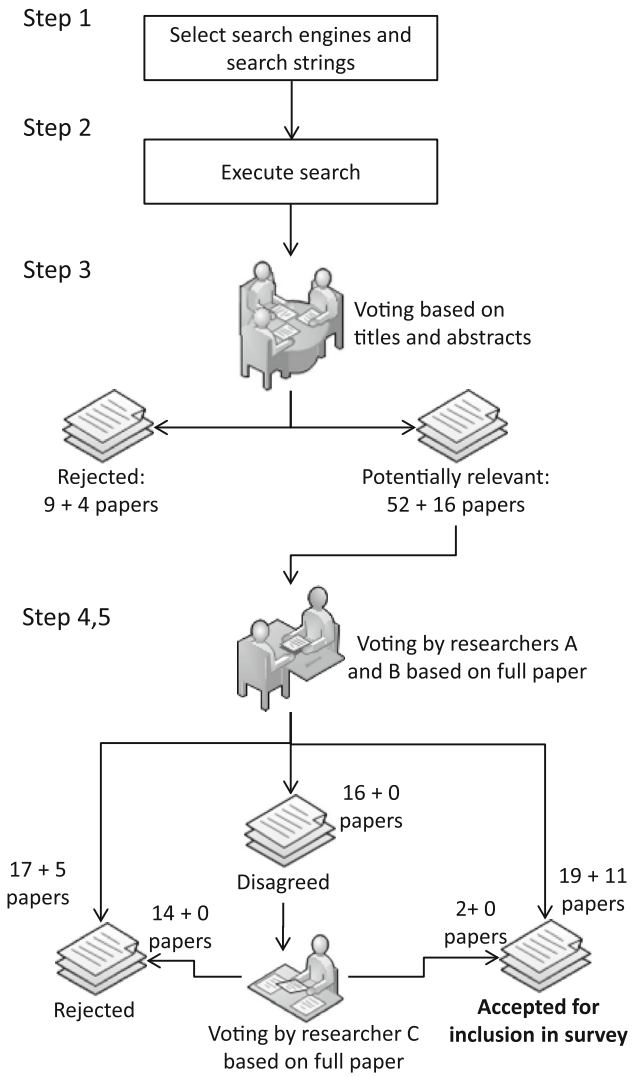

Fig. 1 Overview of search process

Step 4 All the papers that passed the collective voting were reviewed based on the full text and again voted for, separately, by two of the researchers.

Step 5 All papers that were accepted by both researchers were included for further analysis. Papers that were rejected by both researchers were excluded. The papers that were not accepted or rejected by *both* researches were reviewed by the third researcher for a final vote.

After analyzing all the papers found in the main search, a new search was performed to find potentially relevant papers newly added to the databases. Apart from restricting the search to the last three years (2011–2013), the search and voting process were carried out as described above.

Figure 1 illustrates the steps undertaken during the search process, together with the number of papers resulting from

each step. The main search ended in July 2012 and resulted in 61 potentially relevant papers, while the new search ending in May 2013 gave 20 additional papers. In total, this gave a total of 81 papers for further consideration. Based on the inclusion and exclusion criteria, the two-step voting process resulted in 32 (21 + 11) papers to be included in the survey. Sections 4, 5 and 6 provide a summary and classification of the included papers, which are then used as basis for answering the research questions in Sect. 7.

4 Overview and classification of the approaches

For the 32 papers included in this survey, some are written by the same authors and describe different aspects of the same approaches. By grouping the papers based on first author, we end up with 24 approaches for further analysis.

In this section, these approaches are categorized into eight different categories depending on their main focus, and the approaches are described in more detail with special emphasis on their goals and the strategies used to achieve those goals. Hence, this section forms the basis for answering research questions RQ1, RQ2 and RQ3 from Sect. 3.

The eight categories are:

- Approaches addressing the combination of risk analysis and testing at a general level.
- Approaches with main focus on model-based risk estimation.
- Approaches with main focus on test-case generation.
- Approaches with main focus on test-case analysis.
- Approaches based on automatic source code analysis.
- Approaches targeting specific programming paradigms.
- Approaches targeting specific applications.
- Approaches aiming at measurement in the sense that measurement is the main issue.

The categories were selected after the search. They are not completely disjoint; i.e., some papers could be argued to belong to more than one category. The categorization is an attempt to group papers resulting from the search according to topic and relationship, so that those that are related are discussed close to each other. Hence, the categorization is mainly used for presentation purposes.

4.1 Approaches addressing combination at a general level

The four approaches in [2,10,11,23,24], and [34] address risk-based testing at a general level (Table 1). The focus of both [2] and [23,24] is on the risk analysis part. In [2], test case prioritization is achieved by expert meetings where probability indicators and failure costs are determined for each relevant function, and then combined into a risk exposure value. However, other elements, such as frequency of use, may also be taken into account. For [23,24], test case prioritization may also be performed on the basis of either probability or consequence analysis alone.

In contrast, Yoon and Choi [34] assume that the risk exposure value has been predetermined by a domain expert, and uses mutation analysis to assess whether a test case covers a given fault or not. Each test case is then given a total weight based on the correlation with all the risks weighted with their individual risk exposure values.

Felderer et al. [10,11] show a model-based approach to risk-based testing, with the focus on product risks affecting the quality of the product itself. The approach includes a static risk assessment model using the factors of probability, impact, and time, each determined by several criteria. Each criterion is determined by a metric to be evaluated auto-

Table 1 Approaches addressing combination at a general level

Approach	Kind	Goal	Strategy
Amland [2]	RT	Reduce test and production costs	Integration of risk analysis into a testing process to facilitate system testing
Felderer et al. [10,11]	RT	Make product quality testing more effective	A generic risk-based testing process focusing on product risks where the primary effect of a potential problem is on the quality of the product itself
Redmill [23,24]	RT	Make testing more effective	Loose integration of risk analysis and testing at a risk management level
Yoon et al. [34]	RT	Improve the effectiveness of testing	Testing based on results from an independently conducted risk analysis

matically (e.g., code complexity), semi-automatically (e.g., functional complexity), or manually (e.g., frequency of use and importance to the user).

Amland [2] includes a real case study on the application of risk-based testing in a project within the financial domain. The case study indicates a positive effect in reduced time and resources, which also may increase quality as more time may be spent on the critical functions. Yoon and Choi [34] present an experimental evaluation based on a real aircraft collision avoidance system, and finds that their approach is more effective than using risk exposure values alone.

Both [2] and [23,24] emphasize the human and organizational factors. For risk-based testing to reach its potential, there must exist a structured testing process, where everyone involved (e.g., programmers and test managers) understands how the risk analysis results should be used to direct the testing process. In [11], the risk-based testing approach is aligned with standard test processes, and the paper describes four stages for risk-based test integration in a project. Challenges for integration at the initial stage in an anonymized industrial application are discussed, with a number of practical lessons learned. The approach itself is not compared to other approaches.

4.2 Approaches with focus on model-based risk estimation

There are two approaches with main focus on model-based risk estimation, namely [12,13] and [22] (Table 2). Both use state machine diagrams of the system behavior as a starting point. Gleirscher [12,13] analyzes the state machine diagrams to identify hazards threatening the safety of the system. Also, a test model describing both the system and the environment is transformed into a Golog script, from which test cases can be derived.

Ray and Mohapatra [22] estimate reliability-based risks for individual scenarios as well as for the overall system by estimating the complexity for each state in each component and then using well-known hazard techniques for determining the severity of each scenario. Testing is not presented as part of the approach itself, but in the experimental validation using a library management case study. The approach is found to improve test efficiency by finding bugs responsible for severe failures and also finding more faults than two risk analysis approaches used for comparison.

4.3 Approaches with focus on test-case generation

In [21,31], test cases are generated automatically from models with risk annotations. Test-case generation is also the focus of [18,33,37]. [18,21] and [37] focus on safety-critical systems, and [33] on security testing, whereas [31] emphasize the generality of their approach to cover critical situations in any kind of system (Table 3).

For [31], the starting point is a requirements specification formalized as an integrated behavior tree. The tree is then augmented with risk information combining likelihood and severity into risk levels. For each risk exposure, an appropriate test directive should be identified by experienced personnel, describing precisely the test derivation technique and strategy to be used for each risk level. The test themselves can then be generated automatically by the use of a tool.

In [21], fault trees are used as system models, and state charts as system behavior models. Risk analysis information in the form of expected causes of failures are extracted from fault tree analysis and associated with system state chart elements. From this, a risk-based test model is generated automatically, and verified to be correct, complete and consistent using a model checker. The model checking is also used to generate the risk-based test cases.

Fault tree analysis is also used in [18] to identify event sets in order of descending risk. A detailed algorithm is provided for using the event sets to transform the base model, a finite-state automaton describing possible system situations, into a test model from which test cases are generated.

In [33], formal threat models in the form of predicate/transition nets are used to generate the security tests. These are then converted into executable test code using a model-implementation mapping specification.

In [37], the main idea is to use systematic construction or refinement of risk-based test models so that only critical test cases can be generated. The approach uses state charts and model-based statistical testing with Markov chains test models to describe the simulation and system under test.

Tool support is important for both [21] and [31]. While [21] has implemented a prototype tool, Wendland et al. [31] assert that all features discussed in the paper will be integrated in their existing modeling environment tool Fokus!MBT using UML profiles.

Neither [18] nor [33] is supported by a dedicated tool, but both use some existing tools for part of their evaluation. The approach in [18] is demonstrated on a modular production system, where it is found that the approach provides a significant increase in the coverage of safety functions. The approach in [33] is applied in two case studies based on real-world systems, a web-based shopping system and a file server implementation. For both systems, multiple security risks were found, and the approach was also able to kill the majority of the security mutants injected deliberately.

No tool support is mentioned in [37]. The approach is tested on a railway control system, and found to be successful in generating only critical test cases which represent high risk.

Table 2 Approaches with focus on model-based risk estimation

Approach	Kind	Goal	Strategy
Gleirscher [12,13]	RT	Make functional testing of software quality more effective	Functional testing at early stages of system development starting from a test model from which hazards are identified and tested
Ray and Mohapatra [22]	RT	Use risk-based testing to facilitate the detection of faults at the early phase of software development	Estimates risk of various states based on complexity and severity; makes use of UML diagrams

Table 3 Approaches with focus on test-case generation

Approach	Kind	Goal	Strategy
Kloos et al. [18]	RT	Improve testing of safety critical embedded systems	Testing of possible failure modes in order of criticality, where criticality is measured in risk value. The test cases are derived from fault trees
Nazier and Bauer [21]	RT	Facilitate automatic test case generation based on risk	Integrates fault trees into a state chart model from which test cases are generated automatically using a model checker
Wendland et al. [31]	RT	Improve testing by generating risk-optimized test-suites	Test-case generation based on risk information incorporated into the requirements model
Xu et al. [33]	RT	Improve productivity and reduce cost of security testing	Automated generation of security threats from formal threat models
Zimmermann et al. [37]	RT	Improve productivity and reduce cost of safety testing	Algorithmic method for modifying a given model for test case generation in such a way that only test cases satisfying criticality conditions are generated

4.4 Approaches with focus on test-case analysis

The approaches in [7–9,29] focus on analyzing the test cases generated to select and prioritize the most critical ones based on risk (Table 4). For [7,8,29], UML activity diagrams are used as the basic model from which test cases are generated, whereas [9] uses state charts.

For each test case, [7,8] starts by estimating the cost based on the consequence of a fault for the customer or the vendor, and then deriving what the authors refer to as the severity probability for each test case based on the number and severity of defects. Finally, risk exposure (cost multiplied with probability of occurrence) is calculated for each test case, before those with the highest risk exposure are selected for safety tests.

Stallbaum et al. [29] use the test model first for generating unordered test case scenarios, and then for ordering these based on the risk information provided in the model. Risk prioritization may be based on the total risk score, i.e., the sum of the risks of all actions covered by the test case scenario, or by only considering risks not already covered by previously chosen test case scenarios.

Markov chains are used for the test case generation in [9], which also uses the prioritization algorithm of [29] to prioritize previously generated test suites.

Of the approaches in this section, that shown in only [29] is supported by a dedicated prototype tool, but all have been tested on examples taken from real systems. Using a subset of IBM WebSphere Commerce 5.4, the approach in [7,8] is found to be more effective and cover more critical test cases with a slightly better case coverage. A significant increase in coverage is found in [9], where it is also reported that much effort was needed to convince various interest groups (e.g., software developers and management) of the benefits

Table 4 Approaches with focus on test-case analysis

Approach	Kind	Goal	Strategy
Chen et al. [7,8]	RT	Make regression testing more effective	Employs models expressed as UML activity diagrams to support regression testing. The risk analysis is conducted at the level of test cases to select those that are most relevant for critical tests
Entin et al. [9]	RT	Improve the effectiveness of testing	Presents experiences and challenges related to the use of approaches for risk-based test case generation in a SCRUM process
Stallbaum et al. [29]	RT	Early identification of critical faults and test case prioritization	Automatically generates system test cases from activity diagrams and prioritizes them based on risk

of model-based testing. The approach in [29] is evaluated on a real income tax calculation example, and the approach is found to enable early detection of critical faults.

4.5 Approaches based on automatic source code analysis

A different kind of approach is found in [14,32], which are based on automatic source code analysis (Table 5). [32] is one of the few approaches which focus on test-based risk analysis. Risk of code is described as the likelihood that a given function or block within source code contains a fault. A fault within source code may lead to execution failures, such as abnormal behavior or incorrect output. The risk model is updated based on metrics related to both the static structure of code as well as dynamic test coverage. A more complex static structure leads to higher risk, while more thoroughly tested code has less risk.

The approach in [14] is based on [32], but focusing on risk-based testing aiming at test prioritization and optimization. More structural observations are added to the analysis process in [32], such that, e.g., errors in loop conditions result in a higher risk for the affected code block.

Both [14] and [32] are supported by prototype tools for automating the source code analysis. Even though the approach in [32] has been tested on a real program, where the program faults were found to be located in the blocks and functions identified as high risk, the authors are careful to conclude with respect to the generalizability of the results.

4.6 Approaches targeting specific programming paradigms

The risk-based testing approach in [19] (Table 6) is targeted towards aspect-oriented programming (AOP), arguing that AOP solutions typically have an enterprise or platform level scope and, therefore, AOP errors have a much larger impact than errors in traditional localized code. The approach consists of a model for risk assessment, an associated fault model, and AOP testing patterns. The risk model is intended to be used for high-level business decisions to determine the role of AOP in the enterprise or in a particular project. Using the fault model and the test framework, test plans and test cases

Table 5 Approaches based on automatic source code analysis

Approach	Kind	Goal	Strategy
Hosseingholizadeh [14]	RT	Improve source-code risk-based testing by considering also the size and testability of software components	Estimates the maximal risk that can be eliminated over time and use this to prioritize testing
Wong et al. [32]	TR	Estimate risk level without human estimation	Estimates code risks based on the static structure of the source code or the dynamic test coverage of the code

Table 6 Approaches targeting specific programming paradigms

Approach	Kind	Goal	Strategy
Kumar et al. [19]	RT	Improve testing within aspect-oriented programming	A fault model which defines typical faults that occur with aspect-oriented programming and the risks associated with the faults are used to select test cases
Rosenberg et al. [25]	RT	Make testing more effective	Identifies the risk of object-oriented classes

can be created to mitigate risk identified using the model. The framework supports both white-box and black-box testing, using unit and regression testing. The existing AOP testing patterns are targeted towards service-oriented architectures and enterprise architecture solutions.

A more low-level approach is found in [25], identifying the most important classes to test in an object-oriented program. The probability of failure of a portion of code (i.e., a class) is determined by its complexity. For this, [25] uses six metrics identified by the Software Assurance Technology Center at NASA Goddard Space Flight Center. These metrics include both internal complexity (e.g., number of methods in the class and minimum number of test cases needed for each method), the structural position of the class (e.g., depth in inheritance tree and number of immediate subclasses), and relationship to other classes (e.g., coupling and the number of methods that can be invoked from the outside). For all these metrics, threshold values are defined, and a class is said to have high risk if at least two of the metrics exceed the recommended limits. If further prioritization is needed, it is up to the project to determine the criticality of each of the high-risk classes.

The approach in [19] is said to have been used with reasonably large customers, but this is not described in the paper. Empirical assessment is not included in [25] either. Neither approach seems to be supported by a dedicated tool.

4.7 Approaches targeting specific applications

Bai and Kenett [3,4] show an approach for risk-based group testing of semantic web services (Table 7). Test cases are grouped according to their risk level, and the groups with highest risk are tested first. Services that do not meet a defined threshold are ruled out. In this way, a large number of unreliable web services are removed early in the testing

Table 7 Approaches targeting specific applications

Approach	Kind	Goal	Strategy
Bai et al. [3,4]	RT	Reduce test cost and improve test effectiveness	Risk-based test selection and prioritization of test cases with respect to web-services reliability
Casado et al. [5,6]	RT	Improve the effectiveness of web service transaction testing	The web-services are represented by a formal model from which fault trees are constructed. The fault trees are used to make test cases
Murthy et al. [20]	RT	Reduce time and control costs	Integrates risk-based testing principles into application security testing. Employs risk analysis in a straightforward manner to prioritize
Zech et al. [35,36]	RT	Improve effectiveness of security testing for cloud computing	Model-based and making use of risk analysis to test the security of a cloud computing environment among all layers. Test suites are target language and target platform independent

process, reducing the total number of executed tests. In the risk analysis, failure probability and importance are analyzed from three aspects: ontology data, service, and composite service. In addition, a runtime monitoring mechanism is used to detect dynamic changes in the web services and adjust the risk accordingly. The approach is evaluated in an experiment using the BookFinder OWL ontology to compare risk-based testing with random testing. The experiment concluded that the risk-based testing approach greatly reduced the test cost and improved test efficiency.

Web services are also the focus in the framework in [5,6], where the aim is to test the advanced, long-lived transactions used in web services. The conceptual framework is hierarchically organized into four levels, consisting of (1) a method to define functional transactional requirements, (2) division into subsystems of transaction system properties, (3) applying risk analysis for predefined properties of web services transactions, and (4) applying testing techniques to generate test scenarios and mitigate risks. The most developed part of the framework is the use of fault tree analysis to perform risk analysis for the recovery property. In [5], this analysis is exemplified using a travel agency example, but no empirical evaluation is reported.

The approach in [20] focuses on how to introduce risk-based testing principles into application security testing. The approach combines best practices from various areas like NIST and OWASP, and uses threat modeling from which security test scenarios and test cases are derived. The paper reports on a case study comparing traditional testing with risk-based testing on a gaming application. Risk-based testing was found to provide savings in time, cost, and resource usage.

Security testing is also the focus in [35,36], targeting especially cloud computing environments. This is a model-driven approach to risk-based testing, emphasizing the negative perspective where the main goal is not to assure system validity but rather to show its deficiency. Model-to-model transformations are used from a system model to a risk model and then to a misuse case model. In addition, testing is performed by runtime model execution, instead of the more traditional approach of generating executable test code from the models. No empirical evaluation is reported.

Of all the approaches in this section, that shown in only [35,36] is supported by a dedicated tool, developed as an Eclipse plug-in. Tool support is hardly mentioned in the other approaches.

4.8 Approaches aiming at measurement

Of all the approaches in this survey, that shown in [26] is the only one explicitly supporting both risk-based testing and test-based risk analysis (Table 8). With a focus on reliability, the paper describes a risk-driven reliability model and testing process where the risk of software failure is used to drive test scenarios and reliability predictions. Both consumer and producer risks are considered. In addition to comparing empirical values of risk and reliability to specified threshold values, emphasis is placed on evaluating also the model that predicts risk and reliability.

Schneidewind [26] does not seem to be supported by a dedicated tool, but the approach is applied to a real application involving the NASA space shuttle flight software. For this application, all tests are passed with the conclusion that this safety critical software has been accepted. The predicted consumer reliability is compared to the actual reliability, and also to predicted reliability using another approach (Yamada S Shaped Model), with favorable results for the approach in [26].

The approach in [27,28] defines a set of risk-based testing metrics to control and measure (i) the risk-based test cases, (ii) the risk-based testing activities, and (iii) the impact and advantages of using risk-based testing in an organization. The risk-based testing process, together with the proposed metrics, is tested in a real case study involving developers of an Eclipse-based tool environment for automatic development of simulators. The results are positive, concluding that for this application, the proposed approach was able to find the most important defects earlier than a more traditional functional approach to testing, thus saving costs. The case study also demonstrated the need for better tools, and the authors report to be working on such a tool, supporting in particular risk management which test engineers find more difficult to perform than testing.

Table 8 Approaches aiming at measurement

Approach	Kind	Goal	Strategy
Schneidewind [26]	RT TR	Make testing more effectives and predict reliability	Uses the process of risk of software failure to derive reliability predictions
Souza et al. [27,28]	RT	Make testing more effective and measure the effectiveness of testing	Risk analysis followed by testing

5 Maturity level for all approaches

This section characterizes the maturity level for each of the 24 approaches. The maturity level is decomposed into three aspects, namely the degree of formalization, to what extent there is empirical evaluation and whether the approach is supported by a tool, thus forming the basis for answering RQ4. The approaches are organized and presented according to the same eight categories as in Sect. 4.

5.1 Approaches addressing combination at a general level

Table 9 describes the maturity level of the four approaches addressing combination of risk analysis and testing at a general level.

5.2 Approaches with focus on model-based risk estimation

Table 10 describes the maturity level of the two approaches with main focus on model-based risk estimation.

5.3 Approaches with focus on test-case generation

Table 11 describes the maturity level of the five approaches with main focus on test-case generation.

5.4 Approaches with focus on test-case analysis

Table 12 describes the maturity level of the three approaches with main focus on test-case analysis.

Table 9 Maturity level of the approaches addressing combination at a general level

Approach	Formalization	Evaluation	Tool
Amland [2]	Risks are estimated and evaluated by making use of a mathematical model combined with a table based technique	An industrial case study on a retail banking application	No
Felderer et al. [10,11]	Risks are estimated and evaluated by making use of a mathematical model combined with a table based technique	Applied on a web application, in an anonymized industrial project	No
Redmill [23,24]	Risks are identified, estimated and evaluated by making use of a table based technique	No	No
Yoon and Choi [34]	Test cases are prioritized with respect to their weight value, calculated using a mathematical model that considers risk exposure values and correlation values between test cases and risks	Applied on an aircraft collision avoidance system	No

Table 10 Maturity level of the approaches with focus on model-based risk estimation

Approach	Formalization	Evaluation	Tool
Gleirscher [12,13]	Hazards are identified by making use of hazard guide words or fault tree analysis, and then assessed with respect to severity, probability, exposure and detectability	An excerpt from an anonymized case study on commercial road vehicle safety	No
Ray and Mohapatra [22]	Risks are identified by making use of functional failure analysis, fault tree analysis, and software failure mode and effect analysis. Risks are estimated and evaluated by making use of mathematical models	One experiment for cross checking estimated risks with failures observed, and one experiment in which the approach is compare to two related approaches	No

Table 11 Maturity level of the approaches with focus on test-case generation

Approach	Formalization	Evaluation	Tool
Kloos et al. [18]	Safety risks are identified by making use of fault tree analysis. Test cases are derived from state machine diagrams that are constructed from fault trees and state machines representing system specification	Applied on a control software component example of a modular production system	No
Nazier and Bauer [21]	Safety risks are identified by making use of fault tree analysis. The risk information in fault trees is used to update state charts representing system specification. Test cases are traces in the updated state charts	Applied on a gas burner system example	Prototype tool
Wendland et al. [31]	Risks are evaluated with respect to a risk evaluation matrix. Test cases are said to be derived from behavior models with respect to identified test directives for each risk	No	Unclear
Xu et al. [33]	Test cases are generated from formal threat models	Two case studies; one on Magento and one on FileZilla Server	No
Zimmermann et al. [37]	Risks are identified by making use of failure mode and effects analysis or fault tree analysis. Test cases are paths (from initial state to final state) in Markov chains and are identified with respect to the information provided by the failure mode and effects analysis or fault tree analysis	Two examples: an alarm system example and a railway control system example	No

Table 12 Maturity level of the approaches with focus on test-case analysis

Approach	Formalization	Evaluation	Tool
Chen et al. [7,8]	Risks are estimated and evaluated by making use of a mathematical model combined with a table based technique. Test cases are paths in activity diagrams starting at the initial node and ending at the final node	Applied on three components of IBM WebSphere Commerce 5.4	No
Entin et al. [9]	Refers to Stallbaum [29] for prioritizing test cases with respect to risk analysis. Test cases are paths in state machine diagrams, representing system aspects, starting at the initial node and ending at the final node	Applied on graphical user interfaces for power system testing solutions developed by Omicron	No
Stallbaum et al. [29]	Risks are estimated by multiplying the probability that an entity contains fault with the associated damage. Test cases are paths in activity diagrams starting at the initial node and ending at the final node	A practical example based on the flow chart for calculating income tax, provided by the German Federal Ministry of Finance	Prototype tool for the RiteDAP technique

5.5 Approaches based on automatic source code analysis

Table 13 describes the maturity level of the two approaches classified as automatic source code analysis.

5.6 Approaches targeting specific programming paradigms

Table 14 describes the maturity level of the two approaches classified as targeting specific programming paradigms.

5.7 Approaches targeting specific applications

Table 15 describes the maturity level of the four approaches classified as targeting specific applications.

5.8 Approaches aiming at measurement

Table 16 describes the maturity level of the two approaches classified as aiming at measurement.

6 Relationships between publications

In addition to studying the characteristics of the approaches for RT and TR, we have also analyzed the relationships between the approaches by addressing to what extent the papers refer to each other. This gives useful insight into how the combination of testing and risk analysis has grown and developed as a research field. Hence, this section forms the basis for answering RQ5.

The first observation is that only seven of the 24 approaches (nine of the 32 papers) are from before 2009, demonstrating an increasing interest in the field.

Secondly, for five of the approaches (see Table 17), their publications are all isolated from the other TR and RT publications, as they have neither citations nor references within the group of papers included in this survey.

For the remaining papers, their relationships are illustrated in Fig. 2. Here, each arrow represents the fact that

Table 13 Maturity level of the approaches based on automatic source code analysis

Approach	Formalization	Evaluation	Tool
Hosseingholizadeh [14]	The risk of certain source code causing problems is estimated and evaluated with respect to an adjusted mathematical model presented by Wong et al. [32]	Applied on software in an anonymized sample project	Prototype tool for analyzing source code
Wong et al. [32]	The risk of certain source code causing problems is estimated and evaluated by making use of a mathematical model, taking five source-code metrics into account. The authors refer to the usage of graphs, similar to activity diagrams, for generating test cases	Applied on SPACE—an industrial software used for configuring antennas	Prototype tool: χ-RISK

Table 14 Maturity level of the approaches targeting specific programming paradigms

Approach	Formalization	Evaluation	Tool
Kumar et al. [19]	Risks are identified by mapping a given risk to a list of general faults within aspect oriented programming. Each fault has its own qualitative risk level. Risks are estimated by adding qualitative risk levels to obtain an aggregated risk level for a given risk	A financial organization example handling credit card transactions	No
Rosenberg et al. [25]	Risks are estimated by making use of a mathematical model. The likelihood of component failure is of particular focus and is based on a complexity analysis of source code	No	No

Table 15 Maturity level of the approaches targeting specific applications

Approach	Formalization	Evaluation	Tool
Bai et al. [3,4]	Risks are estimated and evaluated by making use of a mathematical model	A web service example used to search for books	No
Casado et al. [5,6]	Risks are identified by making use of fault tree analysis. Test cases are sequences of transactions derived from state diagrams with respect to information provided by the fault tree analysis	A travel agency example	No
Murthy et al. [20]	The authors refer to the usage of a risk analysis approach suggested by NIST and threat modeling/misuse case modeling for risk analysis and test case derivation, respectively	High level results from an experiment on a gaming application	No
Zech et al. [35,36]	Security risks are identified by matching attack patterns on the public interfaces of a system under test. The estimation and prioritization of risks are based on a complexity factor for specific operations in the system under test. Misuse cases are used as a basis for deriving test cases	Applied on a small excerpt of the interface of VMware's vCloud Director	Eclipse plugin

Table 16 Maturity level of the approaches aiming at measurement

Approach	Formalization	Evaluation	Tool
Schneidewind [26]	Risks are estimated and evaluated by making use of mathematical and statistical models.	NASA space shuttle application example problem	No
Souza et al. [27,28]	The authors refer to the usage of a taxonomy based questionnaire for identifying risks. Risks are estimated and evaluated by making use of a table based technique	Case study on the Multi-Physics and Multi-Scales Solver Environment	No

Table 17 Publications that have neither citations nor references within the group of papers included in this survey

Refernces	Publication year
Schneidewind [26]	2007
Murthy et al. [20]	2009
Casado et al. [5,6]	2010–2011
Gleirscher [12,13]	2011–2013
Xu et al. [33]	2012

the approach at the source end of the arrow is referenced by the approach at the target end. As we are mainly interested in possible paths of influence, we have not shown arrows to previous publications/approaches, when a more recent publication/approach is also referenced. E.g., Stallbaum et al. [29] references both Amland [2] and Chen et al. [7,8], but only the last relationship is shown in the figure.

As shown in Fig. 2, the paper by Amland from year 2000 [2] is the only one that is referenced (directly or indirectly) by almost all later approaches. For the rather recent publications by Zech et al. [35,36] and Ray et al. [22], Amland [2] is in fact their only reference to other publications in this survey. Conversely, it should also be noted that in addition to the publications in the previous table, neither Kumar et al. [19], Souza et al. [27,28], nor Bai et al. [3,4] is cited by any later paper, although their first publication appeared in 2009. All in all, we conclude that there seems to be little common ground for the different approaches.

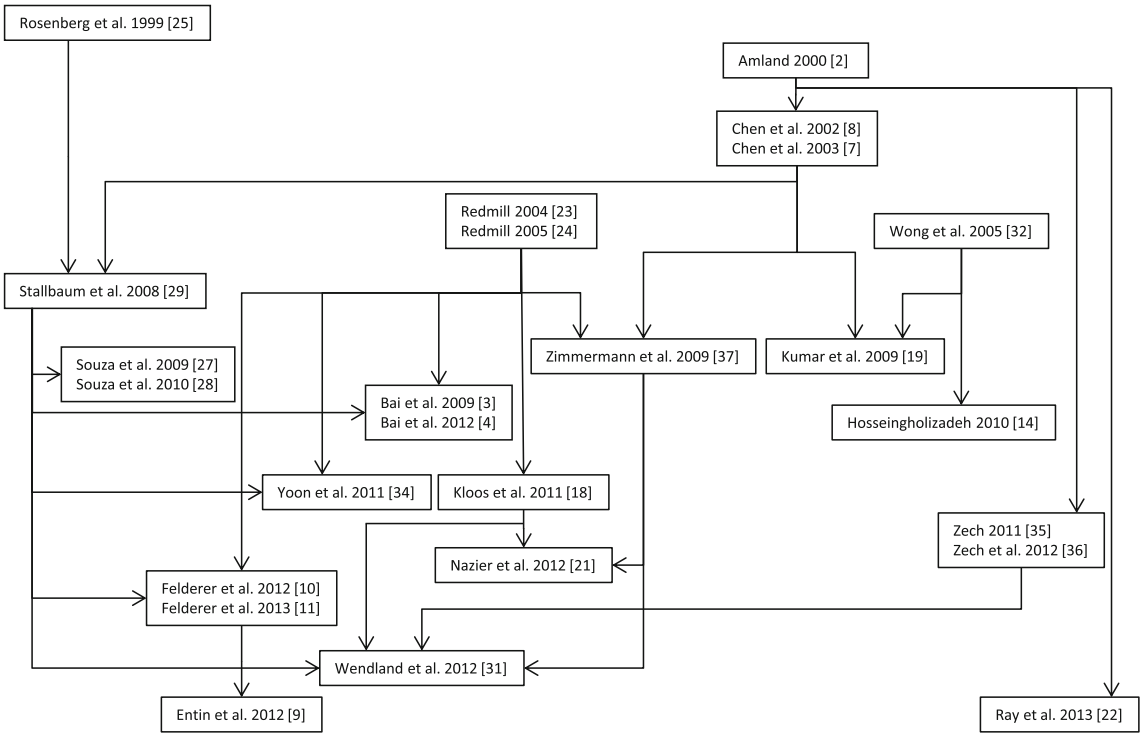

Fig. 2 Relationships between publications

7 Discussion

Based on the summaries and classifications provided by Sects. 4, 5 and 6, the research questions RQ1-RQ5 are now discussed.

RQ1 What approaches exist for performing risk-based testing (RT) and test-based risk analysis (TR)? We have identified a total of 24 approaches, of which only two address TR while the remaining 22 focus solely on various aspects of RT. Of the 24 approaches, four address RT at a general level, two RT approaches have model-based risk estimation as their main issue, five have particular focus on test-case generation, three mainly focus on test-case analysis, one RT and one TR approach address source code analysis, two RT approaches address specific programming paradigms while four target specific applications and one RT and one RT/TR approach aim at measurement in particular. An overview of each of the approaches is given in Sect. 4.

RQ2 For each of the identified approaches, what is the main goal, and what strategies are used to achieve that goal? For the two TR approaches, the goal is to automate risk level estimation [32] and predict reliability [26], respectively. Their strategy is rather different, with [32] using source code analysis while [26] is based on empirical measurement. For the RT approaches, recurring goals are to improve the effectiveness of testing and/or to reduce time and costs. The various strategies used to achieve this are summarized in Sect. 4. Safety and security are of special concern in some of the approaches, in particular those with a focus on test-case generation. In 13 out of the 24 approaches, risk analysis is used only to plan the testing process. Of the 11 approaches that address test design/derivation, only Wendland et al. [31] point out the lack of systematic methods for identifying test cases by making use of the risk analysis.

RQ3 Are there contexts in which TR and RT are considered to be particularly useful?
Most of the approaches are general and not intended to be used in any particular context. For the approaches targeting specific applications, web services/ applications and cloud computing is a common theme, which is also used as examples in [7,8,10,11,33].

RQ4 How mature are the approaches, when considering degree of formalization, empirical evaluation, and tool support?
The level of formality and preciseness varies quite a lot. At the most informal level, risks are assigned qualitative risk levels without conducting initial steps for identifying and estimating risks (e.g., [19]). On the

other hand, at the most formal level, risks are estimated and evaluated using rigorous mathematical and statistical models (e.g., [26]). Moreover, it is worth noting that most approaches explain the process of risk assessment (i.e., risk identification, risk estimation and risk evaluation) either only partly or very briefly. Three notable exceptions are [10,24,28]. For example, in some approaches there are no clear explanations for how risks are identified, while in other approaches there are no clear explanations for how risks are estimated and evaluated. Most of the papers focus on the methodological side of the problem. With respect to dedicated tool support, only one approach presents a complete tool for automatically generating test cases [35,36], while four approaches are supported by a prototype tool [14,21,29,32]. Most of the papers provide little evidence regarding the usefulness of the proposed approach. Some notable exceptions are reported experiences and evaluations from an industrial case study [2] and industrial projects [8,9,11,26,32].

RQ5 What are the relationships between the approaches when considering citations between the publications? The relationships are illustrated in Fig. 2. From the analysis, there seems to be no established set of works that has formed the basis for subsequent research. It is also not possible to identify sets of publications representing different communities ("schools of thought"). Instead, it seems that many of the approaches have been developed in isolation, and with little impact on the field so far.

8 Conclusions

Although the processes for risk analysis and testing are different and carried out for different purposes, they are nevertheless related and in practice often combined. In the systematic literature review, on which this paper reports, we have studied and classified 32 scientific papers focusing on the combined use of risk analysis and testing. As we have argued, there are basically two main strategies for the combined use of risk analysis and testing, namely test-based risk analysis (TR) and risk-based testing (RT). In the first case, testing is used to support the risk analysis process while risk analysis supports testing in the second case. Interestingly, we have found that only two of the 32 papers can be said to address TR, while the remaining 30 focus solely on various aspects of RT.

Within the field addressed by this survey, there is clearly need for further research. In general, the field needs more formality and preciseness as well as dedicated tool-support. In particular, there is very little empirical evidence regarding the usefulness of the various approaches.

Acknowledgments This work has been conducted as a part of the DIAMONDS (201579/S10) project funded by the Research Council of Norway, as well as a part of the NESSoS network of excellence and the RASEN project funded by the European Commission within the 7th Framework Programme.

Appendix: Search strings

The following 23 search strings were used during Step 1 of the search process described in Sect. 3. Initially, we only had search strings 1 and 14, with boolean operators for combining the possible search terms in different ways. However, the initial search revealed the need for including also each possible combination as a search string in itself.

- Search string 1: (risk) AND (based OR driven OR oriented) AND (test OR testing OR verification OR checking)
- Search string 2: risk based testing
- Search string 3: risk driven testing
- Search string 4: risk oriented testing
- Search string 5: risk based test
- Search string 6: risk driven test
- Search string 7: risk oriented test
- Search string 8: risk based verification
- Search string 9: risk driven verification
- Search string 10: risk oriented verification
- Search string 11: risk based checking
- Search string 12: risk driven checking
- Search string 13: risk oriented checking
- Search string 14: (test) AND (based OR driven OR oriented) AND (risk) AND (analysis OR assessment OR evaluation)
- Search string 15: test based risk analysis
- Search string 16: test driven risk analysis
- Search string 17: test oriented risk analysis
- Search string 18: test based risk assessment
- Search string 19: test driven risk assessment
- Search string 20: test oriented risk assessment
- Search string 21: test based risk evaluation
- Search string 22: test driven risk evaluation
- Search string 23: test oriented risk evaluation

References

1. Alam, M., Khan, A.I.: Risk-based testing techniques: a perspective study. Int. J. Comput. Appl. **65**, 33–41 (2013)
2. Amland, S.: Risk-based testing: risk analysis fundamentals and metrics for software testing including a financial application case study. J. Syst. Softw. **53**, 287–295 (2000)

3. Bai, X., Kenett, R.S.: Risk-based adaptive group testing of semantic web services. In: Proceeding of the 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC'09), vol. 2, pp. 485–490. IEEE, New York (2009)
4. Bai, X., Kennett, R.S., Yu, W.: Risk assessment and adaptive group testing of semantic web services. Int. J. Softw. Eng. Knowl. Eng. 595–620 (2012)
5. Casado, R., Tuya, J., Younas, M.: Testing long-lived web services transactions using a risk-based approach. In: Proceeding of the 10th International Conference on Quality Software (QSIC'10), pp. 337–340. IEEE, New York (2010)
6. Casado, R., Tuya, J., Younas, M.: A framework to test advanced web services transactions. In: Proceeding of the 4th International Conference on Software Testing, Verification and Validation (ICST'11), pp 443–446. IEEE, New York (2011)
7. Chen, Y., Probert, R.L.: A risk-based regression test selection strategy. In: Proceeding of the 14th IEEE International Symposium on Software Reliability Engineering (ISSRE'03), Fast Abstract, pp. 305–306. Chillarege Press (2003)
8. Chen, Y., Probert, R.L., Sims, D.P.: Specification-based regression test selection with risk analysis. In: Proceeding of the 2002 Conference of the Centre for Advanced Studies on Collaborative Research (CASCON'02), pages 1–14. IBM Press, USA (2002)
9. Entin, V., Winder, M., Zhang, B., Christmann, S.: Introducing model-based testing in an industrial scrum project. In: Proceeding of the Seventh International Workshop on Automation of Software Test (AST'12), pp. 43–49. IEEE, New York (2012)
10. Felderer, M., Haisjackl, C., Breu, R., Motz, J.: Integrating manual and automatic risk assessment for risk-based testing. In: Proceeding of the 4th International Conference on Software Quality (SWQD'12), vol. 94 of LNBIP, pp. 159–180. Springer, Berlin (2012)
11. Felderer, M., Ramler, R.: Experiences and challenges of introducing risk-based testing in an industrial project. In: Proceeding of the Fifth International Conference on Software Quality (SWQD'13), vol. 133 of LNBIP, pp. 10–29. Springer, New York (2013)
12. Gleirscher, M.: Hazard-based selection of test cases. In: Proceeding if the Sixth International Workshop on Automation of Software Test (AST'11), pp. 64–70. ACM, New York (2011)
13. Gleirscher, M.: Hazard analysis of technical systems. In: Proceeding of the Fifth International Conference on Software Quality (SWQD'13), vol. 133 of LNBIP, pp. 104–124. Springer, Berlin (2013)
14. Hosseingholizadeh, A.: A source-based risk analysis approach for software test optimization. In: Proceeding of the Second International Conference on Computer Engineering and Technology (ICCET'10), vol. 2, pp. 601–604. IEEE, New York (2010)
15. International Standards Organization. ISO 31000:2009(E), Risk management—Principles and guidelines (2009)
16. International Standards Organization. ISO 29119 Software and system engineering—Software Testing-Part 2 : Test process (draft) (2012)
17. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007–001 (2007)
18. Kloos, J., Hussain, T., Eschbach, R.: Risk-based testing of safety-critical embedded systems driven by Fault Tree Analysis. In: Proceeding of the Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW'11), pp. 26–33. IEEE, New York (2011)
19. Kumar, N., Sosale, D., Konuganti, S.N., Rathi, A.: Enabling the adoption of aspects-testing aspects: A risk model, fault model and patterns. In: Proceeding of the Eighth ACM International Conference on Aspect-Oriented Software Development (AOSD'09), pp. 197–206. ACM, New York (2009)
20. Murthy, K.K., Thakkar, K.R., Laxminarayan, S.: Leveraging risk based testing in enterprise systems security validation. In: Proceeding of the First International Conference on Emerging Network Intelligence (EMERGING'09), pp. 111–116. IEEE, New York (2009)
21. Nazier, R., Bauer, T.: Automated risk-based testing by integrating safety analysis information into system behavior models. In: Proceeding of the 23rd International Symposium on Software Reliability Engineering Workshops (ISSREW'12), pp 213–218. IEEE, New York (2012)
22. Ray, M., Mohapatra, D.P.: Risk analysis: a guiding force in the improvement of testing. IET Softw. **7**, 29–46 (2013)
23. Redmill, F.: Exploring risk-based testing and its implications. Softw. Test. Verif. Reliab. **14**, 3–15 (2004)
24. Redmill, F.: Theory and practice of risk-based testing. Softw. Test. Verif. Reliab. **15**, 3–20 (2005)
25. Rosenberg, L., Stapko, R., Gallo, A.: Risk-based object oriented testing. In: Proceeding of the 24th Annual Software Engineering Workshop, pp. 1–6. NASA, Software Engineering Laboratory, (1999)
26. Schneidewind, N.F.: Risk-driven software testing and reliability. Int. J. Reliab. Qual. Saf. Eng. **14**, 99–132 (2007)
27. Souza, E., Gusmão, C., Alves, K., Venâncio, J., Melo, R.: Measurement and control for risk-based test cases and activities. In: Proceeding of the 10th Latin American Test Workshop (LATW'09), pp 1–6. IEEE, New York (2009)
28. Souza, E., Gusmão, C., Venâncio, J.: Risk-based testing: A case study. In: Proceeding of the Seventh International Conference on Information Technology: New Generations (ITNG'10), pp. 1032–1037. IEEE, New York (2010)
29. Stallbaum, H., Metzger, A., Pohl, K.: An automated technique for risk-based test case generation and prioritization. In: Proceeding of the Third International Workshop on Automation of Software Test (AST'08), pp. 67–70. ACM, New York (2008)
30. Sulaman, S.M., Weyns, K., Höst, M.: A review of research on risk analysis methods for it systems. In: Proceeding of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE'13), pp. 86–96 (2013)
31. Wendland, M.-F., Kranz, M., Schieferdecker, I.: A systematic approach to risk-based testing using risk-annotated requirements models. In: Proceeding of the Seventh International Conference on Software Engineering Advances (ICSEA'12), pp. 636–642. IARA (2012)
32. Wong, W.E., Qi, Y., Cooper, K.: Source code-based software risk assessing. In: Proceeding of the 2005 ACM Symposium on Applied Computing (SAC'05), pp. 1485–1490. ACM, New York (2005)
33. Xu, D., Tu, M., Sandford, M., Thomas, L., Woodraska, D., Xu, W.: Automated security test generation with formal threat models. IEEE Trans. Dependable Secure Comput. **9**, 526–540 (2012)
34. Yoon, H., Choi, B.: A test case prioritization based on degree of risk exposure and its empirical study. Int. J. Softw. Eng. Knowl. Eng. **21**, 191–209 (2011)
35. Zech, P.: Risk-based security testing in cloud computing environments. In: Proceeding of the Fourth International Conference on Software Testing, Verification and Validation (ICST'11), pp. 411–414. IEEE, New York (2011)
36. Zech, P., Felderer, M., Breu, R.: Towards a model based security testing approach of cloud computing environments. In: Proceeding of the Sixth International Conference on Software Security and Reliability Companion (SERE-C'12), pp. 47–56. IEEE, New York (2012)
37. Zimmermann, F., Eschbach, R., Kloos, J., Bauer, T.: Risk-based statistical testing: A refinement-based appraoch to the reliability analysis of safety-critical systems. In: Proceeding of the 12th European Workshop on Dependable Computing (EWDC'09), pp. 1–8 (2009)