Network of Excellence

Deliverable D10.4

Enhanced Methods for Risk and Cost Aware SDLC
Abstract

The main objective of NESSoS WP10 is to develop a framework to facilitate a risk and cost aware software development life cycle (SDLC). This deliverable reports on the main results achieved in this work package during the third year of the project. The results build on previous WP10 work and show extensions and further developments of our portfolio of methods and techniques to support the process for maintaining a risk and cost aware SDLC that we presented in NESSoS D10.2 and D10.3.

The contributions of this deliverable can be categorized into four main areas, each of which is a focus area of WP10: i) Methods for information security risk and cost assessment of ICT and cloud service systems; ii) Methods and techniques for security risk management of evolving systems; iii) Security risk assessment leveraging on security assurance; iv) Risk assessment from a formal methods perspective. The WP10 research tasks that have been addressed during the third year are (T10.2) to further develop our methodology for risk and cost aware SDLC, (T10.3) techniques for run-time risk management, and (T10.4) integration of the risk and cost aware methodology.
Keyword List

*Future Internet, security, risk, cost, software development life cycle, cloud services, security testing*
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Type of Change</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Table of Contents</td>
<td>Bjørnar Solhaug (SINTEF)</td>
</tr>
<tr>
<td></td>
<td>First full draft</td>
<td>Kristian Beckers (UDE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jorge Cuellar (Siemens)</td>
</tr>
<tr>
<td>1.0</td>
<td></td>
<td>Leanid Krautsevich (CNR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Aliaksandr Lazouski (CNR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dawid Machinicki (Atos)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fabio Martinelli (CNR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Aljosa Pasic (Atos)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jaime Martin Perez (Atos)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fredrik Seehusen (SINTEF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bjørnar Solhaug (SINTEF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jan Stijohann (Siemens)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ketil Stølen (SINTEF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Le Minh Sang Tran (UNITN)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Artsiom Yautsiukhin (CNR)</td>
</tr>
<tr>
<td>2.0</td>
<td>Review ready</td>
<td>Kristian Beckers (UDE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Leanid Krautsevich (CNR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Aljosa Pasic (Atos)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jaime Martin Perez (Atos)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bjørnar Solhaug (SINTEF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jan Stijohann (Siemens)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Le Minh Sang Tran (UNITN)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kristian Beckers (UDE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bjørnar Solhaug (SINTEF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jan Stijohann (Siemens)</td>
</tr>
<tr>
<td>3.0</td>
<td>Revision after review</td>
<td>Le Minh Sang Tran (UNITN)</td>
</tr>
<tr>
<td>3.1</td>
<td>Final version approved by reviewers</td>
<td>Le Minh Sang Tran (UNITN)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bjørnar Solhaug (SINTEF)</td>
</tr>
</tbody>
</table>
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table of Contents</strong></td>
<td>5</td>
</tr>
<tr>
<td><strong>List of Figures</strong></td>
<td>11</td>
</tr>
<tr>
<td><strong>List of Tables</strong></td>
<td>13</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>17</td>
</tr>
<tr>
<td><strong>2 ISMS-CORAS: Method for Establishing Information Security Management Systems</strong></td>
<td>19</td>
</tr>
<tr>
<td>2.1 Background</td>
<td>19</td>
</tr>
<tr>
<td>2.1.1 CORAS</td>
<td>20</td>
</tr>
<tr>
<td>2.1.2 Legal CORAS</td>
<td>20</td>
</tr>
<tr>
<td>2.1.3 ISO 27001</td>
<td>20</td>
</tr>
<tr>
<td>2.2 The ISMS-CORAS Method</td>
<td>21</td>
</tr>
<tr>
<td>2.3 ISMS-CORAS Example Artifacts</td>
<td>23</td>
</tr>
<tr>
<td>2.4 Conclusion</td>
<td>25</td>
</tr>
<tr>
<td><strong>3 Cloud Service Risk and Cost-Benefit Assessment</strong></td>
<td>27</td>
</tr>
<tr>
<td>3.1 Concepts and Relations Encompassed by the Ontology</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Cloud Security Risk Assessment</td>
<td>28</td>
</tr>
<tr>
<td>3.3 Integration of CBA and Cloud Security Risk Assessment</td>
<td>29</td>
</tr>
<tr>
<td>3.4 Conclusion and Future Steps</td>
<td>32</td>
</tr>
<tr>
<td><strong>4 Methods for Selection of Risk Treatments</strong></td>
<td>35</td>
</tr>
<tr>
<td>4.1 A Method to Early Deal with Evolution in Risk Assessment</td>
<td>35</td>
</tr>
<tr>
<td>4.1.1 Step 1 – Identify Evolving Context</td>
<td>36</td>
</tr>
<tr>
<td>4.1.2 Step 2 – Perform Risk Assessment</td>
<td>37</td>
</tr>
<tr>
<td>4.1.3 Step 3 – Model Context Evolution</td>
<td>37</td>
</tr>
<tr>
<td>4.1.4 Step 4 – Perform Evolution Analysis</td>
<td>38</td>
</tr>
<tr>
<td>4.2 Example</td>
<td>39</td>
</tr>
<tr>
<td>4.3 Conclusion</td>
<td>42</td>
</tr>
<tr>
<td><strong>5 Software Application Security Risk Assessment: Lessons Learned from Practical Applications</strong></td>
<td>43</td>
</tr>
<tr>
<td>5.1 Strict Time Requirements and Scalability are Critical Acceptance Factors</td>
<td>43</td>
</tr>
<tr>
<td>5.1.1 Simplifying the Security Overview</td>
<td>44</td>
</tr>
<tr>
<td>5.2 Traceability is not Sufficient for Industry Acceptance</td>
<td>44</td>
</tr>
<tr>
<td>5.3 The Security Overview Should not Reduce to a DFD</td>
<td>45</td>
</tr>
<tr>
<td>5.4 Manual Generation of Digital DFDs Must be Quick and Comfortable</td>
<td>45</td>
</tr>
<tr>
<td>5.5 Target Systems Scope Needs to be Narrowed Down</td>
<td>45</td>
</tr>
<tr>
<td>5.5.1 Operating System (OS)</td>
<td>45</td>
</tr>
<tr>
<td>5.5.2 Processor Platform</td>
<td>46</td>
</tr>
<tr>
<td>5.5.3 Managed or Unmanaged Code</td>
<td>46</td>
</tr>
<tr>
<td>5.5.4 Our Choice</td>
<td>46</td>
</tr>
<tr>
<td>5.5.5 Incremental Procedure: From Simple to More Complex SUTs</td>
<td>47</td>
</tr>
</tbody>
</table>
5.6 Risk-Assessments and Security Design Reviews can Profit from one Another .................. 47
5.6.1 Security Design Review .................................................. 47
5.6.2 Possible Synergies ......................................................... 48
5.6.3 Excursus: Testing for Design Flaws vs. Testing for Implementation Bugs .................. 48
5.7 Particularities of Web Applications in regard to the Proposed Method ......................... 48
5.8 Summary and Future Work ................................................... 49

6 TEST-DRIVEN SECURITY RISK ASSESSMENT ......................................... 51
6.1 Combining Testing and Risk Assessment ................................................. 51
6.1.1 Model-Based Security Testing .............................................. 51
6.1.2 Model-Based Security Risk Analysis ......................................... 52
6.1.3 Risk-Driven Security Testing .................................................. 52
6.1.4 Test-Driven Security Risk Assessment .......................................... 53
6.2 Example: Test-Driven Risk Assessment using CORAS ....................................... 53
6.2.1 Step 1: Establishing the Context .............................................. 54
6.2.2 Step 2: Risk Identification ..................................................... 54
6.2.3 Step 3: Risk Estimation .......................................................... 55
6.2.4 Step 4: Risk Evaluation .......................................................... 55
6.2.5 Step 5: Test Identification and Test Selection/Prioritization ....................... 56
6.2.6 Step 6: Test Design, Implementation and Execution .................................. 57
6.2.7 Step 7: Risk Validation and Treatment .......................................... 58
6.3 Conclusion and Ongoing Work .................................................................. 59

7 UCON POLICIES SPECIFICATION WITH RISK ....................................... 61
7.1 Attribute Mining Problem ............................................................. 61
7.1.1 Role Mining in RBAC ............................................................ 61
7.1.2 Attribute Mining in UCON ....................................................... 61
7.2 Policy Specification Problem .......................................................... 62
7.2.1 The Case of Single Attribute .................................................... 62
7.2.2 The Case of Several Attribute ................................................... 63
7.3 Conclusion ................................................................................. 64

8 FOUNDATIONS FOR COMPOSITIONAL SECURITY RISK ASSESSMENT .... 65
8.1 Composition and Compositionality ......................................................... 65
8.2 Representing the Target of Analysis ....................................................... 67
8.3 Representing the Risk Model ............................................................... 67
8.3.1 Risk Graph Syntax .................................................................... 67
8.3.2 Risk Graph Semantics ............................................................... 68
8.4 Target Composition ........................................................................... 68
8.4.1 Parallel Composition .................................................................... 69
8.4.2 Sequential Composition ................................................................ 69
8.4.3 Non-deterministic Choice ............................................................ 69
8.4.4 Features ..................................................................................... 70
8.4.5 Risk Graph Semantics for Target Composition .................................. 70
8.5 Conclusion and Ongoing Work ............................................................. 71
List of Figures

Figure 2.1: The ISMS-CORAS method (our contributions are marked in grey) ...................... 21
Figure 2.2: Attacker template ......................................................... 24
Figure 2.3: Network attacker overview diagram ........................................... 25
Figure 3.1: Ontology for integrated risk and cost-benefit assessment ......................... 28
Figure 3.2: Cost-benefit analysis ......................................................... 30
Figure 3.3: Total costs and benefits ....................................................... 32
Figure 3.4: Formal verification: Break-even points ....................................... 33
Figure 4.1: The conceptual model .......................................................... 36
Figure 4.2: The evolution perspectives of contexts ......................................... 37
Figure 4.3: The context evolution model ..................................................... 38
Figure 4.4: The architecture of the Messaging Service within SWIM .................... 40
Figure 4.5: The evolution of Messaging Service .......................................... 41
Figure 6.1: Risk-driven security testing ..................................................... 52
Figure 6.2: Test-driven security risk assessment ........................................... 53
Figure 6.3: Instance of the test-driven risk assessment process .......................... 54
Figure 6.4: Risk matrix with evaluation criteria .......................................... 55
Figure 6.5: CORAS threat diagram .......................................................... 56
Figure 7.1: Role mining in RBAC ............................................................ 61
Figure 7.2: Attribute mining in UCON ..................................................... 62
Figure 8.1: Risk graph ................................................................. 68
List of Tables

Table 2.1: ISMS-CORAS documentation .............................................................. 23
Table 4.1: The steps of the proposed approach. ...................................................... 36
Table 4.2: The max belief and defernt belief. ......................................................... 42
Table 6.1: Test scenarios extracted from threat diagrams ................................. 57
Table 6.2: Test scenarios prioritization ................................................................. 58
Table 6.3: Test result report .................................................................................. 58
Table 6.4: Revised risk assessment result .............................................................. 59
Executive Summary

The main objective of NESSoS WP10 is to develop a framework to facilitate a risk and cost aware software development life cycle (SDLC). This deliverable reports on the main results achieved in this work package during the third year of the project. The results build on previous WP10 work and show extensions and further developments of our portfolio of methods and techniques to support the process for maintaining a risk and cost aware SDLC that we presented in NESSoS D10.2 and D10.3. More specifically, the contributions of this deliverable are as follows.

Chapter 2 gives a presentation of ISMS-CORAS, which is a method developed to support the establishment and documentation of Information Security Management Systems (ISMS) compliant with the ISO/IEC 27001 standard. The method is defined as an extension of the CORAS method, which is a model-driven approach to risk analysis based on the ISO 31000 risk management standard. We show how ISMS-CORAS supports the creation of all required documentation, and how specific ISO/IEC 27001 demands are fulfilled, such as the assessment of legal aspects, the identification of vulnerabilities and attackers, and the systematic consideration of security controls. As part of the validation, we applied ISMS-CORAS to one of the NESSoS Smart Grid scenarios.

Chapter 3 presents an approach to combine security risk assessment and cost-benefit assessment of cloud services. The main challenge that is addressed is how to select cloud services that meet the service level agreements (SLAs) with acceptable cost and with an acceptable level of security risks. This challenge is very relevant for many kinds of cloud services, and the work presented in this chapter is in particular focused on the NESSoS eHealth scenario. In this scenario, cloud service providers are offering resources and services to other cloud users (such as medical staff and care-workers), based on agreed SLAs.

Chapter 4 presents a method for how to manage security risks of evolving software and ICT systems. More specifically, the challenge that is addressed is how to select the security risk treatments and security mechanisms that are most adequate when systems may change in different directions. The objective is to provide decision makers with decision support for ensuring so-called evolution-resilience of risk treatment alternatives. Evolution resilience should ensure that the security risks are kept at an acceptable level under system changes.

Chapter 5 presents a set of lessons learned and recommendations from the application of our method for software application security risk assessment. The method was presented in NESSoS D10.3 and is developed to support a process of continuous security risk assessment suited for iterative software development. The method makes use of both high-level business worst case scenarios and low-level security testing to facilitate the risk identification and estimation. The experiences from the application of the method are structured into a set of seven main lessons learned.

Chapter 6 addresses the systematic and methodological combination of security risk assessment and security testing. There are two main ways of combining the two. First, risk-driven security testing is the systematic use of risk assessment to support and facilitate the security testing. Second, test-driven security risk assessment is the systematic use of testing to support and facilitate the security risk assessment. In WP10 we use a model-based approach to testing and risk assessment and present in more details our approach to test-driven risk assessment using CORAS.

Chapter 7 presents an approach to tackle issues related to attribute mining and policy specification for usage control (UCON). In UCON, access decisions are made on the basis of rules and policies that consist of predicates over attributes. The approach presented in this chapter uses a risk-based approach where the rules should constrain the attribute values in such a way that the decisions of usage permissions never lead to unacceptable risks.

Chapter 8 presents our first steps towards a compositional approach to risk assessment, for which there is no or very limited support in the current state of the art. Compositionality should allow large-scale and complex software systems to be assessed by analyzing individual parts of the systems separately, and subsequently combining the results to soundly deduce the risk picture for the system as a whole. The chapter is rather formal as we are currently developing the formal foundation for compositional risk assessment. The foundation will serve as the basis for defining the compositional proof rules that can be applied by end-users.
1 Introduction

The overall objective of WP10 is to provide support for maintaining a risk and cost aware software development life cycle (RC-SDLC) that enables practitioners to instantiate an engineering process that optimizes value-for-money in terms of minimizing risks while keeping cost low and justified.

Based on the WP10 state of the art and gap-analysis reported in deliverable ID10.1 we defined our overall process for the RC-SDLC as presented in the previous deliverables D10.2 and D10.3. This process is defined to accommodate to an iterative SDLC and to conform to established standards on risk management [31] and information security management [35].

During the first three years of the project WP10 has continued to develop a portfolio of methods, techniques and tools to facilitate security risk and cost assessment in the SDLC. As explained in the previous deliverables, the challenge of supporting the whole SDLC with adequate techniques and tools for risk and cost assessment requires the problem to be tackled from different angles; no single approach, method or technique is likely to have the capacity to provide solutions to all the challenges of upholding an RC-SDLC. Hence, WP10 aims for a portfolio of complementary artifacts supporting different phases, activities or tasks of the overall process.

In this deliverable we report on the main results of NESSoS WP10 that were achieved during the third year of the project. The results can be categorized into four main areas, each of which of which is a main focus area of WP10: i) Methods for information security risk and cost assessment of ICT and cloud service systems; ii) Methods and techniques for security risk management of evolving systems; iii) Security risk assessment leveraging on security assurance; iv) Risk assessment from a formal methods perspective.

More specifically, the structure of this deliverable is as follows. Chapter 2 and Chapter 3 cover the first mentioned area. In particular, Chapter 2 presents ISMS-CORAS [10], which is an extension of the CORAS risk analysis method [41]. ISMS-CORAS provides support for establishing and documenting Information Security Management Systems (ISMS) compliant with the ISO/IEC 27001 [35] standard. The WP10 RC-SDLC is based on both this standard and on the ISO 31000 risk management standard [31], and because CORAS is based on the latter, ISMS-CORAS is strongly aligned with our process for maintaining an RC-SDLC. Chapter 3 focuses on cloud services and the combination of service risk and cost-benefit assessment. It assumes a risk assessment process compliant with ISO 31000 and deals with the problem of selecting cloud services that meet the service level agreements with an acceptable cost and at an acceptable level of risk.

Chapter 4 covers the second area of system evolution, and how select adequate means for mitigating security risks for systems that may evolve in different directions. This is an important challenge for dealing with security risks of Future Internet systems, and the work builds on our previous work on evolving risks [42, 61].

Chapter 5 and Chapter 6 cover the third area on leveraging on security assurance, specifically on security testing. In Chapter 5 we report on lessons learned from the practical application of our method for continuous security risk assessment during an iterative SDLC. The method was presented in D10.3 and utilizes security testing for supporting risk estimation. In Chapter 6 we discuss more generally the combinations of security risk assessment and security testing. We moreover give an overview of our recent work on test-driven security risk assessment. The objective in this approach is the converse of the method in Chapter 5. While the latter uses security tests to support the risk assessment, the former uses risk assessment to support the identification of security test cases.

Chapter 7 and Chapter 8 cover the fourth area of more formal approaches. Chapter 7 concerns usage control and proposes an approach to the specification of risk-based usage control policies. The aim of the work presented in Chapter 8 is to lay the formal foundation for compositional risk assessment. Such formal foundations are important tools for us as method developers to ensure and demonstrate the rigor and soundness of our methods.

Finally we summarize in Chapter 9 the collaborations in WP10 during the third year of the project, before concluding in Chapter 10.

Note that this deliverable only gives an overview of the third year results of WP10. For the full presentation of the methods, techniques and tools the reader is throughout the report referred to the relevant publications. An overview of third year WP10 publications is given in the appendix. All results are new and part of the NESSoS foreground, although some of the work has been conducted in affiliation.
with related project in which NESSoS WP10 partners are involved. For results that build on previous work we make explicit references in the deliverable sections in question.
2 ISMS-CORAS: Method for Establishing Information Security Management Systems

Fulfilling organizations’ security needs is a challenging task, but various security standards, such as ISO 27001 [35], offer support for attaining this goal. ISO 27001 prescribes a process for establishing and maintaining a so-called Information Security Management System (ISMS), which tailors security to the needs of any kind of organization. However, ambiguous descriptions in the standard may pose challenges during the establishment of an ISMS. Some of the ambiguity is deliberate to ensure that such standards can serve a multitude of different domains and stakeholders. This is nevertheless a problem for security experts, who have to choose a method for security analysis that is compliant with the standard. They moreover need to decide, e.g., on the abstraction level for the required documentation without any support from the standard. For example, security experts need to describe the business, organization, etc., and decide on their own what is the most relevant scope of elements to consider. They also have to find a method that allows them to achieve completeness of identifying stakeholders, assets, security goals, and so forth. Moreover, the standard does not provide a method for assembling the necessary information or a pattern for how to structure that information. The importance of these steps becomes apparent when one realizes that essential further steps of the ISO 27001 depend upon such information, e.g., the identification of threats, controls, and vulnerabilities.

In this chapter we present an extension of the CORAS risk analysis method [41, 62] to support the establishment of an ISO 27001 compliant ISMS. In previous work we analyzed the relations between different security requirements engineering and risk analysis methods [7], and our results showed that the ISO 27001 standard has a significant focus on risk analysis. The standard describes how to build an ISMS, and CORAS already supports many of these steps due to its focus on risk management. A further motivation for building on CORAS is that it is based on the ISO 31000 [31] standard, which is also the basis for the risk management process of ISO 27005 [36]. The latter standard refines the risk management process described in ISO 27001.

Additionally, the ISO 27001 standard demands legal aspects (such as laws, regulations, contracts and legally binding agreements) to be considered. CORAS provides support for this during the risk analysis by the extension called Legal CORAS [41]. CORAS moreover comes with tool support for all phases of the process, and the tool supports the specification of all CORAS diagrams, including Legal CORAS. A further useful feature of CORAS is that it facilitates the reporting of the results by a formal mapping from its diagrams to English prose. This mapping enables ISO 27001 document generation.

In summary, we use CORAS as a basis because of its structured method for risk management, its compliance with ISO 31000, the consideration of legal concerns, the tool support and the support for document generation.

We refer to the CORAS extension presented in this chapter as ISMS-CORAS. In Section 2.1 we present the main background of this method, which is CORAS, Legal CORAS and the ISO 27001 standard. In Section 2.2 we describe the steps of the ISMS-CORAS method, focusing on extensions to CORAS and demonstrating how it fulfills the demands of the standard. In Section 2.3 we give two selected examples of artifacts that are produced when applying the method. In Section 2.4 we conclude.

As part of a validation, we have applied ISMS-CORAS to a Smart Grid scenario from one of the NESSoS industrial use cases. Due to space constraints we cannot report on the validation in this chapter, but the reader is instead referred to the technical report [10]. The report also gives a more detailed presentation of the method.

2.1 Background

In this section we briefly describe the most important background to ISMS-CORAS, namely the CORAS method, its extension Legal CORAS, as well as the ISO 27001 standard.
2.1.1 CORAS

The CORAS method is a model-driven approach to risk analysis, and comes with a method, a language to support all steps of the method, as well as a tool that is used throughout the process to conduct the tasks and documenting the results.

The method follows the five steps of ISO 31000, which is context establishment, risk identification, risk estimation, risk evaluation and risk treatment. The three activities in the middle are referred to as risk assessment. Context establishment includes defining the scope and focus of the analysis, modeling the target of analysis at an adequate level of abstraction, identifying stakeholders and assets and defining the risk evaluation criteria. The target is modeled using a (semi-) formal language, such as UML [49], and assets are documented using CORAS asset diagrams. The context establishment also involves a high-level risk identification as part of defining the scope and focus. Risk identification is conducted by the identification, modeling and documentation of threats, threat scenarios, vulnerabilities and unwanted incidents with respect to the target of analysis and the identified assets. The modeling is conducted using CORAS threat diagrams. The risk estimation is also conducted using threat diagrams, and involves the estimation of likelihoods and consequences for the identified incidents and scenarios. Risk evaluation is conducted using CORAS risk diagrams, and includes determining which risks need to be considered for possible treatment by comparing the risks against the evaluation criteria. Finally, risk treatment is to identify means to mitigate unacceptable risks, and is conducted using CORAS treatment diagrams.

2.1.2 Legal CORAS

Legal CORAS is an extension of CORAS specifically for considering legal aspects and legal risk. The approach is based on existing work on legal risk management [43]. The initial target description in Legal CORAS contains a statement about whether and to what extent legal aspects should be considered in the risk analysis. The method elicits relevant legal aspects based upon the final target description.

The source of legal risks are legal norms, which are norms that stem from legal sources such as laws, regulations, contracts and legally binding agreements. When assessing legal risks, there are two kinds of uncertainties that must be estimated. First, the legal uncertainty is the uncertainty of whether a specific norm actually applies to circumstances that may arise. Second, the factual uncertainty is the uncertainty of whether these circumstances will actually occur, and thereby potentially trigger the legal norm. It is by combining the estimates for these two notions of uncertainty that we can estimate the significance of a legal norm and its impact on the risk picture. Legal CORAS comes with the necessary analysis techniques and modeling support, but the involvement of a lawyer or other legal experts is usually required.

2.1.3 ISO 27001

The ISO 27001 standard is structured according to the Plan-Do-Check-Act (PDCA) model. In the Plan phase an ISMS is established, in the Do phase the ISMS is implemented and operated, in the Check phase the ISMS is monitored and reviewed, and in the Act phase the ISMS is maintained and improved.

We focus in our work on the Plan phase, because we provide a specific method for building an ISMS, and because it is during this phase that the security risk analysis is stressed the most. (Cf. NESSoS D10.2 where we define the process for the risk and cost aware software development life cycle.) In future work we will also develop support for the other phases of PDCA. The Plan phase considers the scope and boundaries of the ISMS, its interested parties, environment, assets, and all technology involved is defined. In this phase, also the ISMS policies, risk assessments, evaluations, and controls are defined. Controls are measures to modify risk.

The standard demands a set of documents for certification. In the following we list these documents, giving them names in order to simplify the reference to them later in the chapter. (1) The Scope of the ISMS; (2) the ISMS Policy Statements that contain general directions towards security and risk; (3) the Procedures and Controls in Support of the ISMS; (4) a description of the applied Risk Assessment Methodology; (5) a Risk Assessment Report; (6) a Risk Treatment Plan; (7) documented Procedures to the effective planning, operation and control of the ISMS; (8) ISMS Records that can provide evidence of compliance to the requirements of the ISMS; (9) the Statement of Applicability describing the control
2.2 The ISMS-CORAS Method

The method presented in this section extends CORAS in order to support security management compliant with ISO 27001. Our contribution, namely ISMS-CORAS, follows the steps depicted in Fig. 2.1. The figure also shows the resulting artifacts from applying our method. While keeping the names of the method steps, we focus in our description on the difference to CORAS, and we explain how our changes to CORAS are related to ISO 27001 and its documentation requirements as described above. The steps and artifacts of CORAS that we extended are marked in grey in the figure.

Note, importantly, that the ISO 27001 standard does not have specific demands on the form of the documentation, as “documents and records may be in any form or type of medium” [35]. Hence, we can use CORAS artifacts in the creation of our ISMS documentation.

**Step 1: Establish the Context.** The main objective of this step remains the same, but is extended to fulfill the demands of the standard. This includes the specification of the characteristics of the business, and information about technology relevant to the target description. A specific subtask concerns the documentation of scope exclusions in a scope exclusion table. The table refers to elements in the target description and states reasons for excluding this particular element from the scope of the analysis.

Due to the standard it is mandatory to consider legal aspects in ISMS-CORAS. The identification of relevant legal aspects can be achieved, for example, by using our law patterns method [8, 23] or by...
involving domain experts and lawyers.

ISMS-CORAS requires an explicit description of the location of each element in the target description, due to demands of ISO 27001. Moreover, the location information is also essential for the consideration of legal aspects. For example, according to the German Federal Data Protection Act (BDSG) Sect. 4b, it is not allowed to store personal information outside of the European Union.

The step describes further how to use the target description to identify and document assets. ISMS-CORAS focuses in particular on information assets since an ISMS shall be built for their protection. A further task, as demanded by the standard, is to rate all assets according to their importance in order to prioritize the risk assessment. The rating and priority are documented in asset tables. ISMS-CORAS requires also the definition of asset owners in these tables. An asset owner is an "individual or entity that has approved management responsibility for controlling the production, development, maintenance, use and security of the assets. The term 'owner' does not mean that the person actually has any property rights to the asset" [35].

Moreover, the first step of ISMS-CORAS involves the documentation of existing security controls. These shall be discussed when refining the target description and documented in an existing controls table, which lists the controls and the assets these controls protect.

ISMS-CORAS aims to identify relevant vulnerabilities of the systems in the ISMS scope, and how attackers may exploit the vulnerabilities. The ISO 27001 standard states that the possible impact an exploitation of each vulnerability has on the information asset must be estimated. The documentation of this impact shall consider confidentiality, integrity and availability. We use attacker templates (see Fig. 2.2) to reason about attacker types and attacker motivations in relation to assets and the target description. The instantiation of these templates not only facilitates the security risk identification, but also results in documenting attackers that are out of scope and the assumptions that lead to scope exclusions. Their documentation is vital in order for other security experts to follow the reasoning of the threat model, e.g., in an audit of the ISMS. The attackers that are not excluded, their entry points in the target description, and the threatened assets are documented in attacker overview diagrams. These specify also the elements of the target descriptions and assets that are out of reach of a particular attacker, and therefore can be excluded from further analysis. Attacker templates and attacker overview diagrams are contributions of ISMS-CORAS.

Step 1 also involves the creation of a high-level risk table that defines who or what may cause incidents, how threats harm assets, and the vulnerabilities that the threat potentially exploits. ISMS-CORAS fulfills the ISO 27001 demands for a specific consideration of availability, confidentiality and integrity for information assets, as well. These are documented as high-level security objectives that mitigate the threats in an extended high-level risk table.

**Step 2: Risk Identification.** This step is conducted similarly to CORAS, but includes also the identification of legal risks. The standard risk identification uses threat diagrams specify that in more details how the attackers identified in the previous step may cause security risks by exploiting vulnerabilities. Legal CORAS supports the identification of legal risks and the modeling of relevant legal norms and their sources, the legal and factual uncertainties, as well as the risks that are caused by legal norms.

**Step 3: Risk Estimation.** This step is also similar to standard CORAS. However, ISMS-CORAS focuses additionally on the likelihood estimations of accidental misuses or exploits of the identified vulnerabilities. A specific task is to derive attacker types with a certain skill set, similar to the descriptions proposed in the Common Criteria [34]. The results of this step are documented in threat diagrams. A further task in ISMS-CORAS is to estimate the legal and factual uncertainty of the identified legal norms according to the description of Legal CORAS.

**Step 4: Risk Evaluation.** This step is performed according to standard CORAS, but ISMS-CORAS considers also attacker types and their possible exploits of vulnerabilities for deciding whether a risk requires treatment.

**Step 5: Risk Treatment.** Also this step follows CORAS, but ISMS-CORAS restricts the identification of risk treatments to the normative controls defined in Appendix A of the ISO 27001 standard. The treatments
<table>
<thead>
<tr>
<th>No.</th>
<th>ISO 27001 document</th>
<th>ISMS-CORAS artifacts</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scope of the ISMS</td>
<td>(Semi-) formal target description; scope exclusion table</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>ISMS policy statements</td>
<td>Extended high-level risk tables</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Procedures and controls in support of the ISMS</td>
<td>ISMS procedure table</td>
<td>1, 5</td>
</tr>
<tr>
<td>4</td>
<td>Risk assessment methodology</td>
<td>Description of the CORAS method</td>
<td>1-5</td>
</tr>
<tr>
<td>5</td>
<td>Risk assessment report</td>
<td>Asset diagrams; asset tables; risk evaluation criteria (risk matrix); threat diagrams; risk diagrams; Legal CORAS diagrams</td>
<td>1-4</td>
</tr>
<tr>
<td>6</td>
<td>Risk treatment plan</td>
<td>Treatment diagrams; treatment overview diagrams</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>Procedures to the effective planning, operation and control of the ISMS</td>
<td>Treatment diagrams; treatment overview table; control effectiveness table; written documentation</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>Statement of applicability</td>
<td>Treatment diagrams; treatment overview table</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>Management decision</td>
<td>Written documentation</td>
<td>1, 5</td>
</tr>
</tbody>
</table>

Table 2.1: ISMS-CORAS documentation

have to consider existing controls, and the asset owner is responsible for the controls protecting the asset. This information has to be included in the treatment diagrams. The residual risk has to be documented and the management has to approve it. As in standard CORAS, the treatment plans should consider cost-benefit reasoning, for example by using the CORAS extension we proposed in [71, 72] and reported in NESSoS D10.3.

Step 5 further requires a reasoning why a particular Appendix A control is considered or left out. For this purpose we propose to use treatment overview tables that refer to an asset, its security objective, and relevant treatment or treatment overview diagrams, and a reasoning of why the treatment is sufficient. We also have to document how the effectiveness of each control can be measured in a control effectiveness table that defines measures to assess the effectiveness of each control. The procedures and controls that are part of the ISMS have to be documented, and a further subtask is to document each procedure that is part of a selected control.

ISMS-CORAS Support for the ISO 27001 Documentation Demands. In Table 2.1 we give an overview of how ISMS-CORAS fulfills the ISO 27001 documentation demands as listed in Section 2.1.3. Recall from that section that we do not address the documentation of the (8) ISMS records. The first column is the document number, the second is the name we assigned the document, the third is the ISMS-CORAS artifacts that support the documentation, and the last is the method steps in which the artifacts are created.

2.3 ISMS-CORAS Example Artifacts

As part of the validation of the method we applied it to a Smart Grid scenario from the NESSoS use cases. Due to space constraints we cannot present the results in this deliverable. The reader is instead referred to the full technical report [10].

One of the new artifacts of ISMS-CORAS is the attacker template that is instantiated during Step 1 of the method. The template is shown in Figure 2.2.

Under the basic attacker description, the attacker type specifies the kind of attacker. A specific attacker can combine several types, so it is possible to tick several boxes. The classification of attackers into these types is based on our previous work [6, 9]. For a specific analysis threatened assets are instantiated with the assets documented in the asset diagram, and the assets that may be threatened by the attacker type in question are marked. The specification of the threatened security goals is demanded by the ISO 27001 standard. The entry points are specified my marking the relevant references to the target description.
ISMS-CORAS 21

Attackers. In addition, the documentation of the assets and target description elements supports the change management of the ISMS. All of these elements have to be re-evaluated after a change to the scope of the ISMS occurred and it has to be reasoned if they are still out of reach of the attacker.

We show instantiated attacker templates and attacker overview diagrams for physical attackers (see Tab. 6 and Fig. 8), for network attackers (see Tab. 7 and Fig. 9), for software attackers (see Tab. 8 and Fig. 10), and for social engineering attackers (see Tab. 9 and Fig. 11).

### Table 5: Attacker Template

<table>
<thead>
<tr>
<th>Basic Attacker Description</th>
<th>Refined Attacker Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attacker Type</strong></td>
<td></td>
</tr>
<tr>
<td>□ Physical Attacker</td>
<td>State which kind of skills the attacker needs to succeed.</td>
</tr>
<tr>
<td>□ Network Attacker</td>
<td>Attacker Motivation</td>
</tr>
<tr>
<td>□ Software Attacker</td>
<td>□ financial gain □ self-interest □ revenge □ external pressure □ curiosity</td>
</tr>
<tr>
<td>□ Social Engineering Attacker</td>
<td>Reasoning</td>
</tr>
<tr>
<td>□...</td>
<td>– Describe why the selected attacker motivations are relevant.</td>
</tr>
<tr>
<td><strong>Threatened Assets</strong></td>
<td>– Explain why the remaining entry points are not relevant.</td>
</tr>
<tr>
<td>□ Asset 1</td>
<td></td>
</tr>
<tr>
<td>□ Asset 2</td>
<td></td>
</tr>
<tr>
<td>□ ...</td>
<td></td>
</tr>
<tr>
<td><strong>Threatened Security Goals</strong></td>
<td>Reasoning</td>
</tr>
<tr>
<td>□ Availability</td>
<td>– Explain why the selected security goals of an asset are threatened.</td>
</tr>
<tr>
<td>□ Confidentiality</td>
<td>– Reason also why the remaining security goals are excluded.</td>
</tr>
<tr>
<td>□ Integrity</td>
<td></td>
</tr>
<tr>
<td><strong>Entry Points</strong></td>
<td></td>
</tr>
<tr>
<td>□ Target Description Element 1 □ Target Description Element 2 □ ...</td>
<td></td>
</tr>
<tr>
<td><strong>Assumptions of the Target Description</strong></td>
<td></td>
</tr>
<tr>
<td>□ Target Description Element 1 □ Target Description Element 2 □ ...</td>
<td>Describe all assumptions about the target description.</td>
</tr>
<tr>
<td><strong>Attack Paths</strong></td>
<td>Describe all attack paths from the entry points to the assets.</td>
</tr>
<tr>
<td>(possible vulnerabilities)</td>
<td></td>
</tr>
<tr>
<td><strong>Assumptions about the Attacker</strong></td>
<td>What are the assumptions about the motivation, skills, and resources of the attacker.</td>
</tr>
<tr>
<td><strong>Insider / Outsider</strong></td>
<td>Describe the difference if persons that are inside the scope and persons that are outside are the attacker.</td>
</tr>
<tr>
<td><strong>Recommended Resources</strong></td>
<td></td>
</tr>
<tr>
<td>Describe the resources required for the attacker to conduct the attack.</td>
<td></td>
</tr>
<tr>
<td><strong>Reasons for Scope Exclusion</strong></td>
<td>Describe the reasons for excluding the attacker or variants of the attacker from the scope of the threat analysis.</td>
</tr>
</tbody>
</table>

**Figure 2.2: Attacker template**
elements. In an instantiated attack template the elements are extracted from the target models. The entry points and the specification of the attack paths are based on Microsoft Threat Modeling [67]. An attack path is a description of an attack from an entry point to an asset, within the scope of the analysis. The attacker motivation is specified based on criteria from a SANS institute white paper [3].

Filling in the attacker template using as input the target description and the identified assets serves as a means to ensure completeness of the security risk analysis with respect to the target of analysis and the ISO 27001 demands. It furthermore serves as a basis for the subsequent and more detailed security risk identification, and it documents what may be excluded from the analysis and why.

Another new artifact is the attacker overview diagram, as exemplified in Figure 2.3. These diagrams give a graphical overview of the attackers as specified in the attacker templates, and serve as a means to check their correctness and completeness with respect to the target of analysis. The example is from the Smart Grid scenario and shows a network attacker. The lower compartment shows the entry points that may be targeted, and the upper compartment the assets that may be targeted. Any entry points or assets that are excluded are shown by placing them on the outside of the frame. The relevant security goals are specified by annotating the relations from the attacker to the assets.

2.4 Conclusion

In this chapter we have presented ISMS-CORAS, which is an extension of the CORAS model-driven approach to risk analysis. The method has been designed to support the establishment of an ISO 27001 compliant Information Security Management System, and to produce all the documentation that is demanded by the standard.

In previous NESSoS WP10 work on developing the risk and cost aware SDLC (cf. deliverables D10.2 and D10.3), we built closely in the ISO 31000 risk management standard and the ISO 27001 information security standard. Because CORAS already builds on the former standard and ISMS-CORAS fulfills the demands of the latter, the ISMS-CORAS method supports our overall process for a risk and cost aware SDLC.

We have already applied the method to one of the NESSoS use case scenarios, and in future work we plan to strengthen the validation by comparing ISMS-CORAS with alternative approaches, such as RIGER. The RIGER tool was presented in deliverable D10.3.
3 Cloud Service Risk and Cost-Benefit Assessment

In the e-Health scenario of NESSoS provided by Atos, cloud service providers (CSPs) are offering resources and services to other cloud users (such as medical staff and caretakers) based on agreed service level agreements (SLAs). A provider needs a well-balanced infrastructure, so that he can offer a high level of quality and violate as few SLAs as possible. There is also a special use case with cloud bursting where public cloud services (e.g. storage) can be subcontracted by the CSP. The main prerequisite is that customers trust the external provider and his ability to execute the tasks with the same or better quality, priority, and time budget as before. The forming of trust requires all services deployed in cloud to have reliable risk indicators, as well as an acceptable cost-benefit balance before an SLA is made, and that a continuous evaluation is carried out over time. In addition, societal and policy impact should be evaluated.

Within the provider scenario, risk management methods and collected risk-related data are of significant importance for short-term and long-term capacity planning and cloud management. The goal is to implement the provider scenario by archiving and analyzing the acquired risk information in order to derive indicators and other necessary information for the cost and benefit assessment (CBA) and afterwards also for the process of Societal and Policy Impact Assessment (SAPIA). Results are to be used as a guideline for capacity planning and management decisions. Additionally, short-term risk-related data is needed to compute the risk and to provide a recommendation for accepting/rejecting incoming SLA requests according to the provider's internal policies. The risk assessment models for the providers will build on the existing dynamic risk assessment models and will not be treated in this chapter. Methods that support such models include existing approaches such as CORAS [41], OCTAVE [2], CRAMM [16], RIGER (see NESSoS D10.3) or PILAR [63].

In order to exchange information between different phases (e.g. risk assessment and CBA) we have built a light ontology. These components for the exchange include 1) A central component that can be accessed by the end user and is used to carry out most operations available in CBA and SAPIA tool set; 2) External components used for risk assessment (RA); external components are dedicated tools to which the user is redirected in order to assess risk related to the currently considered context, scenario and use case.

The information exchange between the central component and external components is aimed at redirecting the final user to a proper external component, as well as obtaining information about the results of the conducted risk assessment.

In order to ensure the above mentioned functionality, different messages, with different sets of extra data, will be sent between the central component and external components. The objective of the ontology development is to properly define these messages and all extra parameters.

3.1 Concepts and Relations Encompassed by the Ontology

In this version of the ontology, we focus on the integration of two types of assessment, namely IT security risk assessment and cost-benefit assessment (CBA). For the IT security risk assessment we focus also on the special case of cloud computing and the “delta” it represents with respect to traditional security risk assessment, i.e. the security issues that are particular for cloud computing.

The work we have started on the integration of Societal and Policy Impact Assessment (SAPIA) is not described in this deliverable; as it is still in its initial phase, we rather plan to report it in the next WP10 deliverable. The classes were defined along with the properties for objects – messages sent between all components of the integrated tool set. The messages were divided into several types, depending on the direction and time of their transfer. Figure 3.1 gives a simplified presentation of the classes and the properties which connect them, both defined in the final ontology version.

A message sent between the toolset components is represented by an object of the Message class. This is an object whose integral parameter is an identifier, represented by an object of the MessageIdentifier class.

The message can be extended by adding data which contain IT or cloud security RA starting parameters – an object of the SecurityMeasureSet class. By means of this object it is possible to transfer a set
of user-selected security measures from the central component. These measures were selected by the end user in the initial phase of working with the tool set.

After completing the work in the external component (i.e. completing the RA), the information about the analysis results and the final set of analyzed security measures are sent with the use of an object of the ResultSet class. This object is also attached to an object of the Message class.

In this deliverable we do present all classes that represent objects sent between all components of the tool set: central CBA (or SAPIA in the next version) component and external components (e.g. CORAS or RIGER), as a part of the message (an object of the Message class). The object of the Result and ResultSet classes, for example, will be used to transfer to the central component the results of RA conducted in an external component.

The objects represented by SecurityMeasure and SecurityMeasureSet classes will be sent between all components of the tool set: the central component and components for risk assessment. The RA components may, for example, be RIGER or CORAS. These objects will be sent in both directions, to and from the central component, along with the objects of the classes, namely Result and Message. The objects of the SecurityMeasure and SecurityMeasureSet classes will be used to provide information to the tool set about security measures. If they are sent together with an object of the Message class, the object of the SecurityMeasure and SecurityMeasureSet classes will be interpreted by the components for RA as security measures selected by the end user with which he/she got interested before starting the RA analysis. If they are sent with an object of the Result class, the objects of the SecurityMeasure and SecurityMeasureSet classes will be interpreted by the central component as security measures which were actually analyzed by the end user in a tool for RA.

3.2 Cloud Security Risk Assessment

Our conceptual framework builds on (i) sufficiently large datasets which describe past performance of cloud services and providers, (ii) models to develop predictive probabilities for expected performance. We will add to the fundamental models simpler models for short-term risk assessments with small datasets to help providers to decide on SLA offers in an almost real-time and online setting; this type of setting is suitable for robust models which allows for getting reliable answers quickly and without extensive operations. In the NESSoS e-health scenario, the same models can be added to the medical staff confidence service
which will be enhanced with statistical reliability estimates of the providers’ offers.

Providers strive for a robust and reliable infrastructure since this is the foundation for their quality. For that reason, providers align their resources and system topology to the requirements of their users, aiming at a high rate of SLA fulfillment. In this context it is vital to identify bottlenecks and weak points within the infrastructure. However, not every cloud service or resource has the same failure rate, which means that upgrades do not have the same effects for the overall system performance. Likely, more powerful resources do not resolve the design problems, thus the investment often ends with the same problems while the infrastructure shows an even smaller level of utilization than before. Therefore, risk indicators will be used to identify the bottlenecks and to remove these by a combination of improved resources and organization. The assessed risk for resource failures will also be an instrument to support administrators to carry out short-term cloud optimization which depends on the current workload situation (for all components/experts involved in the cloud). In the real operational environment, the assessed failure/overloading risk will not only depend on a snap-shot of the current situation, but will require extensive analysis of history data, of current workload trends, as well as of experts’ input on expected development. When it comes to traditional IT security risk assessment there is no shortage of data. However, we also wanted to have “delta” assessment for cloud which has some vulnerabilities, threats and therefore risk which are different from these “traditional IT environments”. We did not make any changes when it comes to impact assessment, since the assumption is that loss factors, assets etc., will be similar if not same in cloud and non-cloud environment. The basic data was coming from a number of sources such as ENISA [21] cloud risk assessment, or documents issued by the Federal Risk and Authorization Management Program (FedRAMP)\(^1\), but also internal Atos documentation [5] used by consulting services. According to these Atos guidelines, for example, cloud vulnerability is included in “delta” assessment if:

- Is intrinsic to or prevalent in a core technology of cloud computing, such as VM escape, session riding and session hijacking, or insecure/obsolete cryptography.

- Has its root cause in one of the essential cloud characteristics. Such causes may, for example be i) On-demand self service -> unauthorized access, ii) Pooling/elasticity -> resources re-allocated to a different user at a later point, iii) Pay as you use -> manipulation of metering/billing data.

- Is caused by cloud innovations making existing and tested security controls hard or impossible to implement. Examples include i) IP-based zonning cannot be implemented or network based vulnerability scanning prohibited, ii) difficult to apply hardware security module (HSM) for key storage (no VM binding to hardware), iii) Monitoring, metrics have to be adapted.

- Is prevalent in established state-of-the-art cloud offerings (empirical vulnerabilities). For example missing authorization checks that lead to URL “guessing” attack, or injection vulnerabilities (e.g. SQL injection or cross-site scripting)

For further details, the reader is referred to the white paper issued and published by Atos [5].

\[\text{3.3 Integration of CBA and Cloud Security Risk Assessment}\]

Cost-benefit analysis (CBA) is usually carried out by the decision maker/stakeholder and/or his/her supporting staff, potentially supported by an external consultant acting as a moderator and facilitator. The RA and CBA integration experiments were done in collaboration with developers and decision makers in order to obtain realistic data and validate and improve the methodology and test process. In Figure 3.2 different steps of CBA can be identified in order to describe the working order. The steps 1-3 in the figure have to be performed only once per decision context. The aim of these working steps is to enter basic information and values (e.g. the requirements of the context of a use case). As a general rule, costs should be entered as negative (-) figures, and benefits as positive (+) figures. Steps 4-5 are evaluation steps.

Communication between components that reside on servers apart from the central one is done through a communication interface called REST API. It is an interface designed and implemented in an ad-hoc\(^1\)  

\(^1\)www.fedramp.gov
Figure 3.2: Cost-benefit analysis
manner and consists of several components. The main component is situated on the central server and is responsible for sending and receiving data. Other components that belong to the communication interface are web services based on the REST technology. The web services reside on the servers that host the Risk Analysis tools. They are responsible for receiving the messages sent from the central server, decrypting, forwarding the data depending on the called functionality, encrypting and sending back the response to the central server.

The data transferred between the components through the REST API is encrypted with a cryptographic cipher that ensures security of the data and authentication of the parties that exchange data. Each web service uses a custom library, developed specifically for NESSoS, for encrypting and decrypting the messages.

Also the message content that is exchanged between components is designed specifically for this proof of concept. The messages are exchanged through standard HTTP protocol, but the content is an XML based document, the content itself is based on the purpose of the message.

In order to be even more specific and applied to NESSoS we have made simulation of CBA in order to compare different phases of secure software development life cycle (SDLC). Of course in the real life this makes no sense since a) we already know e.g. that the cost is much higher if the vulnerability or error is discovered at the late SDLC stage, b) it makes no sense to compare different phases as they are complementary. However, as a proof of concept we selected several secure SDLC phases as the outcome of cloud security RA.

In the subsequent steps we even further limited our analysis and included detailed data. The first batch of trials was done only for testing and, as an exercise we selected “security requirements engineering” (as a generic single control) versus “late assurance” (also generic control, mainly based on cost-benefit data from static and dynamic application testing).

External factors and limitations that have an effect on the decision can be categorized as framing conditions. These framing conditions play an extremely important role in all decision-making, and they have to be taken into consideration also in the context of security measures and their evaluation. Such framing conditions can include the following:

- Previous (strategic) decisions (e.g. follow certain security strategy such as the EU initiative on legislation)
- Agreements (e.g. cost- & work-distribution between partners on certain security issues)
- Threat perception and urgency (security incidents or revealing may trigger urgent needs to initiate some security measures, e.g. recent PRISMA scandal)
- Security governance (the rules of interacting with stakeholders, budgeting processes)
- Uncertainty and risk-attitude of decision-maker(s) (e.g. the need to retrieve more information before commitment)

Budget is also typically one of the main framing conditions. In the CBA tool, budget limit is directly integrated to the calculations. This means that if the budget for the security measure investment exceeds, the demo will automatically inform the user about the situation in the Cost and Benefit Values view.

The view Cost and Benefit Breakdown structure is used to define (name) the relevant costs and benefit categories that describes the cost structure of the measure implementation and the measure use, i.e. the costs that are necessary to take into account in the assessment of the measure. There are three major cost and benefit types, namely investment costs, future costs and future benefits. Each of these can be further divided up to three sub-category levels. After selecting the main cost or benefit type mentioned above, the user defines / names the first level of the main cost categories. The first category level can be further divided to the second or third level of sub-cost or -benefit categories according to the user needs. It is not necessary to define costs and benefits up to the third level; this should only be used if a detailed structure of the costs and benefits is necessary.

This structure can be defined for each measure separately as well as for each main cost and benefit type which can be selected on top of the view, which could be useful for NESSoS and SDLC decision makers that have to compare e.g. CBA of tools & method from one category/phase to tools & methods
from another category. This allows the user to switch between the different types of costs or benefits and different measures, and enter his/her own cost and benefit structure for each security measure.

After this we did the second batch of trials, now with more detailed insight and more data about different assurance techniques. We included five control categories, namely random test case generation, specification-based test case generation, static code analysis, manual code inspection, and formal verification. The results we got are depicted in Figure 3.3. The right hand side of the diagram represents the benefits, whereas the left hand side represents the costs. The costs are in turn divided into initial cost and operational cost (leftmost). The operational costs are for the duration of software engineering and development projects.

This visualization screen is meant to give rapid support for decisions of senior management when it comes to the investments in secure software and service engineering. We can see that investment in formal verification, both initial and incremental, is very high and that although benefits could be significant, the choice of these methods cannot be recommended for low cost service and software engineering projects.

The general idea is that the results of assurance for services running in cloud would be attached in some form of machine readable format. The projects such as Assert4SOA\textsuperscript{2} or CUMULUS\textsuperscript{3} are working on the representation of certificate for services, where one type of certificate is based on testing and verification. This means that the selection of an assurance technique has also indirect impact, meaning that the “assurance strength” might be added as an attribute to the service and that the runtime selection of a service (e.g. “secure storage” for cloud bursting) can be done on a basis of certificate, which in its turn incorporates information about the selected assurance techniques.

In addition to cost benefit support that compares different techniques, decision makers were also interested in break-even points, that is, when the benefits are starting to outweigh costs. This is illustrated in Figure 3.4.

### 3.4 Conclusion and Future Steps

In the future versions we will also include other visualization and graphical presentations, and the methods for risk management will be further improved to consider risks caused by sources outside the provider site. Well-known examples are denial-of-service attacks, which not only damage the provider image, but also can lead to significant delays in the job schedule and to a domino effect regarding violated SLAs. An additional source of information is the network load for certain areas; network overload prevents the data transfer on time and has a strong impact on workflow processing.

\textsuperscript{2}www.assert4soa.eu

\textsuperscript{3}www.cumulus-project.eu
The risk self-assessment computed by the providers gives an objective measure—beside resource quality, service charge, and penalty fee—for the SLA assignment by the customers. However, a customer cannot verify the estimated risk as the provider will keep its autonomy and only publish mandatory data. Thus, a provider may publish an extensively lowered internal risk for the job execution in order to increase its resource utilization and the number of accepted jobs. A single user has a small chance to detect such unreliable providers.

Accordingly, in the NESSoS e-health use case, users should be supported by an independent instance collecting the information of published and real risk for external cloud services (e.g. test or evaluation of Alzheimer status). It would provide (i) reliability based on the relationship between previous provider published risk and assessments and final SLA status (historical data) and (ii) its own risk assessment for individual SLA offers from relevant service providers evaluated as unreliable (also based on historical data) or e.g. not compliant with privacy directive. It will support the risk assessment by provider monitoring of the number of fulfilled/violated SLAs, past performance, forecasts, and presumable service level requirements. This assessment may result in a bad rating of the provider, marking him as less trustworthy. Furthermore, cloud medical staffs that aim to offer the best possible service to their customers might use additional service to evaluate the quality of the risk assessment in the received SLA offers from the resource providers.

Patients approaching the external cloud services without medical staff may be granted access to the confidence service, so they can also evaluate individually the provider’s quality. In general, the confidence service could be a major building block for provider competitiveness on fair and objective facts. Currently we are investigating scenarios for cloud bursting in e-health.

Cloud bursting is an application deployment model in which an application runs in a private cloud or data center and bursts into a public cloud when the demand for computing capacity spikes. The advantage of such a hybrid cloud deployment is that an organization only pays for extra compute resources when they are needed.

Experts recommend cloud bursting for high performance, non-critical applications that handle non-sensitive information. An application can be deployed locally and then burst to the cloud to meet peak demands, or the application can be moved to the public cloud to free up local resources for business-critical applications. Cloud bursting works best for applications that do not depend on a complex application delivery infrastructure or integration with other applications, components and systems internal to the data center.

When considering cloud bursting, an organization must consider security and regulatory compliance requirements. For example, cloud bursting is often cited as a viable option for retailers that experience peaks in demand during the holiday shopping season. However, cloud computing service providers do not
necessarily offer a Payment Card Industry (PCI) Data Security Standard (DSS) [51] compliant environment and retailers could be putting sensitive data at risk by bursting it to the public cloud.

Other issues related to cloud bursting arise from the potential for incompatibility between the different environments and the limited availability of management tools. Cloud computing service providers and virtualization vendors have developed tools to send workloads to the cloud and manage hybrid environments, but they often require all environments to be based on the same platform.

Finally, the development work related to cost benefit optimization and fine tuning will be completed, including specific scenario e.g. cloud bursting. When spare resources at the own site are not available, providers need migration over multiple administrative domains, so outsourcing to public cloud is an adequate solution. Since resource providers have a detailed overview about several critical execution points, the offered risks in these aspects will be compared to decide whether outsourcing to a public cloud is feasible option from cost benefit, but also social or policy impact viewpoint. Resource providers are of various types which means that the resources they manage (e.g. storage for Aladdin cloud outbursts, service for simulation or evaluation, data services) and the risks they have to deal with are also different.

The current practice in cloud computing risk assessment, for example in FedRAMP, is including software security under category system and service acquisition since cloud services are usually developed by a different company than cloud service provider. However, use of intermediaries such as “cloud brokers” is making the cloud service supply chain more complex, since some data is in transit through these broker organizations. The brokerage between different cloud service offerings might also be expanded to perform these types of combines RA & CBA, and to do matching on behalf of the end user. In this case it is likely that FedRAMP or similar controls will have to be defined in much more details. For example, control SA-8 currently just refers to “security engineering principles” as a whole, while SA-11 lists more detailed requirements: “The service provider submits a code analysis report as part of the authorization package and updates the report in any reauthorization actions. The service provider documents in the Continuous Monitoring Plan, how newly developed code for the information system is reviewed.” We believe the controls and requirements should be defined in more details for service acquisitions. In addition, in the future we would like to perform risk assessment and CBA for cloud broker itself.

We also expect this work to be useful in standardization use case, jointly submitted to ETSI by NESSoS and three other European projects (main parts are coming from Assert4SOA project). The goal is to enable customers “shopping around” and searching for SaaS services provided by cloud-subscribers over the cloud infrastructure managed by a cloud-provider. “Shopping around” is similar to our previously explained “cloud broker” scenario, since it requires accurate assessments of the costs and risks involved in achieving the desired value. The only difference would be that the broker can have more expertise and tools to do these assessments.

Owners of software services that run on the cloud (not necessarily providers), that are called cloud-subscribers in the gap analysis done for ETSI, should provide transparent quantification of all costs, and clear assurance and evaluation of the set of non-functional properties to support customer service selection, as well disclose information about.

Customers searching for a service should be able to i) review the SLA/certificate schema to ensure all relevant information is represented, ii) ensure that the SLA/certificate specification contains all fields and properties needed to accurately and fairly represent all properties of the cloud-subscriber services being offered, iii) examine individual SLA/certificate data files to ensure proper data-entry and submission by the cloud-subscribers, iv) examine the SLA/certificate viewing/analyzing tool used by the procurement officers to rule out tool processing error.

Accurate tools for cloud customers, such as RA and CBA, and transparency in SLAs/certificates, costs, and data management practices of cloud-subscribers are mandatory to provide the needed insight for informed choices. This environment should enrich the cloud infrastructure to permit customers (or cloud brokers) to compare functionally-equivalent services and select the one with highest assurance that fits their requirements.
4 Methods for Selection of Risk Treatments

Long-living software systems keep evolving as they need to continue to satisfy changing business needs, new regulations and standards, and the introduction of new technologies. Such evolutions might expose the software systems to new risks, and might make the output of the current risk analysis on the software systems become partially obsolete. Consequently, the software systems might no longer be secure. This is in particular the case for Future Internet software and service systems that are highly dynamic and heterogeneous.

The results of a software system risk assessment are typically valid under a given context, which is a particular system configuration, and under certain assumptions about the target system at a particular point of time. Once a particular risk assessment is completed, countermeasures are proposed and decision makers (or managers) face the question of selecting an appropriate countermeasure alternative (i.e., a set of countermeasures) to be implemented in order to mitigate unacceptable risks. However, when the context evolves, also the risks evolve. Previously acceptable risks might become unacceptable or vice versa, or new risks may emerge [41, Chapter 15]. Thus, a current countermeasure alternative may no longer be appropriate and it is necessary to develop new ones to address the evolving or newly emerging risks. This might include adding additional security requirements as a protection to ensure the system security, or the creation of completely new security controls. Obviously, implementing new ones to replace obsolete ones may be more expensive than having one that still may be appropriate for evolving risks. The decision makers then face an alike question of selecting appropriate countermeasure alternative, yet in the extent of evolution of the context and of the risks.

While there exist several established risk assessment methods [27, 29, 57, 53, 44, 14, 2, 4], few provide support for a systematic selection on risk countermeasure alternatives [72, 66, 47], and even less for dealing with evolving risks [42]. Traditional risks assessment methods typically address a context at a particular point in time, and hence cannot guarantee the continuous validity of the risk assessment results in an evolving context. Concerning evolutions, in [42], the authors propose a general technique and guideline for managing risk in changing systems. However, they did not address the uncertainty of evolutions (i.e., likelihood of occurrence), and how to use this information to support the decision making process.

This chapter presents a method as an effort to fill some of that void. The focus of this chapter is not on how to obtain the uncertainty information, but rather on how to make use of them to produce additional factors to support the decision making process. In particular, we propose a method to take the potential changes, their uncertainty, and risk assessment outputs to quantitatively evaluate the evolution-resilience of a countermeasure alternative. The reader is referred to our paper for further details [69].

4.1 A Method to Early Deal with Evolution in Risk Assessment

The proposed method relies on a key concept, namely context, which is the context for the risk assessment. We use the definition of context from [41, Chapter 5], where a context is "the premises for and the background of the risk analysis. This includes the purposes of the analysis and to whom the risk analysis is addressed". The context description that is developed prior to the risk assessment serves as the basis for and input to the risk assessment tasks. It includes all required information to do a risk assessment for a software system, for instance a requirements model of the software system, the domain assumptions, the assets needed to be protected, and so on. The elements in a context, however, may change and evolve over time due to numerous reasons e.g., introduction of new requirements, threats that emerge, changes in security standards, new regulations, etc.

Figure 4.1 presents the conceptual model, expressed as a UML class diagram, on which our method builds. A Context Evolution Model is a collection of evolution rules, which captures the evolutions of context. An Evolution Rule is either Observable Rule, or Controllable Rule. The former captures the evolutions of a context. The latter captures all possible alternatives addressing risks within a context. Further discussion on evolution rules is provided in Section § 4.1.3. An Evolution Rule has one before context and many after contexts (due to several possible evolutions), where Context is defined as above. A context can be enriched with the output of the risk assessment. A Risk Countermeasure Alternative includes a collection of countermeasures, and a list of risks with residual risk levels after applying the
Figure 4.1: The conceptual model.

Table 4.1: The steps of the proposed approach.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identify evolving contexts:</td>
<td>Identify all possible changes that would change the risk picture of the system. Changes could be planned or not.</td>
<td>Any document of changes in context.</td>
</tr>
<tr>
<td>2</td>
<td>Perform risk assessment:</td>
<td>Apply an existing risk assessment method on each identified context. Also, the risk countermeasure alternatives are expected to as a part of output of the risk assessment method.</td>
<td>Contexts as identified in Step 1</td>
</tr>
<tr>
<td>3</td>
<td>Model context evolution:</td>
<td>Establish the context evolution model from the identified contexts and their corresponding risk countermeasure alternatives by using evolution rules.</td>
<td>Contexts with evolution probabilities (from Step 1), risk countermeasure alternatives (from Step 2)</td>
</tr>
<tr>
<td>4</td>
<td>Perform evolution analysis:</td>
<td>Run the evolution analysis on the established context evolution model to calculate the evolution metrics for each risk countermeasure alternative to support the decision making process.</td>
<td>Context evolution model</td>
</tr>
</tbody>
</table>

4.1.1 **STEP 1 – Identify Evolving Contexts**

This step takes all documents about the planned and potential changes of the system as inputs. In [41, Chapter 15], the authors distinguish three evolution perspectives (or perspectives for short): maintenance, before-after, and continuous evolution. The maintenance perspective relates a previous and possibly outdated risk document to the current system with the aim of maintaining the validity of the document. Hence it is not the focus of this work. The before-after perspective predicts the future context by anticipating changes in the current context. Finally, the continuous evolution perspective predicts the evolution of the current context over time based on anticipated gradual changes that can be described as a function of time.

We abuse the notion of before and after contexts to represent both before-after and continuous perspectives. The before context can be understood as the current context, and the after context will be the future context with potential changes. We refer to both before and after contexts as evolving contexts.

Changes in context can be elicited by using the input documents (planned changes), and/or anticipated by using various techniques (anticipated changes) such as brainstorming with chalk and blackboard, or techniques for requirements change anticipation [73, Chapter 6]. Readers are referred to [73, Chapter 6] for the more detailed discussion of these techniques. Note that it is intuitively impossible to identify all changes, but only a list of known changes. The unknown changes are addressed separately in [20].

Both anticipated and planned changes are all uncertain because “the only certainty is that nothing
is certain” (Pliny the Elder\(^1\)). They are therefore associated with a likelihood of occurrence, which is the belief that a change will happen in future. As the result, each evolved context is associated with a belief about whether it will materialize or not. This belief is called *evolution probability* which semantically is accounted by using the game-theoretic approach described in [70].

Figure 4.2(a) demonstrates the *before-after* evolution perspective. A context is depicted as a rectangle with child compartments. The first compartment shows the context name, and the second compartments exhibits the changes comparing to the *before* context. In this perspective, a *before* context might evolve to one or more *after* contexts. We call a possible evolution from the *before* context to an *i*\(^{th}\) *after* context as *evolution possibility* (or possibility). At the end of the day, only one possibility materializes. Figure 4.2(b) illustrates the *continuous* evolution perspective where changes happen continuously. The *before* context at current time \(t_0\) might evolve an *after* context at time \(t_1\) which might continuously evolve at time \(t_2\), and so forth.

There is another evolution perspective called *hybrid* evolution perspective. It is the generic form of both *before-after* and *continuous* evolution perspectives where the *before* context might have many evolution possibilities. Each of these possibilities might further evolve to other possibilities, and so forth. In this sense, the *hybrid* evolution perspective has a tree shape. The proposed method aims to support this *hybrid* evolution perspective, and therefore support both *before-after* and *continuous* ones.

This output of this step is a list of contexts associated with their corresponding evolution probabilities.

**4.1.2 Step 2 – Perform Risk Assessment**

In this step, we can employ state-of-the-art risk assessment methods and techniques (e.g., Attack Trees [57], Cause-Consequence Diagrams [44], and CORAS [41]) to perform risk assessment for identified contexts. The outcome of this step is list of risk countermeasure alternatives, which are also the output of a risk assessment method.

A *risk countermeasure alternative* includes a list of countermeasures, and the residual risks (with residual risk level) of a system after implementing the countermeasures. A countermeasure could be a security controls (e.g., technology, policy or mechanism), or a high level security requirement that mitigates risks. A risk level is a pair of the likelihood by which a risk might occur, and its impact on an asset. Based on risk level, a risk is categorized, such as acceptable or unacceptable. A residual risk level is the risk level after implementing countermeasures.

When performing risk assessment on *after* contexts, we can do either a full risk assessment from scratch, or an incremental risk assessment taking advantage on the risk assessment on the *before* context. Needless to say, the former strategy does not use resources efficiently. The latter is better since it only addresses the changed parts of the *after* context comparing to the *before* context [41, Chapter 15].

**4.1.3 Step 3 – Model Context Evolution**

This step takes the identified contexts and their corresponding risk countermeasure alternatives to establish the context evolution model. We employ an approach dealing requirements evolution [70] to model the context evolution in terms of evolution rules.

---

\(^1\)Gaius Plinius Secundus (23–79), a Roman naturalist, and natural philosopher.
There are two kinds of rules: \textit{observable rule} and \textit{controllable rule}. The former captures the way how the context evolves. The latter captures different alternatives to address risks in each context. An evolved context, as aforementioned, is foreseen with a certain evolution probability. To the sake of brevity, we assume that the evolving contexts identified in Step 1 are complete and mutually exclusive. In other words, among after contexts, there is always one (and only one) context materialized at the end.

Let $C$ be the current context, and $C_i$ be the $i^{th}$ after context of $C$. The observable rule ($r_o$) is described as follows.

$$r_o(C) = \left\{ C \xrightarrow{p_i} C_i \mid \sum_{i=1}^{n} p_i = 1 \right\}$$  

(4.1)

where $n$ is the number of after contexts of $C$, $p_i$ is the evolution probability for which $C$ evolves to $C_i$. The total of all $p_i$ equals one because the after contexts are complete and mutually exclusive. This information is taken from Step 1.

Let $C$ be a context, and $CA_j$ be a risk countermeasure alternative of $C$. The controllable rule ($r_c$) is described as follows.

$$r_c(C) = \{ C \rightarrow CA_j \mid j = 1 \ldots m \}$$  

(4.2)

where $m$ is the number of risk countermeasure alternatives of $C$. This information is taken from Step 2.

The before-after evolution perspective is represented by an observable rule. The continuous and hybrid evolution perspectives are represented as a sequence of observable rules where the current context of an observable rule is the after context of another observable rule, and so forth.

Figure 4.3 shows a graphical visualization of the context evolution model of the hybrid evolution perspective. As both before-after and continuous evolution perspectives are special cases of the hybrid perspective, this graphical visualization of context evolution model is also applicable for these two perspectives. The observable rule is denoted by connections between the before context to the after contexts. The decorators on the connections are the evolution probabilities. To denote the controllable rule, the rectangles representing context are extended with a new compartment containing risk countermeasure alternatives. Risk countermeasure alternatives are represented by round rectangles. Those alternatives which are appropriate for the before context, but inappropriate for an after context are decorated by dotted border with gray text. Rectangles with the same text indicate the same alternatives. The controllable rule then is understood as different risk countermeasure alternatives within a context.

\subsection*{4.1.4 Step 4 – Perform Evolution Analysis}

This step performs the evolution analysis on the context evolution model to calculate evolution metrics. These metrics are used to enrich risk countermeasure alternatives to support the decision making process. In particular, the evolution metrics aim to answer the question of the extent to which a risk countermeasure alternative can resist the evolution. The analysis relies on two quantitative metrics: max belief and deferral belief \cite{70}.
**Max Belief** (MaxB) measures the maximum belief that a risk countermeasure alternative will be appropriate if evolutions happen. By term appropriate, we mean the residual risks after applying the countermeasure alternative in the evolved contexts will still meet the risk acceptance criteria. The system, therefore, will still be secure.

**Deferral Belief** (DefB) is the belief that a risk countermeasure alternative will be inappropriate after evolutions happen. In other words, it is the belief by which the implementation of the risk countermeasure alternative should be delayed.

To aid the calculation of these metrics, we define a binary function appropriate() that takes two inputs: a context C, and a risk countermeasure alternative CA. This function returns true if CA is appropriate within C, or false otherwise. The max belief of CA could be calculated by identifying all evolution possibilities where CA is appropriate with respect to the output of appropriate(). The max belief is then the maximum value of the evolution probabilities of these possibilities. On the contrary, to calculate the deferral belief, we sum up all these identified probabilities. By definition, the deferral belief is the complement of the sum. In this sense, we formulate max belief and deferral belief of CA for the before-after evolution of the context C in the following equations:

\[
\text{MaxB}(CA|C) = \max_{\{\langle C \rightarrow p_i C_i \rangle \in r_e(C)|\text{appropriate}(C_i, CA)\}} p_i \tag{4.3}
\]

\[
\text{DefB}(CA|C) = 1 - \sum_{\{\langle C \rightarrow p_i C_i \rangle \in r_e(C)|\text{appropriate}(C_i, CA)\}} p_i \tag{4.4}
\]

For the continuous and hybrid evolution of the context C, we extend concept max belief and deferral belief to continuous max belief (MaxB) and continuous deferral belief (DefB). The formulas of these extended metrics are as follows.

\[
\text{MaxB}^*(CA|C) = \begin{cases} 
\text{appropriate}(C, CA) \\
\max_{\{\langle C \rightarrow p_i C_i \rangle \in r_e(C)|\text{appropriate}(C_i, CA)\}} p_i \cdot \text{MaxB}^*(CA|C_i) 
\end{cases}
\quad \text{if } C \text{ does not evolve,} \quad \text{otherwise.} \tag{4.5}
\]

\[
\text{DefB}^*(CA|C) = \begin{cases} 
1 - \text{appropriate}(C, CA) \\
1 - \sum_{\{\langle C \rightarrow p_i C_i \rangle \in r_e(C)|\text{appropriate}(C_i, CA)\}} p_i \cdot (1 - \text{DefB}^*(CA|C_i)) 
\end{cases}
\quad \text{if } C \text{ does not evolve,} \quad \text{otherwise.} \tag{4.6}
\]

In function (4.5) and (4.6), by saying the context C does not evolve, we mean that no evolving context of C is identified. This does not mean C stops evolving, but its evolution may be ignored in the analysis due to some reason.

After this analysis, each risk countermeasure alternative is quantified with two evolution metrics: max belief and deferral belief. Together with the benefit and cost of a risk countermeasure alternative (see also our methods to support this [71, 72], and Deliverable D10.3), these evolution metrics can support designers in selecting the most evolution-resilient alternative for the system.

In the perspective of evolution-resilience, a good alternative should have a high chance to continue operating properly without (or with minor) modification, and a low chance to be obsoleted when evolution happens. Therefore, a better alternative is one that has a higher max belief and a lower deferral belief.

### 4.2 Example

This section illustrates the proposed method by an example taken from an industrial project in the Air Traffic Management (ATM) domain: the System Wide Information Management (SWIM) project [60, 1]. The SWIM project aims to provide consistent, efficient, transparent, and secure means for information interchange among ATM systems. This example focuses on the Messaging Service, a part of SWIM, which is responsible for a common and reliable layer to exchange messages within the SWIM architecture.
Figure 4.4 illustrates the Messaging Service (MSG) architecture. In the figure, rectangles denote components of the messaging services, whereas round rectangles denote external components communicating with the messaging services. The headed arrows determine the connections among components. Inside the MSG, The Mediator component is in charge of message transformation. The Message Routing is in charge of point-to-point message transmission. Internal Services within SWIM can directly connect to the MSG. The communications between other ATM Services, External Service and MSG are done through the Web Service interface, and are subject to the supervision of the Security Gateway. Hereafter we illustrate the application of our method on this example.

**Applying Step 1:** For simplicity we do not show the *before* context. Instead, we only describe two anticipated changes might happen in the *before* context.

- **C1** The business rules become more complex, therefore more expressive policies are required to protect confidential resources.
- **C2** The ATM network changes from private network to public internet.

Since these changes are independent, either only C1, or only C2, or both might happen. Consequently, the following after contexts are identified with corresponding probabilities of occurrence:

- **After_1** ($C_1$): no change will happen – 0.10.
- **After_2** ($C_2$): only C1 will happen – 0.30.
- **After_3** ($C_3$): only C2 will happen – 0.35.
- **After_4** ($C_4$): both C1 and C2 will happen – 0.25.

**Applying Step 2:** We perform risk assessment for all identified contexts. We only consider one risk resulting from the risk assessment on the *before* context of MSG: “unwanted access to confidential resources” (R1). The corresponding countermeasure, in a high level of abstraction, is: “implements authentication/authorization mechanism” (SR-1). It is further refined into three operational countermeasure alternatives: $CA_1$: X.509+SAML\(^2\), $CA_2$: Kerberos+LDAP\(^3\), and $CA_3$: SAML+XACML\(^4\). Each countermeasure alternative includes two components: one for authentication (the first component), and another one for authorization (the last one). We use RLV1, RLV1’, and RLV1” to respectively represent for the residual risk level of R1 after applying $CA_1$, or $CA_2$, or $CA_3$.

We do not describe the risk assessment on after contexts, but discuss only the difference between the risk assessment output of these contexts and that of the *before* context, as shown below.

- **$C_1$**: as same as the *before* context.

---

\(^2\)Security Assertion Markup Language  
\(^3\)Lightweight Directory Access Protocol  
\(^4\)eXtensible Access Control Markup Language
Figure 4.5: The evolution of Messaging Service.

- \( C_2 \): a new risk is identified: "unauthorized access to confidential resources because the authorization mechanism cannot capture expressive policies" (R2). The countermeasure for R2 is: "implement a high expressive authorization mechanism" (SR-2). The refined countermeasure alternative will be either \( CA_2 \), or \( CA_3 \). Similarly, the residual risk levels of R2 by applying \( CA_2 \) and \( CA_3 \) are respectively denoted as \( RLV_2 \) and \( RLV_2' \).

- \( C_3 \): a new risk is identified: "system collapses due to the malicious attacks on centralized key server" (R3). The countermeasure for R3 is: "implement a robust key management mechanism" (SR-3). The refined countermeasure alternatives will be either \( CA_1 \), or \( CA_3 \). Similarly, the residual risk levels of R3 by applying \( CA_2 \) and \( CA_3 \) are respectively denoted as \( RLV_2 \) and \( RLV_2' \).

- \( C_4 \): both risks R2 and R3 are identified. Consequently, the only refined countermeasure alternative will be SAML+XACML. The residual levels of R2 and R3 are denoted as \( RLV_2 \), and \( RLV_3 \) respectively.

**Applying Step 3:** Let \( C_0 \) be the before context, the evolution rules are as follows:

\[
\begin{align*}
r_o(C_0) & = \{ C_0 \}^0.1 \rightarrow C_1, C_0 \}^0.30 \rightarrow C_2, C_0 \}^0.35 \rightarrow C_3, C_0 \}^0.25 \rightarrow C_4 \}, \\
r_c(C_1) & = \{ C_1 \rightarrow \{ \{ X.509+SAML \}, RLV1 \}, C_1 \rightarrow \{ \{ Kerberos+LDAP \}, RLV1 \}, \\
& \qquad C_1 \rightarrow \{ \{ SAML+XACML \}, RLV1 \} \}, \\
r_c(C_2) & = \{ C_2 \rightarrow \{ \{ Kerberos+LDAP \}, RLV1, RLV2 \}, C_2 \rightarrow \{ \{ SAML+XACML \}, RLV1, RLV2 \} \}, \\
r_c(C_3) & = \{ C_3 \rightarrow \{ \{ X.509+SAML \}, RLV1, RLV3 \}, C_3 \rightarrow \{ \{ SAML+XACML \}, RLV1, RLV3 \} \}, \\
r_c(C_4) & = \{ C_4 \rightarrow \{ \{ SAML+XACML \}, RLV1, RLV2, RLV3 \} \}
\]

Applying Step 4: Using the functions (4.3) and (4.4) we calculate the \textit{max belief} and \textit{deferral belief} of each risk countermeasure alternative. The results are reported in Table 4.2.

From the evolution-resilient perspective, the alternative SAML + XACML is the preferred since it has the highest \textit{max belief} and the lowest \textit{deferral belief}. The alternative X.509 + SAML places the second, and the last one Kerberos + LDAP is the third. This information is combined with the residual risks to support the eventual countermeasure selection.

---

\(^5\) This is a limit of SAML [52, chapter 6]
Table 4.2: The max belief and deferral belief.

<table>
<thead>
<tr>
<th>Risk Countermeasure Alternative</th>
<th>Residual Risks</th>
<th>Max Belief</th>
<th>Deferral Belief</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA1: X.509 + SAML</td>
<td>R1:RLV1, R3:RLV3</td>
<td>0.35</td>
<td>0.55</td>
</tr>
<tr>
<td>CA2: Kerberos + LDAP</td>
<td>R1:RLV1', R2:RLV2</td>
<td>0.30</td>
<td>0.60</td>
</tr>
<tr>
<td>CA3: SAML + XACML</td>
<td>R1:RLV1&quot;, R2:RLV2&quot;, R3:RLV3&quot;</td>
<td>0.35</td>
<td>0</td>
</tr>
</tbody>
</table>

4.3 Conclusion

The context of life-long evolving software systems might evolve over time, and as a result, the risks of the software system might also evolve. Under the evolution of risks, the systems might be no longer secure. Therefore, new countermeasures need to be implemented to mitigate new risks. This, however, is much more expensive than addressing these risks at development time.

We are aware of only one study in this field of research that has introduced general techniques and guidelines for managing risk in evolving systems. However, this study does not specify how to deal with the uncertainty of evolutions. This chapter presents a pioneer method in filling in the gap by proposing a method inspired from metrics on mutation evolutions of requirements [70] to evaluate the evolution-resilience of the risk countermeasure alternatives which are the output of existing risk assessment methods. This provides more insights into the evolving risks so that decision makers can better select the most appropriate alternative.
5 Software Application Security Risk Assessment: Lessons Learned from Practical Applications

In the preceding NESSoS deliverable D10.3, chapter 5, we proposed a process for continuous security risk assessment suited for iterative software development. The process can be briefly summarized as follows:

1. The identification of security requirements, the deduction of corresponding business worst case scenarios (BWCS) and the identification of non-technical likelihood factors.
2. The creation of a security overview reflecting the application's attack surface and security architecture and serving as a basis for
3. an adapted threat modeling, during which the analyst examines which abstract threats can lead to the previously defined BWCS.
4. An initial risk estimation for each BWCS considering the business impact and the technical and non-technical likelihood of all associated technical threats.
5. The selection and
6. conduction of suitable security tests to better estimate the real likelihood of the concerned technical threats.

In [64], we further refined and adapted the ideas with a particular focus on the systematic identification of security tests based on risk analysis. This chapter describes the practical experiences and the lessons learned from applying the process steps on a Siemens-internal web-based data storage repository and the standard IT parts of a Siemens-developed distributed system used to collect energy consumption data in industrial environments. The lessons were that strict requirements to time spent and to scalability are critical acceptance factors (Section 5.1), traceability alone is not a sufficient benefit for industrial acceptance (Section 5.2), we need more technical information in addition to what was initially included in the Data Flow Diagram based security overview (Section 5.3), the manual generation of data flow diagrams has to be more comfortable (Section 5.4), how risk-assessments and design reviews can profit from one another (Section 5.6), and finally why web applications are a special case for our proposed process (Section 5.7).

5.1 Strict Time Requirements and Scalability are Critical Acceptance Factors

The practical application of the proposed risk analysis method showed that the tight time budget on the one hand, and the complexity of real world systems on the other hand require efficient and scalable approaches. Besides all advantages of an in-depth risk analysis and the belonging preliminary technical system analysis, these activities do take time. Their extent must be appropriate in regard to the overall time budget as they reduce the available resources for practical testing which—at the end of the day—yields the actual "tangible" results, namely exploitable vulnerabilities that the system owner wants to fix. Therefore the risk analysis must be as light-weight as possible, and analysts, once having understood the process and the dependencies of its activities, may be advised to not apply all possible steps in all detail. This will help to get "the biggest security bang for the buck", but it is also important for psychological reasons: Many security practitioners and paying customers do not want to spend much time with seemingly non-profitable and less exciting analysis "overhead", but cannot wait to start with practical security tests.

Yet, it would be preferable to have a process that is designed to be scalable, i.e. allowing systematic and reasonable simplifications\(^1\) which enable the analyst to perform a custom-tailored "just enough" risk analysis. In the following we suggest such systematic simplifications for the creation of the security overview. The prioritizations are based on practical experiences, but are also aligned with best practices described in [26] and [50].

\(^1\)An example of such simplifications are step three and step four of the security design review outlined in Section 5.6.1.
5.1.1 Simplifying the Security Overview

Data Flows

The security overview creation should start by targeting

- data flows between and from actors or 3rd party systems in other trust zones and
- data flows between components of the same trust zone but whose communication passes over untrustworthy channels.

Only if time, the overview should be extended to

- data flows between components in the same local but different trust zone, e.g. local communication between a privileged and non-privileged component which could lead to local privilege escalation.

Our tool development activities will reflect these prioritizations: We will start with the detection of inter-system interfaces, i.e. network and file I/O. Only in a second step, we will target inter-process interfaces, namely local file I/O, registry access, named pipes, the command line and graphical user interfaces. For trust boundary identification, the development effort will be geared towards the automated detection of process privileges, file permissions and permissions of operating system objects.

Security Controls

We suggest adding security controls to the security overview in the following order:

- Authentication
- Encryption
- Authorization
- Input filtering

Only if appropriate in regard to the project time frame, include (in that order):

- Output sanitization
- Logging
- Monitoring
- Integrity checks

Note that this prioritization already reflects the narrowed choice of target systems (see Section 5.5.4) and may be different for other system types, e.g. web applications. The prototypical tool support will first include the detection of the top four controls along with corresponding possible flaws in their design.

5.2 Traceability is not Sufficient for Industry Acceptance

The experiences showed that risk analysis capabilities are appreciated but may not be sufficient for a sustainable industry acceptance. The latter would be easier to achieve if the method provided the following additional two benefits:

1. Improved security test coverage by suggesting a large number of adequate test types, especially less common ones that may not be thought of by the ordinary tester.

2. More concrete security test information. For example, in addition to suggesting "SAL 2 fuzzing" for the interface “registry”, the security analyst could be supported by giving the exact location of the value and the name of the functions that the fuzzing data will/may pass.

The first feature can be achieved by a well-stocked test library; the second requires more technical effort but could leverage the information extracted for a more technical security overview (see Section 5.3).
5.3 The Security Overview Should not Reduce to a DFD

DFDs are useful for capturing and presenting a high-level system overview as well as detecting certain design flaws. However, providing more concrete test information (see Section 5.2) requires bringing the model closer to the target's implementation. We therefore suggest using a more detailed, more low-level model and regard the DFD as a high-level view on it. This ensures that detailed low-level information can be included in the model without having a too crowded DFD. Concretely, we propose the model to be graph-based with

- nodes being the main executable, its belonging shared libraries, the memory regions, functions and, at the lowest level, basic blocks and assembly instructions;
- edges being control flow and, as far as possible, data flow relations\(^2\) between the nodes.

In Section 5.5.4 we discuss existing tools for the generation of such models.

One major challenge that still has to be investigated is how high-level DFDs originating from white board sessions (also see Section 5.4) can be best merged with these "bottom-up" tool-generated low-level parts.

Besides the previously mentioned benefit of providing more concrete test information, a more detailed model will also improve other parts of the framework: First, it allows creating more precise patterns used for mapping technical threat scenarios (TTSs) to test types (as described in NESSoS Deliverable 10.3). Second, since a successfully matched pattern is regarded as a vulnerability indicator, it permits a more precise initial estimation of the technical likelihood of TTSs.

5.4 Manual Generation of Digital DFDs Must be Quick and Comfortable

While some information for the security overview can be automatically extracted from binaries, other high-level and context-sensitive information can only be acquired in white board sessions with architects and developers. To make the resulting manually drawn and annotated DFDs machine readable, they need to be converted in a digital format. A quick investigation revealed that the format of Microsoft Visio diagrams is a promising candidate. It is XML based and thus well-suited for generation and processing.

However, the manual transformation into Visio diagrams has a low (psychological) acceptance factor as it is tedious and takes time. It has to be investigated to what extend using Visio, possibly in combination with a projector, instead of using a white board will results in less dynamic and less interactive sessions. The use of a digital pen or a digital whiteboard may mitigate these drawbacks although this requires software that automatically converts such manually drawn diagrams into Visio formatted DFDs.

Improving the manual generation of DFDs will relieve the analyst from tedious work, and thus lead to less careless mistakes and a higher acceptance of the entire procedure.

5.5 Target Systems Scope Needs to be Narrowed Down

Sections 5.2 and 5.3 describe the need for more concrete test information and a more technical model. For a practical application and evaluation of these modifications, and also for a prototypical realization of tool-support, we need to define more precisely which target systems we want to cover. We identified the following major determining properties:

5.5.1 Operating System (OS)

The OS determines the binary format, available APIs, possible hooking possibilities, and which dynamic system analysis tools can be used.

\(^2\)leveraging existing approaches on data flow analysis in binary files, see e.g. [58] or [74]
5.5.2 Processor Platform

The processor type defines the assembly instruction set, for example 32bit x86 or 64bit ARM, and thereby which reverse engineering tools can be used to generate the low-level part of the model. In this context, the intermediate language of managed code (see Section 5.5.3) can also be considered as processor architecture.

5.5.3 Managed or Unmanaged Code

Non-managed Code

Non-managed or "native" applications, for example written in C/C++, allow using standard debuggers but they cannot be reliably de-compiled, that is transformed to high-level source code. Yet this does not pose a problem as the low-level model parts can be built from binaries. Section 5.5.4 discusses possible existing tool support for this.

Managed Code

Managed code applications, for example written in Java or .NET, on the other hand, can be decompiled\(^3\) and, if the program was not obfuscated, the resulting source code is of high-quality. However, practical experiments showed that it is difficult to synchronize this source code with a debugger. In general, debugging .NET applications turned out to be more complicated than expected, especially if one does not have the original build environment. There are two aspects of managed applications that still need to be investigated:

- How, and with how much effort and what quality, can the graph-based model be generated from the intermediate language (e.g. MSIL for .NET or bytecode for Java) or from the decompiled source code?
- Tools such as ngen (.NET), Java2exe and python2exe convert a managed code application into a native one. We did not yet investigate the exact technical functioning of these tools and thus cannot estimate to what extend analysis techniques from native applications will work on the transformed applications.

5.5.4 Our Choice

We decided to start applying the revised method on unmanaged, 32bit Windows (x86) applications and focus our development effort for tool-support accordingly. The choice includes "rich" or "full" clients, server applications, third party components, and embedded devices.

The above system characteristics match well with the software that is part of many Siemens products. Moreover, other than for web applications (see Section 5.7), a tool-based security overview generation is particularly useful for these kinds of systems as they tend to

- have a complex architecture due to technical or legacy constraints,
- contain third party and (native) legacy components, and
- are developed by different teams, possibly located in different countries.

Besides the aspect of providing benefit, the decision was also driven by technical implementation-related considerations:

- Native code tends to be easier dynamically analyzed with a debugger than managed code.

\(^3\)In case of .NET programs, a tried and tested tool is reflector, available at [http://www.red-gate.com/products/dotnet-development/reflectort/](http://www.red-gate.com/products/dotnet-development/reflectort/)
• There is sophisticated tool support for debugging, disassembling and, very important, for the generation of the intended graph-based model. Practical experiments showed that the python-based PaiMei framework\(^4\) and especially its included scriptable debugger PyDbg, but first and foremost, IDA Pro\(^5\) are well-suited tools. Key features of IDA Pro include, for example, notes from OneNote and Mindmap.

• Windows API call hooking/analysis is a very promising approach for fast results. It is easier to applying this technique on non-managed applications.

5.5.5 Incremental Procedure: From Simple to More Complex SUTs
We intend to start with self-contained modules such as a server application or full-client. Then, step-by-step, we want to extend this by (1) connecting several such modules, (2) including 3rd party components, and finally (3) also target embedded devices.

5.6 Risk-Assessments and Security Design Reviews can Profit from one Another

The activities involved in the proposed risk-assessment showed a great overlap with what is done when reviewing a software system’s security design.

5.6.1 Security Design Review

A very-light weight, but practically tried and tested, DFD-based "security design review" may be summarized as follows (the priorities are based on our experience and may be different in other environments):

• Identify the main subsystems

• For each subsystem, go through its components in order to detect security relevant interfaces (i.e. data flows) to
  – external actors or 3rd party systems in other trust zones (if time, consider also local trust zones and the corresponding data flows that cross them), and
  – system components in other local zones where the communication passes over untrustworthy channels.

• For each of these data flows, check if necessary security controls are in place, starting with
  – authentication,
  – authorization
  – encryption, and
  – input filtering

• If time, also check for the following (lower prioritized) controls (in that order):
  – output sanitization
  – logging
  – non-repudiation
  – integrity checks

• For the rest of the available time or budget: go through the already existing controls and look for weak or insecure conceptual design properties.

\(^4\)https://github.com/OpenRCE/paimei
\(^5\)https://www.hex-rays.com/products/ida/index.shtml
Examples of frequently discovered design flaws are black list input filtering, only one-way authentication, client-side trust (i.e. the implementation of security controls in an untrustworthy client module), and static / hard-coded keys. A Siemens-internal check list with more than 50 possible design flaws supports this step. The entries are grouped according to the kind of security control and are given a priority rating. The list allows systematically adapting the review to the available time or to the desired assurance level.

5.6.2 Possible Synergies

Tool-supported DFD Generation

The proposed risk analysis method and the above outlined security design review both use a DFD-based technical system description. The DFD used in our proposed risk-analysis method is more detailed and, in general, has a higher quality. This is because it is based not only on interviews, workshops, etc., but also on a technical analysis of the real system. The design review would benefit from a tool-supported generation and validation of the DFD.

Sharing Knowledge Captured in Mapping Rules and Test Library/Check-list

The detection of design flaws could benefit from a pattern matching approach similar to that used in the proposed risk assessment procedure. The idea of indicators that map TTSs to tests can easily be adapted to map to design flaws.

As explained in Section 5.6.3, design flaws have a different nature than implementation bugs. Nevertheless, the concept of a test library and rules mapping the vulnerability pattern to its entries can also be used for a more traceable and more standardized security design review. Moreover, the design flaws from the check list used during a design review can be integrated into the risk analysis test library.

Performing the outlined adaptations and using the synergies leads to improved and more efficient design-reviews and risk assessments.

5.6.3 Excursus: Testing for Design Flaws vs. Testing for Implementation Bugs

Tests for design flaws, when compared to tests for implementation bugs, have certain particularities: They are often simple(r) to execute, involve analysis rather than generating and inputting data, and often require a context-sensitive interpretation of the results. Two examples illustrate this.

Assume a frequently discovered security design flaw such as a missing security control (for example an unencrypted network data flow). The test for this flaw may be as simple as asking the architect or developer if the control exists, and if it does not, assess if it is a problem considering the system's context. In case there are no contact persons, the contacts do not know, or in order to validate the given answer, analyzing the network packets will give a quick and technically-founded answer.

Another common design flaw is client-side trust, for example client-side authorization. Again, one could simply ask the architect or developers if they implemented the authorization mechanism in the client. Alternatively, one can screen the network traffic for suspicious data fields or analyze the client's (decompiled) source code or its binaries and look for an authorization routine.

5.7 Particularities of Web Applications in regard to the Proposed Method

Analyzing the Infobase application revealed that parts of the proposed risk analysis procedure can be made more streamlined if used on web applications. This is due to the following particularities: The general architecture is relatively similar for most web applications. For example, a very common set-up consists of a web server, possibly an application server, a data base and a browser as client. In such cases it is easier to instantiate a pre-defined template than going through the effort of generating a new security overview from scratch.

In addition, the limited technological options (HTTP, HTML, SQL, JavaScript, and possibly Flash are almost omnipresent) result in a relatively small number of possible tests that apply to most web applications (see e.g. [25]). While properly conducting such tests is far from trivial, it still questions the usefulness
of our vulnerability indicator idea. When neglecting the benefit of a business-risk-based traceability, going through a check list with the most common web tests may be more efficient while providing the same test coverage.

5.8 Summary and Future Work

In this chapter we discussed the lessons learned from the practical application of the risk assessment, initially proposed in Chapter 5 of NESSoS Deliverable 10.3 and further refined in [64].

Considering the lessons will allow a more efficient (Section 5.3, 5.6.1, and 5.7), more flexible (Section 5.1), and tool supported (Section 5.5.4) procedure with a higher industry acceptance factor (Section 5.2 and 5.4).

Besides the discussed needs for adaptation and extension, the main approach and its three key concepts have proven useful: The creation of a security overview based on static and dynamic binary analysis; the use of vulnerability indicators / patterns that point to possible vulnerabilities and suggest appropriate practical tests; and finally the set up and utilization of a test library with entries categorized according to Security Assurance Levels.

We currently work on the tool-support in form of IDA Pro python scripting snippets⁶ that will form the basic capabilities required for the identification of vulnerability indicators. Moreover, we define additional patterns using the more technical information included in the low-level part of the security overview.

Future work will target technical challenges such as obtaining and including operating system monitoring information in IDA Pro scripts and extending the security overview generation to managed-code languages (see Section 5.5.3). On a more conceptual level, we want to evaluate to what extent an expert system, e.g. PyCLIPS⁷, is a reasonable approach for matching the technical information from the security overview to tests or further analysis activities.

⁶https://code.google.com/p/idapython/
⁷http://pyclips.sourceforge.net/web/
6 Test-Driven Security Risk Assessment

In the development and management of secure software and services, security testing [56] and security risk assessment [36] are means to facilitate the identification of vulnerabilities and security flaws, or for making security assurance. Testing is the process of exercising the software/service system in question to verify that it satisfies specified requirements [68], whereas risk analysis involves the identification and evaluation of threats, vulnerabilities and incidents [31].

In this chapter we present methods and techniques for the combination of security testing and security risk assessment. During security testing, there is on the one hand the need to decide what to test and to develop the test cases, and on the other hand to prioritize and select among the cases; risk assessment may facilitate this by supporting the identification of possible security threats and vulnerabilities, and by supporting the prioritization of security test cases. The purpose here is to use risk assessment to support the security testing, which we refer to as risk-driven security testing. During a security risk assessment, security testing may facilitate the risk identification, as well as the risk estimation. The purpose here is to use testing to support the security risk assessment, which we refer to as test-driven security risk assessment. For further background and results, the reader is referred to [18, 19, 40].

The structure of this chapter is as follows. In Section 6.1 we describe our processes for combining testing and risk assessment, focusing on security and model-based approaches. In Section 6.2 we present test-driven security risk assessment in more details by going through a concrete example. Finally, in Section 6.3 we conclude.

6.1 Combining Testing and Risk Assessment

In this section we describe our processes for testing and risk assessment, referring to the standards on which the concepts and activities are based.

6.1.1 Model-Based Security Testing

Our testing process is depicted to the left of Figure 6.1, and consists of five consecutive phases. (Ignore for now the risk analysis part to the right of the figure.) The first phase is included in what is referred to as the test management process in ISO/IEC 29119, whereas the subsequent four are included in what is referred to as the dynamic test process [33]. The test planning is to plan for what to test (such as functionality, security, performance), and which part of the system to test. The test design and implementation are the activities of deriving the test cases and the test procedures. The test environment set-up and maintenance is to establish and maintain the environment in which the tests shall be executed. The test execution is to run the test procedure defined during the test design and implementation in the established test environment. Finally, the test incident reporting is the reporting and management of test incidents or test failures.

Our process for information security testing is based on the NIST guidelines 800-115 [56]. These guidelines define three phases of a security assessment and testing process, namely test planning, test execution and post test execution. Compared to the process depicted to the left of Figure 6.1, the test planning phases are corresponding, the test execution of the NIST process corresponds to the three activities in the middle of Figure 6.1, whereas the post test execution corresponds to the test incident reporting. Two of the main differences between the processes are, first, that the NIST process is specialized for information security, and, second, that the NIST process puts more emphasis on analyzing the system under test both during planning and execution.

Our approach to software testing is largely model-based. According to the ETSI standard ES 202 951 [22], model-based testing involves five main activities. Modeling for test generation is to define the (computer readable) test model from which the tests will be generated. The model describes how the system under test is intended to interact with its environment. Test selection is the selection of tests from the (possibly infinite) set of tests that can be derived from a model. Test generation is the automatic derivation of abstract test cases from a model. The derivation is based on user defined selection criteria. Test adaption is to derive concrete, executable test cases from the models. Finally, test execution is to execute the concrete test cases. Compared to the test process depicted in Figure 6.1, the test design...
correspond to the three first activities of the NIST standard, while test implementation corresponds to test adaption.

### 6.1.2 Model-Based Security Risk Analysis

Our risk analysis process, which is based on the ISO 31000 risk management standard [31], is depicted to the left of Figure 6.2, and consists of five consecutive phases. (Ignore for now the risk testing part to the right of the figure.) Establishing the context is to define the external and internal parameters to be accounted for when managing risk, and to set the scope and risk criteria for the risk management policy. Risk identification is to find, recognize and describe risks. Risk estimation is to comprehend the nature of risk and to determine the risk level. Risk evaluation is to compare the risk estimation results with the risk criteria to determine whether the risk and its magnitude are acceptable or tolerable. Risk treatment is the process of modifying the risk. The three phases in the middle are referred to as risk assessment.

Security risk assessment can be understood as a specialization of the ISO 31000 process, focusing on information security and properties such as confidentiality, integrity and availability of information and services. For such a specialization we refer to the ISO/IEC 27005 information security risk management standard [36], which in turn is based on ISO 31000.

By model-based or model-driven risk assessment, we refer to risk and threat modeling techniques that support one or more of the phases depicted in Figure 6.2. A wide number of techniques exist in particular for risk identification and risk estimation, such as fault tree analysis (FTA) [27], event tree analysis (ETA) [29], cause-consequence diagrams [46] and Bayesesian networks [11] to mention a few. In our approach to combine testing and risk assessment, we have largely based the risk assessment on the CORAS approach [41]. CORAS instantiates the ISO 31000 process, and is fully model-driven in the sense that models are used in all phases throughout the whole process.

### 6.1.3 Risk-Driven Security Testing

Risk-driven security testing is the systematic use of risk assessment to support and facilitate the testing process. This is illustrated in Figure 6.1 with the testing process to the left and the risk analysis process
6.1.4 Test-Driven Security Risk Assessment

Test-driven security risk assessment is the systematic use of testing to support and facilitate the risk analysis process. This is illustrated in Figure 6.2 with the risk analysis process to the left and the testing process to the right. Again, the supporting process is depicted as black boxes.

During the risk assessment, testing can on the one hand be used to facilitate the risk identification. Security testing can aid risk analysts in the identification of vulnerabilities, threat scenarios, unwanted incidents, and so forth. On the other hand, testing can be used during the later phases to the risk analysis to validate or correct the risk assessment results. In particular, testing can be used to check the estimates of likelihoods and consequences, and to do testing with respect to the existence or severity of vulnerabilities.

6.2 Example: Test-Driven Risk Assessment using CORAS

In this section we present and exemplify an instantiation of the test-driven security risk assessment process depicted in Figure 6.2. The instantiation is based in CORAS [41], and can be understood as an extension of CORAS in which testing is embedded as part of the method.

In this particular instantiation we make use of testing only after the completion of the risk evaluation, i.e. we omit the first testing step that is shown in Figure 6.2. More specifically, after the completion of the risk evaluation, we first do test identification and test selection / test prioritization based on the risk assessment results. Then we do test design, test implementation and test execution. The results of the testing is then fed back to the risk analysis process for risk validation, before the risk analysis is finalized with the risk treatment. This section is structured according to the instantiated process, which is shown in Figure 6.3.
6.2.1 Step 1: Establishing the Context

The activities of the first step are based on Step 1 through Step 4 of the CORAS method. The output of these activities includes a description of the target of analysis, a description of the assumptions, focus and scope of the analysis, CORAS asset diagrams documenting the assets with respect to which the analysis will be conducted, as well as the risk evaluation criteria.

The description of the target of analysis usually includes a large set of models in a suitable language such as the UML [49]. We will not go into the details of the target of analysis here, since our main focus is on the use of testing. The assets we consider are integrity user data, confidentiality of user data and availability of services.

The context establishment also includes the definition of the likelihood and consequence scales for risk estimation and risk evaluation. In this example we use a likelihood scale of the five levels of rare, unlikely, possible, likely and certain. Each of these denote a quantitative frequency interval. For example, possible is the likelihood of approximately once a year and denotes the interval \([5, 20]\) occurrences per ten years.

We use one consequence scale for each asset, where each scale is of the five levels of insignificant, minor, moderate, major and catastrophic. For the asset of availability, for example, moderate denotes the interval of \([30 \text{ min}, 6 \text{ hours}]\) of downtime.

The risk evaluation criteria are specified by the risk matrix depicted in Figure 6.4. We operate with three risk levels, namely low, medium and high. The low risks are acceptable and correspond to the white cells in the matrix. The high risks are unacceptable and correspond to the darkest cells. The medium risks should be monitored or possible evaluated further, and correspond to the cells shaded with lighter gray.

6.2.2 Step 2: Risk Identification

The risk identification corresponds to Step 5 of the CORAS method. The objective is to identify risks by the identification of threats, vulnerabilities, threat scenarios and unwanted incidents with respect to the identified assets. The risk identification is conducted by systematically going through the description of the target of analysis that was developed during the context establishment.

The risk identification is conducted and documented by active use of CORAS threat diagrams. A threat diagram is an acyclic graph with nodes and relations between them.
The nodes are of the following kinds. A threat, which is a potential cause of threat scenarios and unwanted incidents. A threat can be human or non-human, and the former can be accidental or deliberate. A threat scenario, which is a chain or series of events that is initiated by a threat and that may lead to an unwanted incident. An unwanted incident, which is an event that harms or reduces the value of an asset. Scenarios and incidents can be annotated with a likelihood. An asset, which is something to which a party assigns value, and hence for which the party requires protections.

The relations are of the following kinds. An initiates relation is from a threat to a threat scenario or unwanted incident, meaning that the former initiates the latter. The relation can be annotated with a likelihood. A leads-to relation is from a scenario or an incident to a scenario or an incidents, meaning that the former may lead to the latter. The relation can be annotated with a conditional likelihood. A harms relation is from an unwanted incident to an asset, meaning that the former causes harm to the latter when the former occurs. The relation can be annotated with a consequence value.

Initiates relations and leads-to relations can moreover be annotated with vulnerabilities. A vulnerability is a weakness, flaw or deficiency that opens for, or may be exploited by, a threat to cause harm to or reduce the value of an asset.

An example of a CORAS threat diagram is given in Figure 6.5. (Note that the likelihoods and consequences that are inserted in this example diagram are the result of the risk estimation step which comes next.) In this example there is one threat, namely adversary, which is a human and malicious (deliberate) threat. One of the possible threat scenarios is the SQL injection attack. Due to the vulnerability insufficient input validation, this threat scenario may eventually lead to the unwanted incident confidential user data disclosed, which has a major harm on the asset of confidentiality.

6.2.3 Step 3: Risk Estimation

The risk estimation corresponds to Step 6 of the CORAS method. The objective is to estimate risks by the estimation of the likelihood and consequence of each unwanted incident. To facilitate the likelihood estimation of unwanted incidents, and to identify the most important sources of risks, also the likelihoods of threat scenarios and the conditional likelihoods should be estimated and documented. The reasoning about likelihood is moreover facilitated by the CORAS calculus, both for reasoning about probabilities [41] and for reasoning about frequencies [71, 72]. Examples of risk estimation results are shown in Figure 6.5.

6.2.4 Step 4: Risk Evaluation

The risk evaluation corresponds to Step 7 of CORAS, and the objective is to evaluate the risks, prioritize them, and decide which risks should be evaluated further for possible treatment.

A risk is defined as the likelihood of an unwanted incident and its consequence for a specific asset. Hence, each pair of an unwanted incident and an asset constitutes a risk. In Figure 6.5, namely confidential user data disclosed or service unavailable. The former has likelihood unlikely and consequence major, and is therefore a risk at level medium. The latter has the estimates unlikely and moderate, which corresponds to risk level low. To give an overview of the risks and their risk levels, each identified risk is usually given an identifier and plotted into the risk matrix.
Step 5: Test Identification and Test Selection/Prioritization

Step 5 of our process is divided into two sub-steps, each of which is described in the following.

Test Identification

The objective of the test identification is to identify potential test scenarios based on the CORAS threat diagrams. There are in particular two kinds of CORAS diagram elements that should be considered during test identification. The first is the threat scenario element, because these elements give information about how to test. The second is the vulnerability element, because these elements give information about what to search for when testing.

In our model-based approach to test-driven security risk analysis, we aim to use testing for testing the threat models. One of the core questions is then what in the threat models to test. One option is to specify one test case or test scenario for each threat scenario and each vulnerability. For such a strategy, however, the information we have about the relationships between the threat scenarios and vulnerabilities are then lost and not taken into account during testing. A better option is therefore to use the threat diagram relations as the basis for the test identification. In particular, we use the initiates relations and the leads-to relations as a basis, because these are the CORAS threat diagram relations that involve both threat scenarios and vulnerabilities.

The resulting eight test scenarios are listed in Table 6.1. Such a textual specification of the test scenarios can be automatically extracted from the CORAS threat diagrams by the schematic mapping from CORAS diagrams to English prose [41].

Test Selection/Prioritization

The objective of the test selection/prioritization is to make a prioritization of the identified test scenarios and, based on this, select the scenarios that should be developed further into test cases. The output of this task is a prioritized list with the recommended selection.

The prioritization is made based on the information we have obtained during the risk assessment and documented in the threat diagrams. In our approach we use three different kinds of estimates to calculate the priority of a given test scenario, namely severity, testability and uncertainty.
<table>
<thead>
<tr>
<th>ID</th>
<th>Test scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS 1</td>
<td>Adversary initiates social engineering attack with likelihood possible.</td>
</tr>
<tr>
<td>TS 2</td>
<td>Adversary initiates SQL injection attack with likelihood likely</td>
</tr>
<tr>
<td>TS 3</td>
<td>Adversary initiates denial of service attack with likelihood possible</td>
</tr>
<tr>
<td>TS 4</td>
<td>Social engineering attack leads to adversary obtains user account credentials with conditional likelihood 0.3 due to vulnerability insufficient security awareness.</td>
</tr>
<tr>
<td>TS 5</td>
<td>SQL injection attack leads to successful SQL injection with conditional likelihood 0.1 due to vulnerability insufficient input validation.</td>
</tr>
<tr>
<td>TS 6</td>
<td>Denial of service attack leads to service unavailable with conditional likelihood 0.3 due to vulnerabilities insufficient robustness of protocol implementation and insufficient network capacity</td>
</tr>
<tr>
<td>TS 7</td>
<td>Adversary obtains user account credentials leads to confidential user data disclosed with conditional likelihood 1.</td>
</tr>
<tr>
<td>TS 8</td>
<td>Successful SQL injection leads to confidential user data disclosed with conditional likelihood 0.5.</td>
</tr>
</tbody>
</table>

Table 6.1: Test scenarios extracted from threat diagrams

The severity is an estimate of the contribution of the scenario on the overall risk level. A high severity indicates that the scenario potentially has a strong contribution to the risk level and that it should be prioritized. Different functions can be defined for calculating the severity, depending on how we wish to prioritize the test scenarios for the target of analysis in question. For example, one option is to use the results of the risk estimation alone. In that case the leads to relation from Hacker obtains user account credentials should have twice the severity of the relation from Successful SQL injection. Another option is to take into account the difference between the obtained estimates and the worst case scenarios if these estimates are wrong. In that case the resulting severity value prioritizes leads-to relations that are potentially much more severe than what is reflected by the estimate. Nevertheless, whatever the function we choose for calculating the severity, the calculation is done only on the basis of the estimates documented in the threat diagram.

The testability is an estimate of how testable a scenario is. A higher value indicates a higher priority. This is usually an estimate of the resources that are required to conduct the test, but we may also need to take into account other factors. For example, testing for social engineering attacks may not be advisable due to ethical factors.

The uncertainty is an estimate of the degree to which we are uncertain about the correctness of the results we obtained during the risk estimation. A high uncertainty indicates a higher priority, since it means that we need to obtain more information.

Whereas the severity estimates should be deduced from the threat diagrams, the testability and uncertainty estimates must be provided by the analysis team, including the risk analysts, the testers and the target owner/experts.

The threat scenarios ordered by priority are listed in Table 6.2. All estimates for severity (S), testability (T) and uncertainty (U) are made up for the sake of the example. The resulting priorities are calculated by making the product of S, U and T. Other functions for calculating the priorities can be used if the respective factors should be weighted differently for the risk assessment in question. In the example we have set the priority threshold to 10, i.e. all scenarios with priority 10 or above should be selected.

### 6.2.6 Step 6: Test Design, Implementation and Execution

The objective of this step is to execute the test cases that are derived from the selected test scenarios and to report on the results that will be fed back to the risk assessment. The process we have defined does not specify how the tests should be designed, implemented and executed. The test results should nevertheless be reported in a such a way that risk analysts can conveniently make use of them for validating or correcting the current risk estimates. For this we propose that the results are reported in a table format with the four following columns: 1) The test scenario ID, 2) a vulnerability (V) related to the test
Table 6.2: Test scenarios prioritization

<table>
<thead>
<tr>
<th>ID</th>
<th>S</th>
<th>T</th>
<th>U</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS 5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>TS 6</td>
<td>3.2</td>
<td>2</td>
<td>3</td>
<td>19.2</td>
</tr>
<tr>
<td>TS 4</td>
<td>1.5</td>
<td>1</td>
<td>3</td>
<td>4.5</td>
</tr>
<tr>
<td>TS 1</td>
<td>2.5</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>TS 2</td>
<td>2.5</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>TS 3</td>
<td>2.5</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>TS 7</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TS 8</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.3: Test result report

<table>
<thead>
<tr>
<th>ID</th>
<th>V</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS 5</td>
<td>Insufficient input validation</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>TS 6</td>
<td>Insufficient network capacity</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>TS 6</td>
<td>Insufficient robustness of protocol implementation</td>
<td>0.2</td>
<td>0.6</td>
</tr>
</tbody>
</table>

scenario in question, 3) the estimated likelihood (L) that the vulnerability exists, and 4) an estimate of the exploitability (E) indicating how hard it is to exploit the vulnerability in question. The latter is an estimate of the likelihood that an attempt to exploit the vulnerability will lead to the test scenario, where the test scenario corresponds to a threat scenario or an unwanted incident in one of the threat diagrams resulting from the risk assessment.

Table 6.3 gives an example of a test result report for the tests that were selected based on the threat diagram in Figure 6.5. For tests that confirm the existence of a vulnerability, the likelihood should be set to 1, as for insufficient input validation. For vulnerabilities that were not confirmed during the testing, we can be less conclusive since we have no evidence that they do not exist. For such cases we need instead to make an estimate based on our experience and other indicators. Such indicators may, for example, be the number of executed tests or a measure for test coverage.

### 6.2.7 Step 7: Risk Validation and Treatment

The objective of the last step is twofold. First, the risk assessment is finalized by revising the risk models based on the test results. Second, treatments for the unacceptable risks are identified and modeled.

In order to determine whether the risk models need to be revised, the risk analyst needs to check whether the test results are consistent with the current risk assessment results. How the test results should be compared with the risk assessment results may depend on how the tests were defined and executed, and at what level of abstraction. Recall that in our approach, each test scenario corresponds to a relation in the threat diagrams. For the CORAS leads-to relations, we can use the likelihood and exploitability estimates from the test report to make an estimate of the conditional likelihood associated with each vulnerability. The resulting conditional likelihood is then the sum of the product of likelihood and exploitability for each test scenario. Hence, the conditional likelihood for TS5 is then $1 \times 0.8 = 0.8$, and the conditional likelihood for TS6 is $(0.4 \times 0.7) + (0.2 \times 0.6) = 0.4$.

Referring to the CORAS threat diagram of Figure 6.5, the revision is listed in Table 6.4. The first column shows how the threat diagram is updated, whereas the second column gives the initial estimate for the likelihood estimate in question.

Considering the risk evaluation and recalling the risk matrix in Figure 6.4, the risk Service unavailable remains low. The risk Confidential user data disclosed, however, is after the revision defined as a high risk and should therefore be considered for risk treatment.

The risk treatment phase is outside the scope of our approach to test-based risk assessment and therefore conducted according to the standard CORAS method. The interested reader is referred to
Revised threat diagram element | Initial estimate
---|---
SQL injection attack leads to successful SQL injection with conditional likelihood 0.8 due to vulnerability insufficient input validation | 0.1
Threat scenario successful SQL injection occurs with likelihood likely | unlikely
Unwanted incident confidential user data disclosed occurs with likelihood possible | unlikely
Denial of service attack leads to service unavailable with conditional likelihood 0.4 due to vulnerabilities insufficient robustness of protocol implementation and insufficient network capacity | 0.3

Table 6.4: Revised risk assessment result

existing literature for a detailed description of this activity [41]. In our example, there are two attack paths that need to be considered for mitigating this risk, namely the social engineering and the SQL injection attack. For the latter attack, possible treatments are, for example, to implement/improve authentication mechanisms or to implement/improve the user input validation mechanism.

6.3 Conclusion and Ongoing Work

The objective of this chapter was to motivate the methodological and systematic combination of security risk assessment and security testing, and to introduce two different processes for such a combination. On the one hand, the two disciplines can be combined in order to conduct so-called risk-driven security testing. In this case security testing is the main objective, and risk assessment is used to support various phases in the testing process. On the other hand, they can be combined in order to conduct test-driven security risk assessment. In this case security risk assessment is the main objective, and testing is used as support. Our approach is model-based and builds on international standards.

In our ongoing work we are investigating and developing methods and tools for both combinations, although we have focused mostly in the test-driven risk assessment so far. In this work we have used methods for identifying and selecting test scenarios based on CORAS threat models, as described in this chapter.
7 UCON Policies Specification with Risk

The objective of the approach presented in this chapter is to tackle issues connected with attribute mining and policy specification for usage control (UCON) [38, 55]. In UCON, an access decision is made on the basis of rules and policies that consist of predicates over attributes. (Cf. our previous work presented in NESSoS D10.2 and D10.3.) We are going to determine the predicates (rules, policies) on the basis of risk. We assume that granting a permission to a user is connected with the risk that the user may misuse obtained privileges. Thus, rules should constrain the attribute values so as to obtain a permission in a way that there is no risk for a system owner or at least this risk is minimal.

Our work is inspired by works for role mining in role-based access control (RBAC) [54]. Basically, there are two main approaches to role mining [15, 24]: top-down and bottom-up. The top-down approach is based on the information about business processes, security policies and other business information to configure the roles. The bottom-up approach uses existing assignments between users and permissions such as access control lists (ACLs). In this chapter we consider the bottom-up approach trying to apply it to attribute mining in UCON. Further details are presented in our forthcoming publication [37].

7.1 Attribute Mining Problem

The problem of attribute mining in UCON can be defined similarly to role mining in RBAC. In this section we briefly discuss each of them in turn.

7.1.1 Role Mining in RBAC

Suppose there are sets of user (referred to as "subjects" in the following) \( SBJS \) and permissions \( PRMS \) in a system. The role mining result is a triple \( RC = (ROLES, SA, PA) \) where \( ROLES \) is the set of assigned roles, \( SA \subseteq SBJS \times ROLES \) is a subjects to roles assignment, \( PA \subseteq PRMS \times ROLES \) is a permissions to roles assignment. Sometimes the role mining result is delivered together with direct users to permissions assignment \( DSPA \). The subject can play several roles and several roles can give a permission. The role mining schema is presented in Figure 7.1.

![Figure 7.1: Role mining in RBAC](image)

7.1.2 Attribute Mining in UCON

For UCON we consider a set of attributes \( ATTRS \) instead of roles and the set of access decisions \( DCS \) instead of \( PRMS \). Moreover, we consider three additional sets of objects \( OBJS \), environments \( ENVS \) and actions \( ACTS \) together with subjects \( SBJS \). We introduce a new set of entities \( ETTS \) such that it contains all subjects, objects and environments:

\[
ETTS = SBJS \cup OBJS \cup ENVS \cup ACTS
\]

(7.1)

An attribute \( attr \in ATTRS \) can be seen as a function of entities:

\[
attr : ETTS \rightarrow D_{attr}
\]

(7.2)

where \( D_{attr} \) is a domain of the attribute \( attr \). Therefore in UCON there is not only subjects to attributes assignment \( SA \) similar to role mining but two more assignments of objects to attributes \( OA \), environments to attributes \( VA \) and actions to attributes assignment \( TA \). Further, we consider these assignments as parts of single entities to attributes assignment \( EA \).
The attributes are usually combined into rules and policies which are rules of rules (in the following we use “rules” both for “rules” and “policies” for the sake of brevity). The rules determine whether a subject is allowed to execute an action on an object in a current environment. Thus, not only a subject and her attributes determine the access decision. In UCON the access decision depends on attributes of several entities in a system. There are additional assignments between attributes and rules \( RA \) and rules and access decisions \( DA \).

An attribute mining result is a quadruple \( AC = (\text{ATTRS}, EA, RA, DA) \). There is an attribute mining schema in Figure 7.2. The “Rules (Policies, Etc)” box can be considered as a complex construct. For example, in XACML open standard [48] attributes can be combined into rules, rules can be combined into policies, and policies can be further combined into complex policies during a computation of an access decision. All these combinations can be considered as additional assignments. Here we do not consider such assignments for the sake of generality.

![Figure 7.2: Attribute mining in UCON](image)

For attribute mining in UCON a pure bottom-up approach is not possible. An attribute is a function of a subject, an object, an environment or an action. Thus, an attribute is information that usually cannot be determined only on the basis of existing permissions assignment. An attribute should be determined on the basis of additional information like the type of an object. The approach to attribute mining in UCON becomes hybrid i.e. combines techniques of top-down and bottom-up approaches. Note, that a pure top-down approach is still possible for UCON.

### 7.2 Policy Specification Problem

We assume that the \( EA \) assignment is done, thus, attributes are identified, however rules are not set. Thus, we focus on the issues of \( RA \) and \( DA \) assignments which we call a policy specification problem. We use the notion of risk to tackle the problem. A subject may abuse her privileges during the access and cause damage to a system owner. Risk helps to specify conditions when a subject can access an object in a way that there is no risk or risk is minimal.

Solving the general policy design problem is very difficult because real UCON implementations like XACML and U-XACML [39, 48] contain lots of functions to compose complex rules and policies. Evaluation of risk in this case is not trivial. Thus, we start with a simple case of a single rule of a single attribute. Then we briefly discuss how this case can be extended for complex rules and policies.

#### 7.2.1 The Case of Single Attribute

We start with a simple case of a rule of a single attribute and limited set of combining functions. In this case, the domain of attribute values \( D_{\text{attr}} \) can be divided into two sets: values \( D_{\text{G}} \) that allow access and values \( D_{\text{B}} \) that forbid it. A natural way to separate these two domains is to weight risks and benefits of possible access decisions. Such analysis could be performed by policy writers on a subjective basis.

Consider an attribute \( \text{attr} \) and domain of its values \( D_{\text{attr}} \). Assume that there is a function \( \Pr_{\text{vio}} : D_{\text{attr}} \rightarrow [0, 1] \) which gives a probability of a policy violation caused by granting access to a user when \( \text{attr} \) has value \( d \in D_{\text{attr}} \). Such function depends on attribute and its domain and should be identified by experts.

We assume that a system owner obtains utility granting an access to an object. Moreover, we assume that there are no utilities obtained when an access is denied. If we have the cost of abusing the granted access \( U^B \) (which is negative utility) and the gain of granting access \( U^G \) (which is positive utility) then the
average utility $U$ of granting access when attribute takes a value from a sub-domain $D^G_{attr} \subseteq D_{attr}$ is:

$$\langle U^G_{D^G_{attr}} \rangle = \sum_{d \in D^G_{attr}} \Pr_{occ}(d) \cdot ((1 - \Pr_{vio}(d)) \cdot U^G + \Pr_{vio}(d) \cdot U^B) \quad (7.3)$$

where $\Pr_{occ}(d)$ is the probability of occurrence of value $d$, summand $\Pr_{vio}(d) \cdot U^B$ is risk of granting a permission if the attribute value is $d$, summand $(1 - \Pr_{vio}(d)) \cdot U^G$ is benefit of granting the permission. We assume that probability $\Pr_{occ}(d)$ could be taken from statistics. For example, if transitions from one value to another can be modeled with Markov Process, then these probabilities could be seen as steady probabilities.

If we would like avoid risk then the problem is:

$$\langle U^G_{D^G_{attr}} \rangle > 0 \quad (7.4)$$

In this case, domain of good values $D^G_{attr}$ just affects the number of positive summands in Equation 7.3 because we do not allow the accesses that lead to losses. Probability $\Pr_{occ}(d)$ does not impact whether a summand is positive or negative. Thus, to find the values of $D^G_{attr}$ we should find the threshold probability $\Pr_{vio}^+$ solving the following equation:

$$(1 - \Pr_{vio}^+) \cdot U^G + \Pr_{vio}^+ \cdot U^B = 0 \quad (7.5)$$

Trivially, the solution of the equation is:

$$\Pr_{vio}^+ = \frac{U^G}{U^G - U^B} \quad (7.6)$$

Note, that $U^B < 0$ because it is negative utility. Thus, $0 \leq \Pr_{vio}^+ \leq 1$.

The domain of good values is composed of values $d$ in a following way:

$$D^G_{attr} = \{ d : \Pr_{vio}(d) < \Pr_{vio}^+ \} \quad (7.7)$$

The domain of bad values is:

$$D^B_{attr} = D_{attr} \setminus D^G_{attr} \quad (7.8)$$

If we would like to maximize the average utility (to minimize risk) then the problem is:

$$\arg max_{\forall D^G_{attr}} \langle U^G_{D^G_{attr}} \rangle \quad (7.9)$$

### 7.2.2 The Case of Several Attribute

The case of several attributes is more complex. Probability of violation $\Pr_{occ}(g(attr_1, attr_2, \ldots, attr_n))$ and probability of occurrence $\Pr_{vio}(g(attr_1, attr_2, \ldots, attr_n))$ should be computed for a function $g$ of several attributes $attr_1, attr_2, \ldots, attr_n \in ATTRS$. The function $g$ can be determined on the basis of XACML rules and combining algorithms. If domains for the function $g$ are $D_g, D^G_g, D^B_g$ then the problem for avoiding the risk is:

$$\langle U^G_g \rangle > 0 \quad (7.10)$$

The problem for maximizing the utility is:

$$\arg max_{\forall D^G_g} \langle U^G_g \rangle \quad (7.11)$$

We leave a solution of these problems as a future work.
7.3 Conclusion

In this chapter we have addressed the attribute mining problem and the policy specification problem in UCON. Moreover, we showed how the policy engineering problem can be solved using risk as a simple case of a rule of a single attribute.

The following directions can be considered as a future work. First, a method is required to identify probabilities and utilities for an outcome of access decision. An existing method like CORAS [41] could be applied. Second, complex policies should be considered. Current implementations of UCON like XACML consist of tenth of rules; a policy specification method should be applicable to each of them. Another issue here is that although there several implementations of UCON, there is still no formal model. Third, a method for initial identification of attributes is needed.
8 Foundations for Compositional Security Risk Assessment

Risk assessment [31] includes understanding, modeling and documenting what can go wrong for a given target of analysis, how likely it is and how severe the harm is when it happens. In the following we let $T$ denote the target of analysis and $R$ denote the risk documentation. Using a model-based approach, we refer to the latter as the risk model.

The target of analysis is the actual system we wish to assess. The target therefore includes the system services, roles, behaviors, actors and so forth. The risk model that we develop during a risk assessment can be understood as an abstraction of the target in which we model only the behavior that is relevant for understanding the risk, threats, vulnerabilities, incidents, etc. at an appropriate level of details. Risk modeling is hence a technique for capturing, understanding and assessing precisely the parts or aspects of the target that we are interested in from a risk management perspective. An important issue in any risk assessment is therefore the correctness of the risk model. In other words, when conducting a risk assessment, we need somehow to demonstrate or substantiate that $R$ is a correct specification of $T$.

For most risk assessments only selected parts or aspects of a system are addressed. This is because it is often infeasible or too costly to conduct one assessment of the whole system at the same time. For such assessment we make use of established methods and techniques (e.g. [2, 11, 27, 28, 29, 30, 31, 41, 65]) to verify or substantiate the correctness of $R$. Such a traditional approach is fine when we can reach an adequate understanding of the risks by assessing separate parts of the target individually. However, for large, complex systems we may need to consider all parts of the target in combination in order to adequately understand the full risk picture. Taking into account the infeasibility of addressing the full system at once, we need novel techniques for a sound and systematic composition of separate risk models in order to deduce risk information that is valid for the combined target.

More specifically, such techniques involves the decomposition of the target of analysis $T$ into individual parts $T_1, \ldots, T_n$ and assessing each of them separately to build the risk models $R_1, \ldots, R_n$, respectively. For each such part $T_i$ we use traditional methods and techniques to build and verify the correctness of $R_i$, $i \in \{1, \ldots, n\}$. Novel techniques are then needed for deducing the correct risk model $R$ based on $\{R_1, \ldots, R_n\}$ and the composition of $T_1, \ldots, T_n$.

The aim of the work presented in this chapter is to define the principles of compositional risk assessment, and to build a formal foundation for the approach. To the best of our knowledge, there are no existing risk assessment techniques that supports compositionality in the sense the notion is understood in the formal methods community. As our first steps towards a well-defined notion of compositional risk assessment we build closely on the principles established in other domains. For the formal specification and reasoning about risks, we build on our previous work presented in NESSoS deliverables D10.2 and D10.3.

The structure of this chapter is as follows. In Section 8.1 we discuss the notions of composition and compositionality, and how the principles of compositionality can be applied to risk assessment. In Section 8.2 we introduce our formal representation of the target of analysis, and in Section 8.3 we introduce our formalization of the risk models, both syntax and semantics. In Section 8.4 we introduce and formalize our composition operators, before we conclude in Section 8.5.

8.1 Composition and Compositionality

Composition is well-established and much used in software and service engineering and analysis. It is used in different disciplines such as requirements engineering, business modeling, architecture, design and implementation. Modeling languages come with composition constructs such as AND/OR composition in goal-oriented requirements engineering [45], parallel and sequential composition in the UML [49] and AND/OR gates in fault trees [27]. When building models and specifications, composition is a language feature that allows the specification of different parts separately with means to combine them for the purpose of capturing the desired overall specification.

In the setting of risk assessment and risk modeling, an important challenge in developing a compositional approach is the need to take into account dependencies that may exist between different parts of
the target. Such dependencies can affect the combined risk picture in unforeseen ways, and therefore need to be identified and resolved during the composition. For example, when combining services from $T_1$ and $T_2$, we may increase redundancy sufficiently for previously unacceptable risks of availability to become acceptable. On the other hand, composition of services may yield new threats due to the combination of different vulnerabilities.

Our notion of compositional risk assessment, and the principles of compositionality that we aim for, is based on the corresponding concept in program verification as established in the formal methods community. In this setting, de Roever et al. define compositionality as follows: “That a program meets its specification should be verified on the basis of specifications of its constituent components only, without additional needs for information about the interior construction of those components” [17]. In the setting of risk assessment we are concerned about the correctness of the risk model $R$ with respect to the target $T$. Hence, the compositional risk assessment means the following: That a risk model is correct should be verified on the basis of separate parts of the target only, without additional needs for information about the interior construction of these parts.

Compositional risk assessment should be founded on a compositional proof method. Such a method basically consists of two parts, namely a basic technique and a compositional proof technique. The former is for verifying that a risk model $R_i$ is correct for a respective target $T_i$ that is not decomposed further. The latter is for handling the case that $T$ is composed of the parts $T_1, \ldots, T_n$ for a given composition construct. For given composition constructs, a core challenge of developing a compositional approach to risk assessment is to develop compositional proof rules of the following form: From $R_i$ is correct with respect to $T_i$ and $\ldots$ and $R_n$ is correct with respect to $T_n$, conclude that $R$ is correct with respect to $T$.

An important principle of compositionality is that the application of the proof rule shall not require any reference to or further investigation of the internal construction of the different parts $T_i$ of the target; this shall be entirely left to the application of the basic technique. Nevertheless, the applicability and soundness of the compositional proof rules rely in general on some additional conditions that must be satisfied. In order to adhere to the principles of compositionality, these conditions may only be of statements about the target as captured by the risk model, which is our abstract specification.

To put it in a more structure way, a compositional approach to risk assessment requires the development of compositional proof rules of the following form for some $n$-ary composition construct $\oplus$.

\[
\begin{align*}
\text{for } i = 1, \ldots, n: & \quad R_i \text{ is correct with respect to } T_i \\
R & \text{ follows from some combination of } R_1 \ldots R_n \\
& \text{[additional conditions upon } R_1, \ldots, R_n \text{]} \\
\hline
R \text{ is correct with respect to } & \oplus(T_1, \ldots, T_n)
\end{align*}
\]

In developing such rules there are a number of issues that we need to address. We need to determine how the target $T$ and the risk model $R$ should be specified and formalized; how the notion $R$ is correct with respect to $T$ should be captured and formalized; what kind of target composition and decomposition we can support; and what kind of additional conditions we need when defining the compositional proof rules.

In our current work we are building the formal foundation that we need as a rigorous basis for tackling issues such as these. Such a foundation should provide our means for giving formal soundness proofs of the compositional proof rules. The soundness proofs are our techniques as method developers to ensure and demonstrate the soundness of the approach. For risk analysts using the approach, i.e. the end-users, the applicability of the compositional proof rules requires the premises to hold. As mentioned above, this must be substantiated using existing, traditional risk assessment methods and techniques. An important practical difference from formal methods is that while the latter aims to formally prove that a program meets its specification, the same cannot generally not be expected in risk assessment. Risk analysts rather use empirical methods, triangulation, testing, simulation, etc. to substantiate that the premises hold. The compositional proof rules should then be applicable when the analysts have sufficient confidence that the premises hold.

When building our formal foundation, we assume a model-driven approach in which the risk model is represented in terms of risk graphs [13]. The reader is referred to the NESSoS deliverables D10.2 and D10.3 for further introduction to risk graphs, which can be understood as a common abstraction of several risk modeling techniques (such as fault trees [27], event trees [29], attack trees [57], cause-consequence diagrams [46], Bayesian networks [11] and CORAS threat diagrams [41]). Our strategy
is to develop techniques and rules that apply to risk graphs and that can be instantiated in other risk modeling techniques. The purpose of this is to ensure a more general applicability of the techniques than by developing them for one specific and more concrete risk modeling language alone. While we do not assume any specific notation for modeling the target of analysis, we represent it formally by a set of execution traces. Such traces capture possible execution histories, and may represent an existing, running target system or a (model-based) specification of the target, for example a UML [49] model.

In the following chapters we introduce our formal foundation. The reader is referred to our technical report for the full presentation and all formal definitions [72]. The main results of the report are presented in a paper [71].

8.2 Representing the Target of Analysis

The target of analysis is represented by a set of traces \( H \), where each element \( h \in H \) represents an execution history. A history is an infinite sequence of timed events that is ordered by time and that progresses beyond any finite point in time. We let \( E \) denote the set of all events. The set of all timestamps is defined by \( T \equiv \mathbb{R}^+ \), where \( \mathbb{R}^+ \) denotes the set of non-negative real numbers. A timed event is an element \((e, t) \in E \times T\). Hence, the set of all histories \( H \) is the subset of \((E \times T)^\infty\) in which the events are ordered by time, and where time always progresses beyond any finite point in time. For a set of elements \( A \), \( A^\infty \) denotes the set of all infinite sequences over \( A \).

In order to specify and reason about risk, we need to represent likelihoods. In earlier formalizations for risk graphs we have used probabilities to capture likelihoods, and we developed a calculus for probability reasoning in risk graphs (see [13] and NESSoS deliverable D10.2). In our ongoing work we have introduced frequencies as an alternative to probabilities, and developed a calculus for the sound reasoning about frequencies in risk graphs [72]. Operating with probabilities is useful in order to support formal reasoning because of the well-defined probability theory. However, when conducting security risk assessment in practice, probabilities can be hard to obtain and are often difficult for stakeholder to relate to.

To capture frequencies we set the time unit as equal to 1. For simplicity we assume that all frequencies are per time unit. The set of frequencies \( F \) is therefore defined by \( F \equiv \mathbb{R}^+ \). This means that \( f \in F \) denotes the frequency of \( f \) occurrences per time unit.

8.3 Representing the Risk Model

Risk graphs are a risk modeling technique for specifying how scenarios may lead to other scenarios, where each scenario is either a threat scenario or an unwanted incident. Scenarios are represented by vertices, and a relation from one vertex to another means that the former may lead to the latter. A vertex can be assigned a frequency \( f \), which is a specification of the likelihood for the scenario to occur. A real number can be assigned to relations between vertices to capture the statistical relationship between them, i.e. the extent to which one scenario will lead to another. An example of the risk graph notation using frequencies is given in Figure 8.1.

8.3.1 Risk Graph Syntax

Formally, a risk graph is a pair of two sets \((V, R)\) where \( V \subseteq P(E) \times F \) and \( R \subseteq V \times \mathbb{R}^+ \times V \). For a set of elements \( A \), \( P(A) \) is the powerset of \( A \). We refer to the elements of \( V \) as vertices and the elements of \( R \) as relations. We use \( v(f) \) to denote a vertex, while \( v \rightarrow v' \) denotes a relation.

The set of vertex expressions is the smallest set \( X_V \) such that \( P(E) \subseteq X_V \), and \( v, v' \in X_V \Rightarrow v \sqcup v' \in X_V \wedge v \sqcap v' \in X_V \).

We define the function \( s \in X_V \rightarrow P(E) \) that for any vertex expression yields its set of events. Formally, \( s \) is defined as follows.

\[
s(v) \equiv \begin{cases} 
    v & \text{if } v \in P(E) \\
    s(v_1) \cup s(v_2) & \text{if } v = v_1 \sqcup v_2 \\
    s(v_2) & \text{if } v = v_1 \sqcap v_2
\end{cases}
\]
8.3.2 Risk Graph Semantics

Semantically, a vertex formula of the form \( H \vdash v(f) \) is a statement about the frequency of the occurrences in \( H \) of the events in \( v \). The semantics of a formula of the form \( H \vdash v \xrightarrow{r} v' \) is a statement about the statistical relationship between events in \( v \) and \( v' \) as they occur in \( H \). The statistical relationship is stated by the number \( r \). For a given \( H \), the semantics of a risk graph formula is therefore one of the Boolean values True and False. The semantics of a risk graph specification \( H \vdash (V, R) \) is the conjunction of the semantics of each risk graph formula in \( V \cup R \).

Formally, the semantics of a risk graph formula is derived by the function \([\ ]\) that takes the formula and yields a Boolean value. Before we define the semantics we need to introduce some functions on traces.

By \( A^* \) we denote the set of all finite and infinite sequences over \( A \), and by \( A^\omega \) we denote the set of all finite sequences. \( N_0 \) denotes the set of natural numbers including 0. The function \( \#_\in A^\omega \rightarrow N_0 \cup \{\infty\} \) yields the length of a sequence. The operator \( \_\ominus\in \mathbb{P}(A) \times A^\omega \rightarrow A^\omega \) is a filtering operator for filtering out elements; \( A\ominus a \) denotes the subsequence obtained from \( a \) by removing all elements in \( a \) that are not in the set \( A \). Finally, the function \( \_\in H \times T \rightarrow (E \times T)^* \) is for truncating histories. The operator captures the prefix of a history up to and including a certain point in time. Hence, \( h|_t \) is the maximal prefix of \( h \) the timestamps of which all are less than or equal to \( t \).

For \( v \in \mathbb{P}(E) \), the semantics is then defined as follows.

\[
[H \vdash v(f)] \overset{\text{def}}{=} \forall h \in H : f = \lim_{t \rightarrow \infty} \frac{\#((v \times T)\ominus (h|_t))}{t}
\]

Hence, the semantics of \( H \vdash v(f) \) is the statement that the number of occurrences per time unit in \( H \) of the events in \( v \) equals \( f \). The semantics of any other risk graph formula is defined recursively and with the similar interpretation of statements made about the occurrences of events in \( H \) and their statistical relationships. The reader is referred to the technical report [72] for the definitions.

We refer the reader to the technical report also for the calculus with rules for reasoning about frequencies in risk graphs. In the report we moreover generalize the syntax, semantics and calculus to support the specification and reasoning about countermeasures (risk treatments) and consequences, and to allow the specification of intervals when exact values for frequencies and consequences cannot be obtained during a risk assessment.

8.4 Target Composition

In this section we introduce our operators for composition of the target of analysis. Target composition is the last part of the foundation for compositional risk assessment as presented in this chapter.
Given the trace sets \( H \in \mathcal{P}(\mathcal{H}) \setminus \emptyset \) that we use to represent the target of analysis, we now define composition operators \( \oplus \) to facilitate the composition \( \oplus(H_1, \ldots, H_n) \). In the following we introduce binary composition operators. By associative and commutative properties the operators can be used also for multiple decompositions. Note that in our ongoing work, we are exploring also further possible operators for target composition.

### 8.4.1 Parallel Composition

For trace sets \( H_1 \) and \( H_2 \), the parallel composition \( H_1 \parallel H_2 \) yields the set \( H \) in which each trace is an interleaving of one trace from \( H_1 \) and one trace from \( H_2 \).

Before giving the formal definition, we need to introduce some new functions. We use \( \pi_i \) to extract the \( i \)-th element of a tuple. Hence, for the tuple \((a_1, a_2)\), we have \( \pi_1(a_1, a_2) = a_1 \) and \( \pi_2(a_1, a_2) = a_2 \).

For any set of pairs of elements \( P \) and pair of sequences \( t \), by \( P \circ t \) we denote the pair of sequences obtained from \( t \) by truncating the longest sequence in \( t \) at the length of the shortest sequence in \( t \) if the two are of unequal length; for each \( j \in \{1, \ldots, k\} \), where \( k \) is the length of the shortest sequence in \( t \), selecting or deleting the two elements at index \( j \), depending on whether the pair of these elements is in the set \( P \). For example, we have that \( \{(1, f), (1, g)\} \circ ((1, 1, 1, 1), (f, f, f, g, g)) = ((1, 1, 1), (f, f, g)) \). See [59] for the formal definition.

Parallel composition \( \_ \parallel \_ \in \mathcal{P}(\mathcal{H}) \times \mathcal{P}(\mathcal{H}) \to \mathcal{P}(\mathcal{H}) \) is now defined as follows.

\[
H_1 \parallel H_2 \equiv \{h \in \mathcal{H} | \exists s \in \{1, 2\}^\infty : 
\pi_2(\{1\} \times (E \times T)) \circ(s, h) \in H_1 \land 
\pi_2(\{2\} \times (E \times T)) \circ(s, h) \in H_2\}
\]

Parallel composition is associative and commutative.

### 8.4.2 Sequential Composition

For trace sets \( H_1 \) and \( H_2 \), the sequential composition \( H_1; H_2 \) yields the set \( H \) in which each trace is the concatenation of one trace from \( H_1 \) with one trace from \( H_2 \).

The function \( \_ \circ \_ \in A^\omega \times A^\omega \to A^\omega \) is for trace concatenation, i.e. for gluing them together. Hence \( s_1 \circ s_2 \) is the sequence of length \#s_1 + \#s_2 \) that equals \( s_1 \) if \( s_1 \) is infinite, and is prefixed by \( s_1 \) and suffixed by \( s_2 \) otherwise.

Recall that the set of histories \( \mathcal{H} \) are infinite sequences. Executions that terminate are therefore represented by histories \( h \) whose events at one point and beyond are terminating events \( \tau \). When the histories \( h_1 \) and \( h_2 \) are concatenated and the former terminates, the former is truncated at the last event before the terminating event prior to concatenation. The timestamps of the latter are incremented with the timestamp of the last event before the first terminating event of the former.

The function \( \text{term}(\_ \in \mathcal{H} \to T) \) yields the timestamp of the terminating event of a history; the function is undefined when there is no terminating event. The function \( \text{inc}(\_ \in \mathcal{H} \times \mathbb{R}^+ \to \mathcal{H}) \) increases all timestamps in a trace with a positive real number.

Sequential composition \( \_; \_ \in \mathcal{P}(\mathcal{H}) \times \mathcal{P}(\mathcal{H}) \to \mathcal{P}(\mathcal{H}) \) is now defined as follows.

\[
H_1; H_2 \equiv \{h \in \mathcal{H} | \forall h_1 \in H_1, h_2 \in H_2 : h = h_1 \text{ when } \text{term}(h_1) \text{ is undefined}
\land
h_1|_{\text{term}(h_1)} \circ \text{inc}(h_2, \text{term}(h_1)) \text{ otherwise}\}
\]

Sequential composition is associative.

### 8.4.3 Non-deterministic Choice

For trace sets \( H_1 \) and \( H_2 \), non-deterministic choice is the composition \( H_1 \sqcup H_2 \) where the histories are from any of the composed sets.

Non-deterministic choice \( \_ \sqcup \_ \in \mathcal{P}(\mathcal{H}) \times \mathcal{P}(\mathcal{H}) \to \mathcal{P}(\mathcal{H}) \) is therefore defined as follows.

\[
H_1 \sqcup H_2 \equiv H_1 \cup H_2
\]

This composition operator is associative and commutative.
8.4.4 Features

When applicable, the standard composition operators defined above may be useful in the setting of risk assessment. However, such bisected decompositions may not be adequate for the way different parts of a system are addressed in practice. Instead, one assessment may be concerned with certain actors and services regarding certain aspects of the system, while another may be concerned with a different set of actors and services. Yet, the two assessments may overlap in many ways. For example, one assessment may address the service consumers of a web-application and focus on the confidentiality of certain information, while another assessment may address the service provider and focus on integrity. A further example is the decomposition of a system according to different views [32], such as an information viewpoint, an enterprise viewpoint and a technology viewpoint. If individual risk assessments are conducted for each view, specific compositional techniques are needed for deducing the combined result.

When applicable, the standard composition operators defined above may be useful in the setting of risk assessment. However, such bisected decompositions may not be adequate for the way different parts of a system are addressed in practice. Instead, one assessment may be concerned with certain actors and services regarding certain aspects of the system, while another may be concerned with a different set of actors and services. Yet, the two assessments may overlap in many ways. For example, one assessment may address the service consumers of a web-application and focus on the confidentiality of certain information, while another assessment may address the service provider and focus on integrity. A further example is the decomposition of a system according to different views [32], such as an information viewpoint, an enterprise viewpoint and a technology viewpoint. If individual risk assessments are conducted for each view, specific compositional techniques are needed for deducing the combined result.

In building our formal foundation for compositional security risk assessment, we use the notion of feature to refer to such parts of aspects of a system. Formally, given the trace set feature for each view, specific compositional techniques are needed for deducing the combined result. Point, an enterprise viewpoint and a technology viewpoint. If individual risk assessments are conducted for each view, specific compositional techniques are needed for deducing the combined result.

For a set of elements $A$ the sub-trace relation $\_ \prec \_ \in A^\omega \times A^\omega \rightarrow \text{Bool}$ is defined as follows.

$$s_1 \prec s_2 \equiv \exists s \in \{1, 2\}^\infty : \pi_2(((\{1\} \times A) \oplus (s, h_2)) = h_1$$

Hence, we have, for example, that $\langle b, f, g \rangle$ is a sub-trace of $\langle a, b, c, d, e, f, g, h \rangle$, whereas $\langle b, f, i \rangle$ and $\langle b, g, f \rangle$ are not.

Whether a feature $F$ is represented by a set of events or a set of sub-traces, we use $F \times H$ to denote the target $H$ with respect to the feature $F$. In both cases, the feature specifies a subset of $H$.

8.4.5 Risk Graph Semantics for Target Composition

Having introduced composition operators for the representation of the target, we need to define the semantics of risk graphs with respect to composition.

For parallel composition, sequential composition and non-deterministic choice this is straightforward. For each of these operators, the result of composing two sets of histories is a set of histories. Hence, the definition of the semantics is as defined in Section 8.3.2.

For a feature $F$, $F \times H$ specifies a subset of $H$. We could therefore choose to define the semantics directly for the resulting subset. However, in order to keep the information about the different features and the target systems they refer to, we need to represent them as a pair of events/sub-traces and histories in our formalism.

When $F$ is a set of timed events, i.e. $F \in \mathbb{P}(E \times T)$, the risk graph is a set of statements about the traces of $H$ in which the events in $F$ occur. For the definition we need to introduce the function $s[n] \in A^\omega \times \mathbb{N} \rightarrow A$ to yield the $n$th element of a sequence. For a sequence $s$, $s[n]$ yields the $n$th element of $s$ if $n \leq \#s$.

We now define the semantics of $F \times H$ as follows when $F$ is a set of timed events.

$$[F \times H \vdash v(f)] \overset{\text{def}}{=} \forall h \in H : (\exists n \in \mathbb{N}, e \in F : h[n] = e \Rightarrow f = \lim_{t \rightarrow \infty} \#((v \times T) \oplus (h|_t)))$$

When $F$ is a set of traces, the risk graph is a set of statements about the traces of $H$ that have sub-traces in $F$. The definition is as follows.

$$[F \times H \vdash v(f)] \overset{\text{def}}{=} \forall h \in H : (\exists h' \in F : h' \prec h \Rightarrow f = \lim_{t \rightarrow \infty} \#((v \times T) \oplus (h'|_t)))$$

Like in Section 8.3.2, the semantics of any other risk graph formula with respect to a feature is defined recursively.
8.5 Conclusion and Ongoing Work

The objective of this chapter was to present our formal foundation for a compositional approach to security risk assessment. Based on this foundation, our goal is to develop a set of compositional proof rules for the sound composition of risk assessment results from different risk models.

By using binary composition operators $\oplus$, the compositional proof rules we develop in our ongoing work are of the following form.

\[
\frac{H_1 \vdash (V_1, R_1) \quad H_2 \vdash (V_2, R_2) \quad \text{additional conditions}}{H_1 \oplus H_2 \vdash (V_1 \cup V_2, R_1 \cup R_2)}
\]

The underlying formalism presented in this section is rather technical, but serves as an important tool for us as method developers to ensure rigor and prove the soundness of the approach. End-users, i.e. risk analysts, should apply the rules we develop directly on the models they create. What the risk-analysts need are therefore methods and guidelines for how to apply the rules to enable a compositional risk analysis of large scale systems.

In order to ensure the applicability and usefulness of the approach we are in our ongoing work also developing realistic use case scenarios to evaluate and validate the approach.
9 Interactions and Collaborations

NESSoS WP10 is a transversal work package in the sense the methods and techniques for security risk and cost assessment span over the SDLC. In this chapter we give a brief overview of the relations to the other technical work packages, as well as relevant collaborations.

WP6: Security requirements for services. One of the important challenges for security risk management of Future Internet system is how to deal with evolution and changes. The approach to evolution modeling presented in Chapter 4 makes use of previous work on security requirements evolution [70] on the specification of evolution rules. We have also published further work on tool-supported security risk analysis of evolving systems that orchestrates models-based requirements engineering and security risk assessment. This topic was presented in D10.3, but two more recent publications have followed [12, 61].

WP7: Security service architecture and design. The development of the ISMS-CORAS [10] method for establishing and documenting ISO 27001 ISMS as presented in Chapter 2 started out as a WP7-WP10 collaboration to explore the pattern-based cloud analysis described in D7.3 and D7.4 to support the security risk assessment prescribed by the standard. Currently, the two approaches complement each other, and we plan to leverage on this by combining the two to facilitate the establishment and documentation of ISMS for cloud computing systems.

WP8: Programming environments for secure and composable services. No new WP8-WP10 interactions have been fostered during the third year of the project. We have, however, very recently started to develop techniques for using security metrics to support the security risk assessment. In WP10 we are in particular investigating security indicators for risk monitoring (cf. D10.3), which could be supported by the monitoring infrastructure of WP8.

WP9: Security assurance for services. The current WP10 work on security metrics is also related to this topic in WP9. In WP10 we use metrics to aggregate low-level data gathered from active testing or passive testing (monitoring) to derive the more high-level data that is needed for the risk assessment. More generally, the WP10 work on testing presented in Chapter 5 and Chapter 6 has strong relations to testing in WP9. WP10 is developing methods, techniques and tools for combining risk assessment and testing, both for test-based security risk assessment and risk-based security testing. The purpose of the former is to use testing to facilitate the risk assessment, for example to validate or correct existing risk assessment results. The purpose of the latter is to use risk assessment to guide the test process, for example using risk assessment to identify and prioritize test cases.
10 Conclusion

In this deliverable we have reported on some of the main results of NESSoS WP10 achieved during the third year of the project. The results build on the work conducted previously in this work package, and contributes to support a risk and cost aware software development life cycle (RC-SDLC).

As explained in the introduction, the various contributions in this deliverable can be categorized into four topics: i) Chapter 2 and Chapter 3 concern methods for information security risk and cost assessment of ICT and cloud service systems; ii) Chapter 4 concerns methods and techniques for security risk management of evolving systems; iii) Chapter 5 and Chapter 6 concern methods for security risk assessment leveraging on security assurance; iv) Chapter 7 and Chapter 8 tackle security risk assessment challenges from a more formal perspective.

A strategy of WP10 is to build a portfolio of methods, techniques and tools to support and facilitate the process for maintaining an RC-SDLC that we presented in deliverables D10.2 and D10.3. The portfolio consists of complementary artifacts that support different phases, activities or tasks in the overall process, and the artifacts can leverage each other and be orchestrated in different ways.

As described in the DoW, there are three WP10 tasks for which research activities were planned for the third year of NESSoS:

- Task 10.2: Methodology for risk and cost aware SDLC
- Task 10.3: Run-time risk management
- Task 10.4: Integration of the risk and cost aware methodology

Most of the work presented in this deliverable contributes in different ways to Task 10.2, although particularly the methods in Chapter 2 and Chapter 3 as they are more general. Task 10.3 is mostly addressed in Chapter 6 with the use of testing and monitoring for facilitating the security risk assessment, and in Chapter 7 with risk-based policies for usage control. Task 10.4 involves the integration of the methodology for managing risk and cost awareness within the full SDLC, leveraging on the assurance techniques and security metrics of WP9. Chapter 5 and Chapter 6 contribute to this task with techniques and tools for risk-based security testing and test-based security risk assessment. For the test-based security risk assessment we have also ongoing work on using security metrics for capturing relevant test data, both from active security testing and passive security testing (monitoring).
A NESSoS WP10 Third Year Publications


References


[58] E. J. Schwartz, T. Avgerinos, and D. Brumley. All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask). In Security and Privacy (SP), 2010 IEEE Symposium on, pages 317–331, 2010.


