Network of Excellence

Deliverable D10.2

Initial Methods for Risk and Cost Aware SDLC
Abstract

This deliverable presents the initial WP10 methods for a risk and cost aware software development life cycle (SDLC). The methods are accommodated to the heterogeneous, modular and dynamic aspect of the Future Internet, with support for assessing security risks and costs in a modular approach and in which risks and costs evolve. The targeted methods enrich the whole SDLC and have the potential to govern the development process where risk and cost are prominent drivers. While security needs to be ensured from many perspectives, the methods aim to facilitate the demonstration of the value of security solutions and their return on investment from a business oriented perspective.

An overall process for a risk and cost aware SDLC is presented, and is our first attempt to integrate security risk management and security cost management into the phases of an iterative SDLC. Several specific methods and techniques are introduced for providing support for specific phases or tasks in this overall process. This includes a method for evaluating and selecting identified treatment options for security risks that is based on techniques for balancing risk and cost in the SDLC, a method for the continuous assessment of risks in run-time, and methodological support for assessing risk and cost in the development of web-applications and web-services. The deliverable moreover proposes a formal foundation for risk and cost management that serves as a basis for rigorous analysis and reasoning about risk and cost.
Keyword List

Future Internet, security, risk, cost, software development life cycle, return on investment in security
## 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>K. Stølen (SINTEF), B. Solhaug (SINTEF), M. S. Lund (SINTEF)</td>
</tr>
<tr>
<td>0.2</td>
<td>Draft introduction</td>
<td>M. S. Lund (SINTEF)</td>
</tr>
<tr>
<td>0.3</td>
<td>Integration of initial input from partners</td>
<td>M. S. Lund (SINTEF), B. Solhaug (SINTEF), J. Cuellar (SIEMENS), K. Radatz (UDE) (UDE), I. V. Torres (ATOS), A. Yautsiukhin (CNR)</td>
</tr>
<tr>
<td>0.4</td>
<td>Integration of input from partners</td>
<td>M. S. Lund (SINTEF), B. Solhaug (SINTEF), K. Radatz (UDE) (UDE), A. Yautsiukhin (CNR)</td>
</tr>
<tr>
<td>0.5</td>
<td>Added first draft of appendix C</td>
<td>A. Yautsiukhin (CNR), L. Krautsevich (CNR), A. Lazouski (CNR), F. Martinelli (CNR)</td>
</tr>
<tr>
<td>0.6</td>
<td>Finalised appendix on RC-SDLC; full draft of chapter on RC-SDLC</td>
<td>M. S. Lund (SINTEF), B. Solhaug (SINTEF)</td>
</tr>
<tr>
<td>0.7</td>
<td>First full draft of chapter and appendix on method for balancing risk and cost</td>
<td>S. Eicker (UDE), A. Nagel (UDE), K. Radatz (UDE)</td>
</tr>
<tr>
<td>0.8</td>
<td>Editing, polishing and amending chapter and appendix on method for balancing risk and cost</td>
<td>B. Solhaug (SINTEF)</td>
</tr>
<tr>
<td>0.9</td>
<td>First full draft of chapter and appendix on method for risk and cost of web applications</td>
<td>J. Cuellar (SIEMENS), B. Grobauer (SIEMENS), B. Meyer (SIEMENS), M. Ochoa-Ronderos (SIEMENS), J. Stijohann (SIEMENS)</td>
</tr>
<tr>
<td>1.0</td>
<td>Chapter on likes with other WPs; conclusions; polishing</td>
<td>B. Solhaug (SINTEF), M. S. Lund (SINTEF)</td>
</tr>
<tr>
<td>1.1</td>
<td>Finalization for review</td>
<td>B. Solhaug (SINTEF), M. S. Lund (SINTEF), A. Yautsiukhin (CNR), J. Cuellar (SIEMENS)</td>
</tr>
<tr>
<td>1.2</td>
<td>Updates according to review</td>
<td>B. Solhaug (SINTEF), A. Yautsiukhin (CNR), J. Stijohann (SIEMENS), M. Gonzalez (ATOS), A. Pasic (ATOS), K. Radatz (UDE), M. S. Lund (SINTEF)</td>
</tr>
</tbody>
</table>
## Document Review

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Reviewer</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-10-06</td>
<td>1.1</td>
<td>Gabriele Costa (CNR) and Widura Schwittek (UDE)</td>
<td>Several minor corrections and suggestions were given by the reviewers that have been taken into account in the finalization. A main concern of both reviewers was that parts of Chapter 8 was too technical. This has been mitigated by adding concrete and illustrative examples.</td>
</tr>
</tbody>
</table>

## Report Versioning

<table>
<thead>
<tr>
<th>Reports Versioning</th>
<th>Outline (0.1)</th>
<th>Draft (1.0)</th>
<th>Under internal review (1.1)</th>
<th>Accepted (1.2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>2011-05-10</td>
<td>2011-08-26</td>
<td>2011-09-20</td>
<td>2011-10-21</td>
</tr>
</tbody>
</table>
Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE OF CONTENTS</td>
<td>7</td>
</tr>
<tr>
<td>ACRONYMS</td>
<td>11</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>13</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>15</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>17</td>
</tr>
<tr>
<td>2 THE INDUSTRIAL NEED FOR RISK AND COST AWARE SDLC</td>
<td>19</td>
</tr>
<tr>
<td>2.1 The Industrial Need for Risk and Cost Management</td>
<td>19</td>
</tr>
<tr>
<td>2.2 The Need for a Risk and Cost Aware SDLC</td>
<td>20</td>
</tr>
<tr>
<td>2.3 Risk and Cost Aware SDLC for Services</td>
<td>21</td>
</tr>
<tr>
<td>2.4 Conclusion</td>
<td>21</td>
</tr>
<tr>
<td>3 MAIN CONCLUSIONS FROM SOT A FOR RISK AND COST AWARE SDLC</td>
<td>23</td>
</tr>
<tr>
<td>3.1 Risk Methodology Integration</td>
<td>23</td>
</tr>
<tr>
<td>3.2 Risk versus Cost</td>
<td>23</td>
</tr>
<tr>
<td>3.3 Risk from a Run-time Perspective</td>
<td>24</td>
</tr>
<tr>
<td>3.4 Risk Assessment for Web applications</td>
<td>24</td>
</tr>
<tr>
<td>3.5 Risk from a Formal Methods Perspective</td>
<td>25</td>
</tr>
<tr>
<td>4 OVERALL PROCESS FOR RISK AND COST AWARE SDLC</td>
<td>27</td>
</tr>
<tr>
<td>4.1 Risk Management</td>
<td>27</td>
</tr>
<tr>
<td>4.2 Information Security Management</td>
<td>27</td>
</tr>
<tr>
<td>4.3 Iterative System Development</td>
<td>29</td>
</tr>
<tr>
<td>4.4 Overall Process of the Risk and Cost Aware SDLC</td>
<td>29</td>
</tr>
<tr>
<td>4.5 Instantiation of the NESSoS Methods in the Overall Process</td>
<td>31</td>
</tr>
<tr>
<td>5 METHOD FOR BALANCING RISK AND COSTS IN THE SDLC</td>
<td>33</td>
</tr>
<tr>
<td>5.1 Total Lists of Types of Costs and Benefits</td>
<td>33</td>
</tr>
<tr>
<td>5.1.1 Basic Principles</td>
<td>33</td>
</tr>
<tr>
<td>5.1.2 Total List of Types of Costs</td>
<td>34</td>
</tr>
<tr>
<td>5.1.3 Total List of Types of Benefits</td>
<td>34</td>
</tr>
<tr>
<td>5.2 Method Description</td>
<td>34</td>
</tr>
<tr>
<td>5.2.1 Step One – Customising</td>
<td>35</td>
</tr>
<tr>
<td>5.2.2 Step Two – Estimating</td>
<td>35</td>
</tr>
<tr>
<td>5.2.3 Step Three – Verifying</td>
<td>36</td>
</tr>
<tr>
<td>5.3 Conclusion</td>
<td>36</td>
</tr>
<tr>
<td>6 METHOD FOR CONTINUOUS ASSESSMENT OF RISK AND COSTS IN THE SDLC</td>
<td>37</td>
</tr>
<tr>
<td>6.1 Risk-aware Decision Making in Usage Control under Uncertainties</td>
<td>37</td>
</tr>
<tr>
<td>6.2 Intentional and Unintentional Uncertainties</td>
<td>37</td>
</tr>
<tr>
<td>6.3 Cost-effective Policy Enforcement Model</td>
<td>38</td>
</tr>
<tr>
<td>6.4 Conclusion</td>
<td>39</td>
</tr>
</tbody>
</table>
# C Risk Aware Decision Making in Usage Control under Uncertainties

- **C.1** Introduction ............................................................... 89
- **C.2** Intentional and Unintentional Uncertainties ......................... 90
- **C.3** Cost-effective Policy Enforcement Model ................................ 91
- **C.4** Risk of Violation of Policy of Several Attributes ..................... 93
- **C.5** Experiment ................................................................. 93
- **C.6** Conclusion ................................................................. 94

# D Method for Risk and Costs Assessment with Particular Focus on Web Applications

- **D.1** Motivation ................................................................. 97
- **D.2** Main Concepts: Vulnerability, Loss Event, Attack, Step, and Capability ................................................................. 98
- **D.3** Abstraction Levels, Vulnerability Assessments and Threat Modelling ................................................................. 99
  - **D.3.1** Vulnerability Assessment ............................................. 100
  - **D.3.2** Capability-based Threat Modelling ................................... 102
- **D.4** Preliminary Results ........................................................ 103
  - **D.4.1** A Suitable Risk Ontology ............................................. 103
  - **D.4.2** A First Language Proposal for Attack Transitions ................ 105
  - **D.4.3** Attack Steps ............................................................. 105
  - **D.4.4** Further Properties ..................................................... 106
  - **D.4.5** Verification Techniques and Tools ................................... 107
- **D.5** Conclusion ................................................................. 107

# E Formal Foundations for Risk Management

- **E.1** Introduction ............................................................... 109
- **E.2** Risk Graphs ............................................................... 109
  - **E.2.1** The Syntax of Risk Graphs .......................................... 110
  - **E.2.2** The Semantics of Risk Graphs ...................................... 110
- **E.3** Reasoning about Likelihoods in Risk Graphs ............................. 111
- **E.4** Reasoning about Dependencies in Risk Graphs ......................... 113
  - **E.4.1** The Syntax of Dependent Risk Graphs .............................. 113
  - **E.4.2** The Semantics of Dependent Risk Graphs .......................... 114
  - **E.4.3** Rules for Dependency Reasoning ................................... 114
- **E.5** Risk Graphs for Changing Risks ....................................... 115
  - **E.5.1** The Syntax of Risk Graphs with Change ............................ 115
  - **E.5.2** The Semantics of Risk Graphs with Change ....................... 116
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.6</td>
<td>Relating Risk Model to Target Description</td>
<td>117</td>
</tr>
<tr>
<td>E.7</td>
<td>Instantiation of CORAS</td>
<td>118</td>
</tr>
<tr>
<td>E.7.1</td>
<td>Dependent CORAS</td>
<td>120</td>
</tr>
<tr>
<td>E.7.2</td>
<td>CORAS with Change</td>
<td>122</td>
</tr>
<tr>
<td>E.8</td>
<td>Conclusions</td>
<td>122</td>
</tr>
</tbody>
</table>
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSI</td>
<td>Bundesamt für Sicherheit in der Informationstechnik</td>
</tr>
<tr>
<td>CTMC</td>
<td>Continuous-Time Markov Chain</td>
</tr>
<tr>
<td>DTMC</td>
<td>Discrete-Time Markov Chain</td>
</tr>
<tr>
<td>ERM</td>
<td>Enterprise Risk Management</td>
</tr>
<tr>
<td>ETA</td>
<td>Event Tree Analysis</td>
</tr>
<tr>
<td>FI</td>
<td>Future Internet</td>
</tr>
<tr>
<td>FTA</td>
<td>Fault Tree Analysis</td>
</tr>
<tr>
<td>GRC</td>
<td>Governance, Risk management, and Compliance</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electrotechnical Commission</td>
</tr>
<tr>
<td>ISMS</td>
<td>Information Security Management System</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>ITIL</td>
<td>Information Technology Infrastructure Library</td>
</tr>
<tr>
<td>OGC</td>
<td>Office of Government Commerce</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>RC-SDLC</td>
<td>Risk and Cost aware Software Development Life Cycle</td>
</tr>
<tr>
<td>ROI</td>
<td>Return on Investment</td>
</tr>
<tr>
<td>ROISI</td>
<td>Return on Information Security Investment</td>
</tr>
<tr>
<td>ROSI</td>
<td>Return on Security Investment</td>
</tr>
<tr>
<td>RUP</td>
<td>Rational Unified Process</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>SDLC</td>
<td>Software Development Life Cycle</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>SotA</td>
<td>State of the Art</td>
</tr>
<tr>
<td>UCON</td>
<td>Usage Control</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
</tbody>
</table>
List of Figures

Figure 2.1: Business expectations ................................................................. 20
Figure 4.1: Risk management process .......................................................... 28
Figure 4.2: ISMS process ............................................................................. 28
Figure 4.3: Rational Unified Process ............................................................. 29
Figure 4.4: Generic risk and cost aware SDLC ........................................... 30
Figure 6.1: Markov chains of scientific institute .......................................... 39
Figure 8.1: Risk graph ................................................................................. 47
Figure 8.2: Example risk graph from eHealth domain ................................... 49
Figure A.1: Risk management process ......................................................... 62
Figure A.2: ISMS process ........................................................................... 63
Figure A.3: Spiral model ............................................................................. 65
Figure A.4: Rational Unified Process ............................................................ 65
Figure A.5: Generic risk and cost aware SDLC ........................................... 66
Figure A.6: Iterations of the RC-SDLC ........................................................ 67
Figure A.7: Risk and cost aware Rational Unified Process .......................... 68
Figure B.1: How the proposed method fits into the ISO 31000 standard ........ 74
Figure B.2: Total List of Types of Costs ....................................................... 79
Figure B.3: Types of Benefits according to BSI ........................................... 80
Figure B.4: Total List of Types of Benefits .................................................. 81
Figure B.5: Three Steps .............................................................................. 82
Figure C.1: Markov chains for a seller at an auction .................................... 92
Figure C.2: Markov chains of scientific institute ......................................... 92
Figure C.3: Accuracy of prediction ............................................................... 94
Figure C.4: The time required for making 1000 decision in case of CTMCs of different size . 94
Figure D.1: Ontology for capability-based risk analysis .................................................. 104
Figure E.1: Risk graph ................................................................. 110
Figure E.2: Interval arithmetic ......................................................... 111
Figure E.3: Risk graph for changing risks .................................................. 116
Figure E.4: Two views on risk graphs for changing risk ........................................ 117
Figure E.5: Relating target model and risk model ............................................. 118
Figure E.6: Instantiation of risk graphs in CORAS ........................................... 119
Figure E.7: Dependent CORAS diagram ................................................... 121
Figure E.8: Dependent CORAS diagram with different assumption and target 121
Figure E.9: Combining dependent diagrams into one ...................................... 122
Figure E.10: Instantiation of risk graphs with change in CORAS .......................... 123
List of Tables

Table 5.1: Example of SDLC-specific Total List of Types of Costs .................................................. 35

Table B.1: Common Criteria ............................................................................................................ 76
Table B.2: Types of Costs, according to Elsener .............................................................................. 77
Table B.3: Types of Costs, shortened illustration according to Gadatsch and Mayer ..................... 77
Table B.4: Types of Costs according to ITIL .................................................................................... 78
Table B.5: Example of SDLC-specific total List of Types of Costs .................................................. 79
Table B.6: Example of a SDLC-specific total list of types of costs for the extended user training program .................................................................................................................. 83
Table B.7: SDLC-specific total list of types of costs with quantified costs ....................................... 83
Table B.8: Verified total list of costs and benefits ............................................................................. 85

Table C.1: Decision matrix .............................................................................................................. 91

Table E.1: Guidelines for consistency checking likelihoods .............................................................. 112
Table E.2: Naming conventions ...................................................................................................... 115
Table E.3: Denoting likelihoods ...................................................................................................... 116
1 Introduction

The main objective of WP10 is to incept a risk and cost aware Software Development Life Cycle (SDLC) that enables practitioners to instantiate and exploit an engineering process that optimises value-for-money in terms of minimising effective risk, while keeping cost low and justified.

During the SDLC there is a need to ensure security from many perspectives, and obviously the value of security solutions and their return on investment have to be clearly demonstrated from a business oriented perspective. The value of the chosen security solutions has to be derived from the risk analysis. The net value of the investment is obliged to be derived by analysing the cost that comes along with creating security solutions and implementing security measures. The integration of risk and cost analysis in the whole SDLC, and an extension of the overall approach toward execution time, is the necessary response to these needs.

The identification and assessment of risks and the analysis of costs associated with countermeasures can then enable the making of the most cost effective security design decisions and the selection of implementation techniques. We accommodate the methodology to the nature of Future Internet (FI) services that are heterogeneous, modular, and dynamic and in which risks and costs evolve. In order to manage these inherent aspects of the Future Internet scenarios we develop a modular approach to risk and cost analysis. The modular approach fosters a process for risk and cost aware SDLC in which analysis and risk mitigation strategies can be dynamically updated by addressing specific parts in isolation.

In this report we thus provide a methodology for performing a risk and cost aware SDLC by leveraging on and extending state-of-the-art methods, techniques and tools. The targeted methodology enriches the whole SDLC and it has the potential to govern the development process where risk and cost are the dominant drivers.

The report is structured as follows. Chapter 2 motivates the results presented in this deliverable by reporting on the industrial needs in relation to the risk and cost aware SDLC. Chapter 3 summarises the main conclusions from our state-of-the-art investigation. In Chapter 4 we present the overall process of the risk and cost aware SDLC. On the one hand the overall process is a first attempt to integrate security risk management and security cost management into the phases of an iterative SDLC. On the other hand, the chapter explains how specific risk and cost analysis methods and techniques developed in the context of WP10 can be utilised to support specific phases or tasks in this overall process.

These specific methods and techniques are presented in the subsequent four chapters. In Chapter 5 we present a method for evaluating and selecting identified treatment options for security risks that is based on techniques for balancing risk and cost in the SDLC. Chapter 6 presents a method for continuous assessment of risks in run-time that provides decision support regarding access and usage control. Chapter 7 addresses the methodological needs in relation to assessing risk and cost in the SDLC while focusing on web applications. In Chapter 8 we propose a formal foundation for a risk and cost aware SDLC by the formalisation of risk modelling techniques.

In Chapter 9 we discuss the links with the technical work of other NESSoS WPs and outline potential areas of integration. Finally, in Chapter 10 we make the conclusions of the report.

Each of the main chapters, i.e. Chapter 4 through Chapter 8, are written as extended abstracts of the work carried out with the details elaborated in appendices. In order to shorten the way from deliverable to publication and thereby ease the dissemination work, the appendices are written in the form of papers. Appendix A provides the details of the risk and cost ware SDLC presented in Chapter 4. Appendix B provides the details of the method for balancing risk and cost in Chapter 5. Appendix C provides the details of the method for continuous risk assessment at run-time presented in Chapter 6. Appendix D provides the details of the method for risk and cost assessment focusing on web-applications and web-services presented in Chapter 7. Appendix E provides the details of the formal foundation presented in Chapter 8.
2 The Industrial Need for Risk and Cost Aware SDLC

In this chapter we provide an overview of the causes that make Risk and Cost Management a crucial activity for industry and, in particular, for IT companies. Continuous Risk and Cost assessment provides a company with certain guarantees of gains while also helps to minimize losses, to correct vulnerabilities or to take mitigation actions. An actual concern is that IT risk and IT security risk assessment in the context of the Future Internet will be performed in very different usage environment conditions to those the industry is used to. The adaptation of industry processes to this new evolving environment requires a smarter, more adaptive and agile risk management for software services at all the stages of the Software Development Life Cycle (SDLC). This is the polyhedral problem that is addressed in this deliverable from many different perspectives.

2.1 The Industrial Need for Risk and Cost Management

Companies all over the world operate in an increasingly dynamic environment. Many markets or financial prospects are uncertain, with a set of external cross-* (cross-sector, cross-layer, etc.) risks with a variable probability of occurrence causing a different impact to each business. For large organisations threats and attacks can come in a wide number of forms and from many different directions. Managing the risks involved requires sophisticated planning and operation at an acceptable financial cost, and cost-efficiency remains on the top of industry needs, as illustrated by Figure 2.1 which is snipped from a Gartner report [21]. Risk assessment and risk management (see e.g. [31]) are rather well understood practices by industry with rather coherent definitions of processes or terminology across sectors and governance layers. With regard to IT risks, and IT security risks in particular, we can notice that these risks (and operational risks in general) are increasingly managed in the same frameworks as business, financial, systematic or market-wide risks.

Enterprise Risk Management (ERM), for example, is a risk-based approach to managing an enterprise, integrating concepts of IT security risks. ERM addresses the needs of various stakeholders in an enterprise, with the aim of reducing gaps between governance layers and ensuring coherence between risk management layers. In this direction it is worth to mention the MASTER project\(^1\) that builds on the COSO Framework. This framework defines “Enterprise Risk Management-Integrated Framework” [12] as a

“...process, effected by an entity’s board of directors, management, and other personnel, applied in strategy setting and across the enterprise, designed to identify potential events that may affect the entity, and manage risk to be within its risk appetite, to provide reasonable assurance regarding the achievement of entity objectives.”

This need for tighter integration of governance, risk and compliance levels is even more evident with the shift toward service oriented computing and in particular with the introduction of clouds. The MASTER stack of GRC tools targets the whole chain of risk and cost issues covering not only service owners (enterprises), but external entities such as cloud providers, software providers, security solution providers, IT auditors and other key stakeholders. Another argumentation for industrial need related to “risk stack” is given by the Cloud Security Alliance\(^2\).

The second steady driver for risk and cost management is coming from the lack of meaningful and widely accepted metrics in software and service security. The risk management process provides meaningful, although not fine-grained and fine-tuned, metrics and indicators for impact (how much does this failure cost?), which facilitates decision makers’ evaluation and choices when it comes to implementation of security controls or investments in software and service engineering. However, cost-driven choice of security controls is often based on static risk assessment driven by fixed inputs, and is a typical top-down process which is not sufficient to assure risk and cost-awareness in environments with complex governance such as outsourcing environment or dynamic IT architectures.

\(^1\)http://www.master-fp7.eu/
\(^2\)https://cloudsecurityalliance.org/
Business expectations for IT call for greater productivity and continued cost-efficiencies

<table>
<thead>
<tr>
<th>Business expectations</th>
<th>Ranking of business priorities CIOs selected as one of their top 5 priorities in 2010, and projected for 2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reducing enterprise costs</td>
<td>2</td>
</tr>
<tr>
<td>Increasing the use of information/analytics</td>
<td>3</td>
</tr>
<tr>
<td>Improving enterprise workforce effectiveness</td>
<td>4</td>
</tr>
<tr>
<td>Attracting and retaining new customers</td>
<td>5</td>
</tr>
<tr>
<td>Managing change initiatives</td>
<td>6</td>
</tr>
<tr>
<td>Creating new products or services (innovation)</td>
<td>7</td>
</tr>
<tr>
<td>Targeting customers and markets more effectively</td>
<td>8</td>
</tr>
<tr>
<td>Consolidating business operations</td>
<td>9</td>
</tr>
<tr>
<td>Expanding current customer relationships</td>
<td>10</td>
</tr>
<tr>
<td>Supporting regulation, reporting and compliance</td>
<td>11</td>
</tr>
<tr>
<td>Creating new sources of competitive advantage</td>
<td>12</td>
</tr>
<tr>
<td>Expanding into new markets and geographies</td>
<td>13</td>
</tr>
</tbody>
</table>

* New question for that year

Figure 2.1: Business expectations

Applications that rely on service compositions with unknown behaviour, simply stated, cannot be certified as secure. Behaviour of users, operators, service providers, data transmission mechanisms, etc. is increasingly non-deterministic and these issues should also be addressed, in so far as possible, by risk assessment. While industry (especially safety critical software) has some experiences with modelling these “usage environment conditions”, we must recognise that with the Future Internet many of these usage conditions are changing and that certain approaches (e.g. model checking, rule based systems, formal methods) might not be compatible with iterative and agile risk assessment. One possible approach is to build an “usage and environment conditions database” that encompasses as many possible service-usage-environment etc. constellations, that could lend itself to query by risk assessment components.

2.2 The Need for a Risk and Cost Aware SDLC

The Software Development Life Cycle (SDLC) aims at defining, acquiring and implementing application systems. This includes all stages of activity from requirements identification through implementation of the application system. It is during the analysis phase, when it is usually performed the first risk and cost analysis, which, at least in the case of spiral SDLC, should be continuously monitored and updated in each iteration and along the whole development life cycle. In relation to this it is important to distinguish between SDLC risks (project risks) and IT security risks, the latter being the main concern with respect to the NESSoS project. Some of the most common SDLC risks (not IT security risks) considered by Software Analysts within the SDLC are the following: Use of inappropriate SDLC for the application system, inadequate controls in the SDLC process, inadequate involvement of stakeholders, inappropriate selection of technology and architecture, and poor quality management [57]. These risks might be related to IT security threats or risks such as unauthorised access, misuse of information, weak application controls, or fraud. It is therefore important to define the vector of (potential) cost and benefits (The Value Function), assign respective costs to them and by this to adequately evaluate the risk reduction with the help of realistic scenarios and probabilities. However, security economics research and use in industry best practice is rather limited either to the application domain e.g. IT in the finance sector, or it addressing only a limited threat and risk spectrum. Industry needs a more practical and hands-on approach to make
efficient, rational and transparent decisions during SDLC.

Therefore, in our opinion, risk management should be as flexible as SDLC, following an incremental, modular and component-based approach. It should moreover be based on continuous feedback from run-time risk monitoring and mitigation to cope with the highly dynamic and evolving nature of the Future Internet.

2.3 Risk and Cost Aware SDLC for Services

In a dynamic service-based environment risk analysis has to be much more agile than what is offered by traditional risk analysis methods. It must identify risks at different stages of the engineering life cycle, analyse compositional risk effects and inter-dependencies, and enable and provide reaction when policy violation occurs. A certain degree of “intelligence”, as well as agility, is needed in order to assure proper risk analysis of software services throughout their life cycle and different methods might be necessary at different stages. Identification, analysis and evaluation of security threats and vulnerabilities are the only way to understand and measure the impact of the risk involved in each stage.

Future Internet is characterised by highly dynamic Service Oriented Architectures (SOA) where outsourcing and distributed management constitute the norm rather than the exception. This will increase complexity in “high-level” assumptions and requirements, such as those coming from trust, privacy or compliance with regulations. We thus need an approach to risk management that supports the understanding of risk at each stage and iteration of software engineering, that supports iterative and rapid updates of the risk landscape as a response to changes in the environment, but is also able to cope with composition and decomposition of these “high-level” assumptions and requirements, as well as their treatment across service chains.

One of the main SOA principles is composability, a concept originated from the area of component oriented architecture. Closely related to composability are other fundamental SOA principles like reusability, autonomy, and loose coupling. Therefore, a risk and cost aware SDLC for SOA-based applications should take into account risks stemming from composability, reusability, autonomy and loose coupling. There is a need to support, during the entire design cycle, participation of different actors with greatly differing expertise, risk appetite, etc. Back and forth tracking, even during composite execution, should be possible in this type of SDLC. End-to-end dynamic risk assessment would feed iterations of SDLC with inputs from various perspectives (e.g. envisaged use of service or different contexts).

Additional iterations in SDLC for services might include risk observation and orientation in composition design (including a set of risk and cost-aware composition design patterns), supporting for service search based on “high-level” assumptions and requirements (e.g. trust or privacy) and on secure composition design patterns.

One important prerequisite, mentioned in [47], is the translation of non-trivial “high-level” requirements, often expressed in natural language, into controls and policies that can be deployed and monitored in an operational infrastructure and that can generate evidences which, at a later stage, enable end-to-end risk assessment and eventual auditing. Set of operational risk policies should be used as an interface for service infrastructure and internal control process. Evidence aggregation, correlation, analysis, control refinement and risk re-assessment are some of the tasks related to this challenge. Other major risks categories for SDLC that have been identified by industry include risk allocation and multi-party assessment, fuzzy security objectives and risks related to new type of threats (e.g. convergence of internet of things with internet of services).

2.4 Conclusion

Traditionally, risk management is seen by industry as a process aiming at an efficient balance between realising opportunities for gains while minimising vulnerabilities and losses. It is an integral part of management practice and an essential element of good corporate governance. It is generally accepted in that risk analysis, assessment and mitigation are part of risk management. Risk management is a recurrent activity that copes with the analysis, plan, implementation, control and monitoring of implemented measurements and the enforced security policy. In contrast, risk analysis is executed in certain time intervals
and – up to the next execution – provides a temporary view of the risks. For the Future Internet applications, where dynamicity, context-awareness and non-deterministic system behaviour are more rule than exception, we must be able to identify and assess risks at all stages of SDLC. Different risk assessment methods might be needed at different stages, while incremental updates of previous risk assessments should enable more flexibility and agility. Modularity and certain degree of risk reasoning through “risk agents” might be needed to analyse compositional risks and inter-dependencies and to enable (automated) reaction to “risk trigger” events. Novel methods for risk values, metrics or indicator aggregation are also needed.
3 Main Conclusions from SotA for Risk and Cost Aware SDLC

The state-of-the-art (SotA) and a gap analysis with respect to the needs regarding a risk and cost aware SDLC is reported in the internal NESSoS deliverable ID10.1 [55]. In this chapter we summarise the SotA and the main conclusions of this report.

A main goal of WP10 is to develop a methodology for performing a risk and cost aware SDLC in the setting of FI services and systems. While leveraging on and extending SotA methods, techniques and tools, the targeted methodology will enrich the whole SDLC and has the potential of governing the development process with risk and cost as dominant drivers.

The goal is ambitious and needs to be attacked from several different angles; no single approach, method or technology is likely to have the capacity to provide solutions to all the challenges of a risk and cost aware SDLC. Consequently, the SotA and gap analysis reported in ID10.2 is organised into a number of research areas, each with a potential to contribute to the overall goal of WP10. These research areas correspond closely to the main chapters of this deliverable, and the below SotA summary are structured according to the latter.

3.1 Risk Methodology Integration

In FI systems, security needs to be considered from the early phases of the SDLC. To achieve this, analysts often need to combine several methods and techniques to analyse and solve their problems. At the same time, analysts need to implement an overall risk management process that make use of such methods and techniques.

In the context of WP10, we use the ISO 31000 risk management standard [31] as the starting point. The standard is quite general and provides the necessary underlying concepts of risk assessment, as well as a risk management process, without restricting to specific risk assessment methods or techniques. In fact, several established approaches to risk management, such as OCTAVE [2], CORAS [40], CRAMM [13] and the NIST Risk Management Guide for Information Technology Systems [56], closely correspond to the overall process described by ISO 31000.

Considering the SotA methods and techniques that can be combined to support a risk and cost aware SDLC, one approach could be to integrate specific promising techniques into one holistic technique customised for our purposes. However, such a technique would be quite rigid and with little flexibility to be adapted to varying needs, environments, end-user preferences, etc. Instead, our aim is to identify and formalise interfaces between existing framework such that they can leverage on each other. Such an interface includes a conceptual integration that allows mapping of modelling elements, as well as a process level integration in terms of orchestration of techniques. The orchestration will allow separate techniques to be carried out separately, however interacting with and leveraging on other processes by invoking the well-defined interfaces.

3.2 Risk versus Cost

The main gaps for the management of risk and cost identified in ID10.1 are the need for a more precise and refined economic security measure, as well as the lack of security levels in SLAs.

Return of security investment (ROSI) [54] concerns determining what is the right amount of money and time to invest in security. Such a question is challenging in many respects, and even more so considering that “ubiquitous network connectivity, novel architectures, and business models fostering massively distributed computing (aka Cloud computing) are about to change the security landscape” [7]. In estimating the ROSI, one may build on traditional financial ratios and techniques for estimating the return of investment (ROI) [7, 54]. However, the ROSI is not always clear because “the benefits produced by security programs, assets and investments are not always apparent despite their existence”, and “measuring the results and progress of security activities is difficult; some of the best results might be when impacts to business are completely mitigated and nothing bad occurs” [43]. Moreover, how to identify and estimate
the inputs to calculate the ROSI may be challenging as “there is no universal calculation of Information Security ROI” [58].

In the gap analysis, we argue that there is a need for specific ratios for the security domain. There is also a need for techniques for aggregating and mapping security costs to different security levels in order to justify investments, and the security investment costs must be closely related to security risks and their mitigation in the SDLC. Considering SLAs, there is a need to integrate security levels into these, including a description of the costs associated with each security level.

### 3.3 Risk from a Run-time Perspective

With respect to risk from a run-time perspective, a main challenge is to adapt risk analysis to the changing environment of FI in such a way that it can be applied at run-time. Related to this is the challenge of defining run-time mitigation strategies, of which run-time reconfigurable security mechanisms based on risk is a special case of great importance.

Traditional risk assessment (e.g. [31, 56]) is intended to be performed from time to time. More mature methodologies see risk assessment as a part of a continuous and repetitious risk management process (e.g. [2]). Although now it is widely accepted that risk assessment must be performed regularly, the more and more dynamic FI requires faster re-evaluation of risks. In other words, real-time risk assessment is necessary for timely reaction on changes in the operational environment.

In our SotA and gap analysis deliverable [55] we have identified in the literature a number of preliminary steps toward real-time risk assessment. These approaches use various means for intrusion detection, sensors, indicators, signalling, etc. to support continuous estimation of security risk levels. With real-time constraints, many challenges nevertheless remain: How to collect the data, how to process them and make them suitable for the analysis, how to make the analysis efficient, fast, and less subjective, and, last but not least, how rapidly to react on the detected problems.

Risk analysis for modern systems also must take into account the structure of the considered system in order to make the analysis more efficient. Only the part which has changed must be reconsidered if new data is available. Moreover, considering the structure of the analysed system should help to identify the source of the problem. Since current systems often consist of subsystems which cannot be directly monitored (e.g., Web Services or IaaS, PaaS and SaaS in Clouds) modular approaches for real-time risk analysis are required.

### 3.4 Risk Assessment for Web applications

With respect to risk assessment of web applications we identify the gaps to be the need for embedding risk assessment in a Security Maturity Model and a specialisation of risk assessment to web-based applications. The latter will need to include systematic use of information about known threats and vulnerabilities, as well as integration of static analysis and white-box testing in the risk assessment.

Although web applications are a quite special type of IT systems, with its own vulnerabilities, threats and risks, there is no standard risk management methodology for them, or particular modules to support them. Some authors propose to change the classical SDLC in the case of web applications to accommodate the evaluation of security in an additional SDLC phase of security assessment after requirements, design, and implementation phases [35]. Other authors propose to centre the risk analysis for web-applications on identifying vulnerabilities and associating them with the activities that generate them within the system [17].

In the same way that general purpose risk assessment methodologies have been specialised to risk assessment methodologies for IT systems, those, in turn, should be specialised even further for the particular needs of web-based applications. This implies also developing new building blocks for risk assessment as part of the SDLC, in a similar way as proposed in [35], but including also methods for an early risk assessment. The new building blocks should provide, for instance, a guide to the application owner regarding the selection and use of secure coding guidelines or when to perform which tests in which depth.
3.5 Risk from a Formal Methods Perspective

With respect to risk from a formal methods perspective, the identified gap is the need for a formal semantic foundation for risk analysis with the expressiveness and generality to underpin risk analysis techniques, as well as related activities such as system specification, security requirements engineering and policy specification.

Several of the classical risk assessment techniques that exist can be said to have formal foundations. These include Fault Tree Analysis (FTA) [27], Event Tree Analysis (ETA) [29], system analysis using block diagrams [28], Markov analysis [30], and Bayesian networks [6]. While traditional risk analysis techniques like FTA and ETA do have a formal foundation in propositional logic and probability theory, this is usually only utilised for likelihood estimation and some kinds of reliability analysis. The same is generally true also for Markov analysis and the application of Bayesian networks. This means that to a lesser degree are the formalisms used to support the whole risk analysis or risk management process and to little degree are the formalisms utilised to support activities related to risk analysis and risk management, such as analysis of security requirements and policies, risk-based testing, and so fourth.

Various goal-oriented approaches, even though aligned with risk analysis, do not provide full risk analysis methodologies. In [3], an algorithm for analysis of risk in relation to delegation based on goal trees with trust relations is provided, and in [42] goal trees are used to model risk scenarios. However, as with the traditional techniques discussed above, the approaches provide only support for analysis in certain specific tasks of the risk analysis method. The risk graph approach of [9] is applied both for likelihood analysis and for reasoning about and resolving mutual dependencies between risk scenarios expressed in an assumption-guarantee style. The approach is closely related to the CORAS method [40], but does still not provide explicit support for the full risk analysis process.
4 Overall Process for Risk and Cost Aware SDLC

This chapter presents our first steps toward a Risk and Cost aware Software Development Life Cycle (RC-SDLC) for the development of secure software services and systems. Both system development and risk analysis of existing systems are well-understood; the challenge in providing an RC-SDLC is to integrate security risk management and security cost management in all the phases of the SDLC in such a way that security risk and security cost considerations are drivers of the development and not something that is applied afterwards.

When providing a SDLC like the one we are aiming at, there are two pitfalls we should seek to avoid. First, making it so general that it does not provide any real guidance as to how to put it into practice. Second, making it so specific that it is not applicable because of the assumptions and constraints placed upon potential end-users. To avoid the latter pitfall, we do not assume any specific SDLC, risk management methodology or cost model as the starting point of the RC-SDLC; we rather make the general assumption that the SDLC is iterative and that the risk management methodology conforms to the ISO 31000 risk management standard [31]. There is hence a number of existing approaches that can instantiate the RC-SDLC that we propose. In order to avoid the former pitfall, we provide guidelines and examples on how to tailor the RC-SDLC to a specific organisation with specific choices of SDLC, risk management methodology, cost models, and so forth. For this task we look to the ISO/IEC 27001 [32] standard so that our guidelines are specialisations of a suitable subset of the requirements given in this standard.

The chapter is structured as follows. In Section 4.1 and Section 4.2 we briefly introduce the ISO 31000 and ISO/IEC 27001 standards, respectively. In Section 4.3 we discuss existing iterative SDLC processes. Section 4.4 introduces the overall process of the RC-SDLC and relates this to system development methodologies. Finally, Section 4.5 discusses how the methods introduced in Chapter 5 through Chapter 8 can be instantiated in the RC-SDLC process. The reader is referred to Appendix A for several details that are omitted in this chapter.

4.1 Risk Management

Risk Management is guided by the ISO 31000 risk management standard. Figure 4.1 is adapted from this standard and illustrates the seven activities of the risk management process. The five activities in the middle constitute the core activities of a risk analysis and result in the documentation of risks, as well as the documentation of treatment options for unacceptable risks. The remaining two activities are continuous activities of the overall risk management process. The reader is referred to Appendix A for further details about the ISO 31000 process.

ISO 31000 comes with no guidance on concrete risk analysis techniques and must therefore be operationalised by a risk analysis method implementing the standard (for example [2, 13, 40]) in order to be put into practice. While a risk analysis method provides methodological advice on how to carry out the various activities of risk management, a risk analysis technique is more narrow in the sense that it addresses only some aspects of the risk analysis process. A risk analysis method typically makes use of one or more risk analysis techniques. It follows that ISO 31000 moreover provides no specific guidance on risk analysis techniques in the context of an SDLC.

4.2 Information Security Management

The ISO/IEC 27001 standard [32] defines an Information Security Management System (ISMS) as “the part of the overall management system, based on a business risk approach, to establish, implement, operate, monitor, review, maintain and improve information security”. The standard describes a Plan-Do-Check-Act process model for ISMS process, as shown in Figure 4.2, which is adapted from the standard. Plan is to establish ISMS policy, objectives, processes and procedures relevant to managing risk and improving information security to deliver results in accordance with an organisation’s overall policies and objectives. Do is to implement and operate ISMS policy, controls, processes and procedures. Check is to assess and measure process performance against ISMS policy, objectives and practical experience. Act is to take corrective and preventive actions to achieve continual improvements of the ISMS.
Figure 4.1: Risk management process

Figure 4.2: ISMS process
Related to the ISO 31000 risk management process, the five activities of context establishment, via risk assessment, to risk treatment is embedded in the Plan/Establish stage. We moreover find the activities of Monitor and Review in the Check/Monitor and Review stage, and we find the activity of Communicate and consult in the Act/Maintain and Improve stage.

### 4.3 Iterative System Development

Iterative system development can be categorised according to three classes of SDLCs, ranging from “heavy” to “light” methodologies, namely the spiral model, the unified process and agile methods. In this chapter we focus on the unified process, referring to Appendix A for descriptions of the remaining two.

There are several variants of the unified process, the best known and most used of which is probably the Rational Unified Process (RUP) [36]. RUP is structured around two dimensions as depicted in Figure 4.3. One dimension progresses along the lifespan of the system development and is divided into the four phases of inception, elaboration, construction and transition. The other dimension is activities, or disciplines, such as requirements, design, implementation and test, that run across all the phases.

RUP is highly iterative, with the possibility of several iterations within each of the phases. As can be seen from the figure, these iterations are iterations of the actual development, so each iteration includes all cross-cutting activities. The process emphasises component based development and continuous integration as a way of reducing (project) risk. This also leads to a natural partitioning of the work, since development of components can be associated with iterations.

### 4.4 Overall Process of the Risk and Cost Aware SDLC

In an RC-SDLC there will in addition to activities directly related to risk and cost be activities that are only indirectly related, such as requirement specification, design, and test, and on which risk and cost will have an impact. Detailed discussions on activities only indirectly related to risk and cost is outside the scope of this work; instead the focus will be on the risk and cost activities, as well as the placement of such activities in relation to other SDLC activities.

Figure 4.4 illustrates a generic iterative SDLC made risk and cost aware by the introduction of the
(generic) activities of *Risk analysis* and *Cost/benefit analysis*. In this illustration we assume the general activities such as *Requirements*, *Design*, etc. to comprise also security related sub-activities so that for example the activity *Requirements* includes the definition of security requirements, and so fourth. Based on this assumption we have indicated in the figure where the work of the technical NESSoS work packages fits in.

Although the various activities of the RC-SDLC are placed sequentially in this illustration, it is important to be aware that the effort in each activity varies in each iteration. Implementation and testing are, for example, not addressed in the early iterations, while requirements are addressed to little or no extent in the late iterations. This also means that the security risk and cost analyses can precede any of the other activities in the life cycle and therefore can be understood as an activity orthogonal to the other activities in the RC-SDLC.

The motivation for the placement of the activity *Risk analysis* after the activity *Requirements* is based on ISO/IEC 27001 which has information security policies and expectations as a prerequisite for the ISMS process, while risk analysis is part of the *Establish ISMS* stage of the process. As it can be argued that security policies is a prerequisite to risk analysis it is also possible to argue that security policies can be an outcome of risk analysis (as a risk treatment option). We will however assume that there exist initial security requirements prior to risk analysis and that the security requirement may be revised either as part of the risk treatment step of risk analysis or in subsequent iterations of the SDLC.

*Cost/benefit analysis* follows *Risk analysis* as the outcome of risk analysis is important input to this activity; as is stated in Chapter 5, the cost/benefit analysis method provided in this deliverable “can be considered as an extension to the treatment evaluation phase of existing risk management frameworks”. Further, *Cost/benefit analysis* precedes *Design*, *Architecture* and the other development activities as the cost/benefit analysis in many case will influence decisions made in these subsequent activities.

The risk and cost aware SDLC is iterative, but it should be noted that the iterations will not be identical even though each iteration has the same generic activities. For example, *Risk analysis* will typically comprise of high-level risk analysis in the initial iteration, detailed risk analysis in subsequent iterations and maintenance and/or run-time risk analysis in the final iterations. Other activities may have different workload for the different iterations. For example the activity *Implementation* may have little or no workload in the initial iteration but heavy workload in the later iterations.

Related to the RUP (and other Unified Processes) the iterations of the generic RC-SDLC correspond to the iterations of the RUP, i.e. Initial, Elab #1, Elab #2, Const #1, Const #2, etc. We should then see Risk & Cost Analysis as one of the cross-cutting activities together with Requirements, Analysis & Design, and so fourth. This is illustrated in Figure A.7 on page 68 which is modified from Figure 4.3. As this enhancement of the RUP is currently theoretical, the graph showing the allocation of effort to the Risk & Cost Analysis activity is a rough estimate of what we expect in a “risk and cost aware RUP”.

**Figure 4.4: Generic risk and cost aware SDLC**

NESSoS - 256980
4.5 Instantiation of the NESSoS Methods in the Overall Process

On the one hand, the aim of WP10 is to incept a risk and cost aware SDLC by adapting and specialising existing iterative SDLCs. On the other hand, WP10 aims at developing and providing methods and techniques to support specific phases and tasks of the overall process of such a risk and cost aware SDLC. In Chapter 5 through Chapter 8 below several such methods are presented. In the following we briefly place these in the context of this chapter and explain how each of them can be instantiated in various phases of the overall RC-SDLC. A more detailed discussion is given in Appendix A.

- The method for balancing risk and cost in the SDLC presented in Chapter 5 addresses the needs for a systematic approach to estimating and assessing the cost of identified risks as balanced against the costs and benefits of implementing identified options for risk treatment. In particular, the method targets the final phase of risk treatment in the risk management process of the ISO 31000 standard. This means that the proposed method can be instantiated during the final phases of an iteration of the risk and cost assessment process, within the wider iteration of the overall RC-SDLC.
  
  Given the documentation of the identified risks and the treatment options, the method for balancing risk and cost uses techniques for identifying, categorising and estimating costs and benefits associated with the risks and treatments within a specified time frame. The identified treatments are assessed so as to select the most advantageous options for the further SDLC process.

- The method for continuous assessment of risk and cost presented in Chapter 6 focuses on the continuous assessment of risk at run-time. The method is an approach to support access and usage decisions when there may be uncertainties regarding satisfaction or violation of the relevant policies.
  
  In the context of the ISO 31000 risk management process, the proposed method fits mostly into the risk monitoring of the continuous activity of Monitor and review. However, the method also utilises risk assessment techniques for risk identification, risk estimation and risk evaluation in the sense that the reference monitor makes continuous probability estimates that serve as the basis for risk and cost relevant decision making.

- The approach to the management of risk and cost in the development life cycle of web applications and web services that is presented in Chapter 7 focuses in particular on extracting potential attacks based on identified vulnerabilities as well as the capabilities of attackers.
  
  Related to the ISO 31000 standard, the method addresses mostly risk identification and risk estimation and aims at yielding formal risk models that document all sequences of actions that can lead to a given attack. It supplements more traditional risk management frameworks by providing specialised support for mitigating security risks related to web applications and services. A core feature of the envisage approach is the automatic generation of risk models during risk identification that can help designers of security systems to accelerate recurring tasks and to prevent errors and omissions.

- The formal foundation for a risk and cost aware SDLC presented in Chapter 8 aims at providing a rigorous basis for analysing and reasoning about risk in the risk management process. The formalisation of risk graphs and the calculi to support the reasoning about them provide techniques to support the ISO 31000 phases of risk identification, risk estimation and risk evaluation.
  
  In the wider context of a risk and cost aware SDLC there will be several iterations of the security risk assessment process. While the development life cycle evolves there may be needs to assess how the risk picture will change in the face of, for example, competing design decisions. In such a setting, the support for modelling and reasoning about changing and evolving risks can be utilised in assessing the risks associated with each alternative.

  Modular risk analysis may also serve as a useful technique in the setting of Future Internet services and systems as these are inherently heterogeneous and modular. The support for modelling and reasoning about dependencies in risk models presented in Chapter 8 may serve as useful techniques in this setting.
5 Method for Balancing Risk and Costs in the SDLC

Existing risk management frameworks (e.g. OCTAVE [2], CRAMM [13], the NIST Risk Management Guide for Information Technology Systems [56], CORAS [40]) provide methods to assess risks and identify treatments for unacceptable risks. However, the frameworks lack elaborate methods, techniques and tools for assessing and weighting treatment costs and the resulting benefits. ISO 31000 [31] mentions “balancing the costs and efforts of implementation against the benefits derived”, but gives no advice how this could be achieved. This lack of guidance motivates the development of a method for balancing risk and cost. Since the assessment of costs and benefits in this context is a complex endeavour, it cannot satisfy the needs of decision makers by providing the results of a cost/benefit analysis only. Instead, we argue to use a systematic approach, which explicitly incorporates the experience of experts in an iterative way.

In this chapter we propose the initial steps toward such a method, which can be considered as an extension of the risk treatment phase of existing risk management frameworks. (See Figure 4.1 on page 28.) In particular, the method has been designed to be embedded into the ISO 31000 standard on risk management [31]. Embedding the method in this standard ensures generality and allows the method to be integrated into the risk treatment phase of any risk management framework that operationalises the ISO standard.

The method assumes results from previous risk analysis steps such as risk models, risk estimations and identified options for risk treatment. The results of the method provide an improved foundation for further decisions in comparison to the results of a cost/benefit analysis. It has been designed with cost-efficiency in mind and is divided into three different steps.

This chapter is structured as follows. Section 5.1 introduces our concept of a “total list of types of costs” and a “total list of types of benefits” which is the basis for our method. The method itself is described in Section 5.2. The chapter finishes with Section 5.3 which gives a short conclusion and an outlook for further work. The reader is referred to Appendix B for further details and for examples illustrating the use of the method.

5.1 Total Lists of Types of Costs and Benefits

The total list of types of costs and the total list of types of benefits are the foundation of our method. They contain domain knowledge regarding costs and benefits specific for the development, introduction and use of software and will be used as guidance to build the project specific customized lists. With this concept of total lists we aim at putting all elements of possible types of costs and benefits into one list in order to derive concrete company, project and treatment specific lists from this. The total lists are derived from fields such as IT service management and business studies taking into account theory as well as practice. Hence, for the total list of types of costs, for example, several cost plans from literature were used; the total list of types of benefits was derived from risk lists. In the following subsections the development of the lists will be described in more detail.

5.1.1 Basic Principles

In order to get a consistent and reliable total list of types of costs, basic principles on how to build this list must be defined first. From the literature, four principles can be identified [23]: Cleanliness and clearness, consistency and no overlapping, completeness, and cost effectiveness. The first two principles are both aimed at providing a clean list of costs. As they sometimes influence each other in a negative way [23], Rüth tolerates a mixed type of costs in the case that its value does not matter [51]. Zilahi-Szabo’s demand for a minimum value for a separate cost category is also introduced because of cost effectiveness [61]. Cost effectiveness can therefore be seen as an encompassing principle. Hence, Rüth does not see cost effectiveness as a separate principle itself.
5.1.2 Total List of Types of Costs

In this sub-section the total list of types of costs will be derived from combining the business economics view, IT cost lists and the IT management view.

**Business Economics View**

Elsässer states that all IT costs have to be measured, classified and allocated regarding business economics principles (basic principles) [18]. Hence, this perspective is also relevant for IT costs. Some of the identified criteria are, for example, type of production factor, type of goods, type of cost, accountability, origin of cost etc., which can then be used to build and structure the total list of types of costs. Haberstock concludes that a classification is usually done by the type of the production factor. On a lower level another criterion can be used to structure the list [23]. Following Hummel and Männel a practicable classification for types of costs uses different criteria. So the structure of the resulting total list of types of costs is a combination of all criteria [26].

**IT Cost Lists and IT Management View**

Various cost lists can be found in literature, such as IT cost lists from Gadatsch and Mayer [20] or Elsener [19]. The total list of types of costs proposed in this method is among others, such as cost plan according to Jaspersen [34], built upon these cost lists. Besides these classifications, ITIL (published by the Office of Government Commerce (OGC)) offers a classification of costs which is part of the Financial Management for IT Services. [45] ITIL uses three criteria for grouping costs. Two of them have already been mentioned, namely “accountability” and “type of cost”. These are completed by “operating costs” and “investment costs”.

**SDLC-specific View**

In order to get a complete overview of the costs of a software project Henrich proposes to use the categories from IT cost lists in combination with phases from the software development process. At last Henrich states that it should be differentiated between one-time costs and running costs [25]. For the SDLC-specific view in Table 5.1 development phases according to the Rational Unified Process (RUP) [36] are used. The reader is referred to Figure 4.3 on page 29 for an overview of the SDLC phases of RUP.

5.1.3 Total List of Types of Benefits

The total list of types of benefits has been derived from risk management processes and from literature dealing with IT security and IT project management. The German federal office for information security (Bundesamt für Sicherheit in der Informationstechnik (BSI)) identifies two main types of benefits of risk treatments: qualitative benefits and quantitative benefits. The BSI states that increasing availability, integrity and confidentiality will primarily reduce the costs of incidents with a high probability [60].

Further on, there are some benefits which will also result from treating risks like image improvement, improvement of cost effectiveness, minimisation of unwanted incidents, minimisation of loss of knowledge and improvement of system stability [60]. IT project management literature also identifies several types of benefits from treating risks. Aichele [1] identifies three categories of benefits: direct benefit, indirect benefit and elusory benefits. Direct benefit is defined as saving of current costs. Indirect benefit is defined as saving of upcoming costs. Elusory benefit is a qualitative factor like e.g. enhanced data security. The total list of types of benefits shown in Figure B.4 in Appendix B has been derived from these different views.

5.2 Method Description

An overview of the steps of the method is given below, each of which will be described in more detail in the following sub-sections. The reader is referred to Appendix B for a running example of the use of the method.
### Table 5.1: Example of SDLC-specific Total List of Types of Costs

<table>
<thead>
<tr>
<th>Types of Cost</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Personnel Costs</strong></td>
<td>One-Time</td>
<td>Personnel costs for development team</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>Personnel costs for maintenance</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Material Costs</strong></td>
<td>One-Time</td>
<td>Purchase development hardware</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>Maintenance costs for hardware</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>External Services</strong></td>
<td>One-Time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overhead Costs</strong></td>
<td>One-Time</td>
<td>Certificate costs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Step 1: Customising

- **Step 1a:** Determine relevant timeframe
- **Step 1b:** Customise total cost list
- **Step 1c:** Customise total list of benefits

Step 2: Estimating

Step 3: Verifying

5.2.1 Step One – Customising

This step is organised as a workshop with experts from the department of the enterprise and/or external consultants. As a prerequisite it has to be explicitly determined by the experts of the enterprise for which timeframe the costs and benefits have to be evaluated. After that the relevant costs and benefits for the enterprise will be identified. Hence, the goal of this workshop is to prepare a customised cost and treatment list in regard to the timeframe given by the risk horizon.

The required input for phase one is the total cost list (e.g. hardware, software, human resources, etc.) and a list of possible types of benefits (e.g. image, finance, etc.) given by the proposed method and in addition the knowledge of the company's experts. The total list of types of benefits as well as the total list of types of costs is customisable. The enterprise value, for instance, might be of interest for a stock corporation but not for a private limited company.

5.2.2 Step Two – Estimating

Step two is also organised as a workshop with representatives from the client. It requires the customised lists developed during phase one as well as risk treatments, risk assessments and the risks themselves -
all given by the used risk analysis method (e.g. CORAS). During the workshop the costs of the risks will be estimated on the one hand and on the other hand the treatment costs for the risks will be appraised.

The estimation process is designed as an iterative process which will be repeated until the experts (e.g. project managers) come to a common conclusion. For structuring the process any useful method such as a Delphi study [39] or planning poker known from Scrum can be used [11].

The output of this step is the quantified benefit as a delta between costs for untreated and treated risks and the quantified costs of the treatments where the costs are estimated in regard to the given timeframe from step one.

5.2.3 Step Three – Verifying

As well as the first two phases phase three is also organised as a workshop. The needed input is provided by step two. In this phase the estimations made in step two will be verified by other experts of the company. So the output of the method will be a verified cost and benefit analysis which can then be used for further decision support methods like for instance ROI, ROSI, ROISI, etc. to evaluate the options with respect to return on investment in security.

5.3 Conclusion

In this chapter we have elaborated a new method of how to measure costs and benefits of risk treatments. The new proposed method can be embedded into any ISO 31000 based risk management methods and extends the last phase treat risks. Furthermore the method does not limit the user to a given decision model like ROI or ROISI, but gives the opportunity to freely choose a decision support method of preference.

Future work has to be done on the evaluation of the total list of types of cost and the total list of types of benefit. It has to be determined whether both lists are really a complete repository for types of costs and types of benefits or if they have to be expanded. A tool supporting the workflow of the different steps should also be made available. This tool could e.g. take into account dependencies between different treatments. For our example a plausibility check which prevents the users from entering false data could make the second day of the training program obsolete.
6 Method for Continuous Assessment of Risk and Costs in the SDLC

Our current work mostly relates to the continuous assessment of risk in run-time. Our goal is to consider computation of risk values in run-time when parameters required for the computation may change. The continuous assessment of risk in other parts of the SDLC is to be considered in the later phases of the project. The reader is referred to Appendix C for details and examples that have been omitted in this chapter.

6.1 Risk-aware Decision Making in Usage Control under Uncertainties

Access control is based on the assumption that the required attributes do not change during the access. This assumption is valid if usage sessions are short with respect to the frequency of changes of attributes. New technologies (such as Web Services, Grid, and Cloud computing) made usage sessions longer. Now, between sending a request and getting a result a considerable amount of time is required for data processing. With longer sessions we cannot rely on the assumption that attributes do not change during the session. This problem has been solved with a new paradigm – usage control model (UCON) introduced by Sandhu and Park [52, 38, 46]. The most important features of the UCON model are mutability of attributes and continuity of control. Mutability of attributes means that attributes required for making an access decision can change during a usage session. Therefore, it is not enough to evaluate attributes once before the access is granted, but continuous evaluation is required when the usage session is active.

Unfortunately, continuous usage control is not always feasible, because of a number of uncertainties which exist in the system generated by natural causes [8]. These uncertainties cause a difference between a real attribute value and a value used for making the decision. First, there are inevitable delays of information delivery in any distributed system. Second, some attributes cannot be checked frequently because it is either impossible, or impractical, or too costly. Thus, there is a trade-off between the cost of an attribute check and a possible cost if attribute is not checked. Third, there are the attributes which require considerable time to be processed before they can be sent to the decision making engine. Hence, frequently the decision about a usage session must be made when precise values of attributes are not available.

We exploit risk for making access and usage decisions when some uncertainties do exist. The idea is to assess possible losses and benefits of taking a decision. In most cases risk is used to judge if a system is secure enough or some additional controls should be installed. Uncertainties in these cases are expressed as average probability that a threat occurs. The well-known formula is used for computation of risk [22, 56, 2] when the probability of an unwanted event is multiplied by the impact of the unwanted event. The event is that the attribute value changed and an access should be revoked according to that value.

6.2 Intentional and Unintentional Uncertainties

Freshness of the attributes is unintentional uncertainty occurring due to attributes mutability. We launched three types of freshness uncertainties.

Freshness I. corresponds to the scenarios where only a part of attribute changes can be detected. For example, sensor sends information about location of a person only once per hour, because it has limited resources (power, bandwidth, memory). Thus, there is a possibility of the policy violation in-between checks despite that all values satisfy the policy.

Freshness II. implies that an attribute may change during inevitable time delays needed for the delivery (due to a network latency) and decision making (evaluation of logical predicates).
Freshness III. corresponds to scenarios where the current attribute value is uncertain since some update queries are pending and may not be committed by the time of the policy re-evaluation. For example, the reputation of a seller is formed of the deals evaluated by other buyers, when the unrated deals are not taken into account.

Correctness of Attributes. is affected by additive noises that usually exist in case of non-accurate measurements. For example, the location attribute can be sensed only with the given precision.

Trustworthiness of Attributes. appears as a result of altering attributes by the attribute provider or as the result of attacks occurred during attributes delivery, storing, etc.

6.3 Cost-effective Policy Enforcement Model

The reference monitor chooses between two alternatives (grant access and deny/revoke access) only one, which is as good as possible. Ideally, the reference monitor grants access to legitimate users when the policy holds, and forbids the access to unauthorised entities if the policy does not hold. In the presence of uncertain attributes, the reference monitor is unable to infer accurately whether the policy holds, and, consequently, to choose a good alternative. Thus, there are four scenarios how the reference monitor acts processing uncertain attributes:

- **true positive**: continue access and the policy is satisfied;
- **false negative**: continue access and the policy is failed;
- **false positive**: revoke access and the policy is satisfied;
- **true negative**: revoke access and the policy is failed.

**True positive** and **true negative** are correctly selected alternatives, while **false negative** and **false positive** are erroneous. Each scenario has a monetary outcome, cost, the reference monitor loses/gains choosing this scenario: $C^{fn}$ - for false negative scenario; $C^{tp}$ - for true positive one; $C^{fp}$ - for false positive one; $C^{tn}$ - for true negative one.

Having these costs determined we are able to apply the Decision Theory in order to make a reasonable decision: to allow access or to deny it.

The last thing we need to perform in the analysis is the probability that the policy is satisfied. Let $H_t$ be an event specifying that a real attribute value ($a^{real}$, $t$) does satisfy a security policy at time $t$, while $\overline{H_t}$ specifies the opposite. The reference monitor operates only with observed attributes and can compute $H_t^{obs}$, i.e. the policy holds for observed attributes at a given time $t$. Usually, the reference monitor possesses uncertain knowledge about real attribute values. Assume the reference monitor can measure this uncertainty by computing the conditional probability $Pr[H_t | H_t^{obs}]$ that the policy really holds at $t$ knowing that observed attributes satisfy the policy at $t$. Let $H_{\lceil t, t_{now} \rceil}$ specify that the sequence of real attribute values satisfies the policy starting from $t$ until $t_{now}$.

In usage control the policy is evaluated every time an attribute changes. In case the time between the value measuring $t$ and decision-making $t_{now}$ is large enough some values might be missed. Thus,

$$Pr_{RM} = Pr[H_{\lceil t_{perm}, t_{now} \rceil} | H_t^{obs}] = Pr_{cor}[H_t | H_t^{obs}] \cdot Pr_{tr}[H_t | H_t^{obs}] \cdot Pr_{fr}[H_t | H_t^{obs}]$$

specifies the probability that the policy really holds by $t_{now}$ knowing that observed attributes satisfy the policy at time of issuing $t$. The overall probability $Pr_{RM}$ is a combination of the probability of fresh data ($Pr_{fr}[H_t | H_t^{obs}]$), probability of correct data ($Pr_{cor}[H_t | H_t^{obs}]$), and the probability of trusted data ($Pr_{cor}[H_t | H_t^{obs}]$).

The probability that the received data is correct could be found from the precision of the measuring method and tools. There are also a number of work on determining the trustworthiness of a data provider (see the paper on reputation, e.g., experience-based computational trust in [37]). Thus, we assume that this value also could be found. Therefore, we pay more attention to finding the probability of the value to be fresh, i.e., the probability that no changes occurred since the last check we were able to observe.

\[1\]Here we assume that the three compounds are independent.
We propose to use Markov Chains in order to find this probability. The available information about a change of the attribute allows using one of the following two models:

1. We model an attribute using a *discrete-time Markov chain* (DTMC) if we know the number of attribute changes occurred since the last check.

2. We use a *continuous-time Markov chain* (CTMC) to model an attribute when we do not know the number of changes but know the time passed from the since check.

**Example 6.1** Consider an auction where the policy allows selling goods if the rating of a seller is greater than 1. Currently, we know that a seller has rating 3, but this information is old, because two more deals have been accomplished since the last update of the rating. The auction has to know if the seller should be able to sell goods or, maybe, his activity should be suspended. Such situation could be modelled with the Discrete-Time Markov chain, where the nodes denote the values of the rating, and the edges show the possible changes of the rating.

**Example 6.2** As another example, we consider a location of a user in a building as an example of the attribute. The building contains six rooms: two laboratories (Lab1 (1) and Lab2 (3)), the office (2), the corridor (4), the bar (5), and WC (6). The security policy allows the access to the resource from rooms 1 and 2, and forbids the access from other rooms. The Markov chain representing the building contains six states (Figure 6.1a). The nodes represent the rooms, and the edges represent possible transitions (doors) between rooms. The modified Markov chain is presented in Figure 6.1b. Since states 3, 4, 5, and 6 are forbidden, they are replaced with a single absorbing state a.

Naturally, in order to apply Markov chain theory we need to identify the required transition and/or rate parameters. In many cases these values usually do not depend on time and could be computed in advance. Probably, additional resources could be needed to find these values, but such analysis has to be done only once during a short period.

When we know the probability of policy violation $P_{RM}$ using the Decision Theory we conclude that the access should be allowed if:

\[
(1 - P_{RM}) \times C^{dp} + P_{RM} \times C^{dn} > P_{RM} \times C^{fn} + (1 - P_{RM}) \times C^{fp}
\]  

(6.1)

### 6.4 Conclusion

In this work we presented an approach which helps to make decisions in run-time under uncertainties. The uncertainties we considered are caused by the not up-to-date attributes which are required for making the decisions. Two situations were considered: first, when we know the number of changes of attributes; second, when we know only the time passed after the last update. In both cases Markov Chains allow us predicting the probability of policy failure and risk assessment helps to make a rational decision about allowing or denying further access. We made a prototype of the proposed usage control model. The
results of testing showed, that the model gives accurate predictions, and the performance of the model is good enough for application in dynamic systems.

As the future work we would like to make the approach more effective by using the values (probabilities) found during the previous access check rather than recomputing the values from the beginning. Such online approach should significantly reduce the computational cost. Another possible direction of the model improvement is to consider cases of dependent attributes. This issue requires complex mathematical models for implementing correlated Markov chains.
7 Method for Risk and Costs in the SDLC with Particular Focus on Web Applications

Although the range of IT systems is very wide, and web applications are only one of many possible types of IT systems, there is an urgent need to secure web applications and services. This is mainly due to the number and importance of web applications and services are steadily increasing. Many other services and solutions are becoming web-based. Also, the overall structure of web applications and services is common and standard; many security related issues in different instances of web applications and services are the same. Web applications are moreover arguably the most important sources of insecurities regarding IT systems. The list of insecurities are being propagated in the Internet rapidly, and most attackers have information about how to try to penetrate such systems.

In order to reach an adequate level of security for the corresponding applications a risk aware approach to the SDLC is needed that enables a modular design of complex models for formal risk analysis. As elaborated both below and in Chapter 8, formal risk analysis provides a rigorous basis for analysis and reasoning about vulnerabilities, threats and risks.

The assurance that the potential threats do not yield unacceptable risks is a requirement in a security certification above a certain level of strength. Many of the formal tools for risk modelling were developed to fulfill the requirements of certification processes. Although there are several general purpose tools for risk modelling, risk evaluation, and treatment, there is still a gap between the theoretical concepts and their practical usability. In fact, there exist vast lists with potential risks, threats, and vulnerabilities for web based applications, but the currently available risk analysis methods and tools are missing a language to consistently represent or reason about them, or instantiate them with parameters or values. In order to make use of generic lists of web application threats, risks, and vulnerabilities, an enhanced logic-style notation would be helpful. This language can be used to represent attacks as sequences of steps, each one with the necessary prerequisites, resources, capabilities, and consequences.

We envision a generic tool for the generation of formal risk and threat models of web applications. The tool shall rely on a database of available known common vulnerabilities and threat scenarios and assist the user in the construction of a risk model and a parametrisation for a target system. Appropriate filter functions can be used for the presentation of the results of the analysis and for writing documentation. Therefore, the planned tool will be extending state-of-the-art methods and techniques for model-driven risk analysis.

The tool generates a risk model from a description of attack steps. Attack steps are simple rules formulated in an enhanced propositional logic style. The formulation has a clearly defined semantic and is powerful enough for the description of general reusable components. A specific threat or risk is an ordered sequence of possible attack steps. This approach reflects directly the heterogeneous and compositional structure of the original security problem.

We do not expect that such a tool will reach the expertise and know how of a security specialist but the tool can help the designer of a security system to accelerate cumbersome recurring tasks when modelling more complicate scenarios and to increase the overall level of security by avoiding errors and omissions in the analysis.

The background to our approach and the preliminary results are presented in the following sections. The reader is referred to Appendix D for a more detailed and elaborate presentation.

7.1 Main Concepts: Vulnerability, Loss Event, Attack, Step, and Capability

The notions of “vulnerability” [41, 14] and “loss event” are central to the analysis of risk in IT systems. For instance, ISO 27005 [33] defines risk as “the potential that a given threat will exploit vulnerabilities of an asset or group of assets and thereby cause harm to the organisation”. (The events that cause harm to the organisation are, of course, the loss events). But the relation between those two central concepts – vulnerability and loss event – is not straightforward; several concepts are needed to link them.

That is because the mere presence of a vulnerability does not ensure the existence of a possible loss event, and does not determine the probability or cost of such a loss event. They all depend, in particular,
on the possibility of the attacker to perform an *attack*, a series of different *steps* that lead to a loss event. It is not surprising that in the literature it is common to see an attack as the path of attack steps. This is done for instance in attack patterns, see for instance [5, 4, 10].

In other words, exploiting a vulnerability does not always imply directly a loss event, but can be a useful step in that direction. Each attack step may exploit one or several vulnerabilities, but in general the attacker may need some preconditions to access the vulnerability. We call them *capabilities*. For instance, to preform a step, the attacker may require certain credentials that he acquired in a previous step, or physical access to certain parts of a system. This notion of capability is often implicit in existing formalisations or representations of attacks or attack steps. A step that does not result in a loss event is performed by the attacker with the purpose of obtaining new capabilities that can be used in a later step. A step may provide him some knowledge or access to a resource in a way that then he is able to read certain data or to execute some code. Concisely, we will use capabilities to model the states of a transition system: the attack steps modify the capabilities of the attacker\(^1\). This explicit use of capabilities is useful to derive attack paths: an attacker carries out a series of attack steps in order to gain certain capabilities that allow him to perform further steps that finally allow him to reach his aim, e.g., to read or modify certain data, sabotage a certain system, etc.

Other factors that determine if a vulnerability gives rise to an attack step are the frequency with which threat agents\(^2\) try to exploit a vulnerability and the success rate, influenced by the difference between the threat agent's attack strength and the strength of the system to resist the attack.

The attack frequency, in turn, is determined by the threat agents' motivation to carry out an attack action and, again by the basic capabilities of a generic attacker. (If the attacker notices that he is unable to obtain basic capabilities, he will desist more rapidly from the attack).

Regarding the strength of the system to resist the attack, notice that the Open Group's Risk Taxonomy [59] states:

*Vulnerability is the probability that an asset will be unable to resist the actions of a threat agent. Vulnerability exists when there is a difference between the force being applied by the threat agent, and an object's ability to resist that force.*

In other words, even if a vulnerability is present, there may be also supplementary mechanisms in the system that resist the exploitation of the vulnerability.

### 7.2 Abstraction Levels, Vulnerability Assessments and Threat Modelling

Risk analysis of web applications can be performed at different abstraction levels and in different stages of the development life cycle.

Consider first, for instance, a directory and the collaboration services of a large enterprise with several hundreds of thousands of users and their devices. Here, it is impossible to analyse in detail the possible vulnerabilities of each involved component. Thus, it is necessary to abstract away from the single software vulnerabilities and, instead, use rough estimations of the resistance that the security controls offer. On the other hand, a rather detailed threat modelling is possible, including the attacker's motivation and resources, his possible attack steps, and the needed capabilities. This is reasonable since the deployment environment is known and the possible threat agent profiles can be anticipated. We say that in a global scenario, with many elements, an abstract *threat modelling* dominates the risk analysis.

Now consider the case of a single software application, say in the earlier phases of the SDLC, during the design or implementation, before the release. The development team has a lot of information regarding the software design decisions, the architecture, the actions taken during the SDLC, etc. In this case, a

---

\(^1\)In most cases, the step of one agent modifies only his own capabilities, but sometimes, it may change the global state of the system, changing thereby the capabilities of all attackers. For instance, the attacker may publish a list of passwords in a blackboard, a memory location that is visible by many attackers, or directly in a forum in Internet.

\(^2\)We talk of an “attacker” if we have one particular in mind, say performing a series of steps or attempting to achieve a certain goal. But we refer to “threat agents”, collectively, a set of attackers, each one following perhaps a different strategy, etc. The main point, formally, is that threat agents are independent of each other, and each one has different knowledge and capabilities.
more concrete and detailed vulnerability assessment becomes easier to obtain and dominates the risk analysis.
Therefore we propose a methodology for supporting model-based risk analysis for web services based on two central pillars:

**Rule-based vulnerability assessments.** It is assumed that following the best-practices regarding the activities to be performed during the SDLC, software artifacts\(^3\), and architectures of web services, the most common well known vulnerabilities can be avoided. During a typical system development many reasons may lead to deviations from the best practices. The type and amount of this deviation may be used to estimate the probabilities of possible vulnerabilities in the system. We propose to obtain rules that help us precisely in describing this dependency.

**Capability-based threat modelling.** Based on a collection of attack patterns and steps, for which capabilities, exploited vulnerabilities, and required attacker strength (expertise, resources, etc.) have been made explicit, threat modelling is to be supported.

A combination of both approaches yields flexible possibilities to perform risk analyses for systems of different purposes, varying sizes, and in different phases of the SDLC.

### 7.2.1 Vulnerability Assessment

The goals of the intended vulnerability assessment method are: (1) to develop methods and rules for obtaining useful indicators for the probability of the existence of particular vulnerabilities, and their characteristics in a given system, (2) to improve the prediction of the general security level of a given system, and (3) to determine the most important next steps required to secure the system, from an economic point of view.

The basic idea is to create a set of rules, perhaps based on a check-list or questionnaire, to detect deviations during the design or implementation of the target application from accepted best practices, in terms of the secure development life cycle activities, the security architecture chosen, and the software artifacts present, especially regarding the security mechanisms used.

The comparison comprises different aspects such as the basic web application architecture (e.g., 2-layer, 3-layer, . . .) with its basic components (e.g., application server, web server, database, . . .), the trust boundaries and interfaces within the application or system, the assets and their position in the application architecture among others. (For a more comprehensive overview see the Appendix D)

The resulting difference is used, first, to highlight components and design aspects that entail an increased vulnerability risk, second, to determine (via rules) indicators for particular possible vulnerabilities and their characteristics, and third, to understand the options with respect to their risk and cost effects and to choose the steps required to secure the system, from an economic point of view.

A rule or parameter-based approach to assess a web application's risk implies a major difficulty: one has to choose an appropriate number of parameters and cover different degrees of detail and abstraction. The set of parameters should cover the most significant aspects, but their number and collection still has to be efficiently manageable. The challenge is to find a good cost-benefit ratio, since in software security, tiny details can make a difference.

### 7.2.2 Capability-based Threat Modelling

As foundation for the methodology, a collection of attack patterns/steps has to be created. For each attack step, the following is made explicit: which (access) capabilities are required for carrying out the attack step; which capabilities the successful execution of the attack step yields; which vulnerabilities are exploited by the attack step and what attack strength (expertise, resources, etc.) is required to carry out the attack step.

We envision the following steps in capability-based threat modelling: First a model of the system under consideration is created that shows how each component of the system provides access to other parts

\(^3\)Software artifact is a tangible by-product created during the development of software and possibly documenting the activities of the SDLC. A software artifact can be for instance a requirements document.
of the system (including information assets). Then the system model is enriched with information about likely vulnerabilities as elicited by template-based vulnerability assessment. Afterwards, the attackers are modelled by defining their initial capabilities, motivations (in terms of capabilities they want to gain), and attack strengths. From the system owner’s view, loss events are modelled in terms of attackers gaining certain capabilities. Based on this information (i.e., the system model, information about likely system vulnerabilities, the attackers, and loss events as identified by the system owner) attack paths and success probabilities for these attack paths are then derived.

7.3 Preliminary Results

In this section we introduce preliminary results that go in the intended direction described above. On the one hand, we propose a new suitable risk ontology that is fine-grained enough to explicitly model attack steps. On the other hand we discuss possible formalisations that would allow to automatically check for paths leading to an attack given a catalogue of atomic attack steps.

7.3.1 A Suitable Risk Ontology

In order to support the methodology envisioned above, we introduce a suitable risk ontology. Several proposals for ontologies regarding information risk exist [40, 16], but do not adequately address certain concepts that are central to the methodology as envisioned above. For example, attacks usually encompass a series of actions that step by step lead an attacker toward his aim. With each step, the attacker gains certain capabilities (e.g., access to user credentials, access to the login page of an application, increased user rights, etc.), until he finally has gained enough capabilities to carry out his aim. While the concept of attacks progressing in stages is central to many risk ontologies [40, 16], the concept of capabilities is at best implicit. The ontology presented in the appendix makes the concept of capabilities explicit.

On the other hand, in order to achieve the high degree of reusability as envisioned above, the ontology must be centred around the most basic and general concepts of risk. The existing extensive collections of attack patterns [5, 4, 10] and weaknesses [41, 14] suggest that attack patterns (more generally, attack actions) and weaknesses/vulnerabilities are hot candidates for central, reusable concepts of a risk ontology. The ontology presented is centred around the notion of attacker actions. In this respect, the proposed ontology is very similar to that of Elahi et al. [16].

Figure D.1 on page 104 shows a UML class diagram of the proposed risk ontology. As elaborated above, central to the ontology is the concept of action with its dependence on capabilities, vulnerabilities and resources/expertise of the attacker: necessary precondition for the the execution of an action by an attacker against an object (more about objects below) is that the attacker has certain capabilities. Further, an action may also require certain resources/expertise on part of the attacker and the presence of certain vulnerabilities in the targeted object, but whether resources/expertise/vulnerabilities constitute necessary preconditions or are treated as factors that influence the success probability of an action is not predetermined by the ontology: there are numerous ways to reason about success probabilities and many methodologies will treat resources/expertise and presence of vulnerabilities as factors influencing probabilities rather than “binary” conditions that are or are not present. For a thorough description of the ontology see Appendix D.

7.3.2 A First Language Proposal for Attack Transitions

There exist a well documented set of vulnerabilities and attacks against web applications and services in catalogues like OWASP4. As argued above, many of these attacks are compositions of small steps (actions), which gradually increase an attacker’s capabilities. In this section we discuss a formal model for these actions that allows for automatically computation of all paths (sequences of steps) that lead to a given attack. This approach complements the formal approach to risk modelling by means of risk graphs that is presented in Chapter 8. As part of future work we will explore the potential of using risk graphs

---

4https://www.owasp.org/
for representing attacks, that in turn can be instantiated in established risk modelling approaches such as fault trees, event trees or CORAS threat diagrams.

Following the previously discussed ontology, in order to attack a given asset, an attacker must posses a set of capabilities (e.g., reading or writing rights in a given machine). That is, he needs a set \( L^A \) of capabilities that lead to attack \( A \). This process is often decomposable in many steps, that gradually allow the attacker to perform the attack by gaining intermediate capabilities, should exploitable vulnerabilities exist. In its simplest form, an action is a triple consisting of a list of capabilities, the success probability of the action and a new list of capabilities gained after the realisation of the step. We can express this formally as:

\[
a : L \xrightarrow{p} L'
\]

where \( a \) is a label for the attack step, \( L \) is the original list of capabilities, \( p \) is the probability that this particular attack step takes place\(^5\) and \( L' \) is the new list of capabilities.

To deduce what sequence of actions lead to \( A \), one could see the list \( L \) of capabilities as a conjunction of atomic capabilities and then embed the attack steps in propositional logic (initially omitting the probabilities). Constructing a proof of the derivation of \( L^A \) would imply the existence of a sequence of steps leading to \( A \), thus it is equivalent to the satisfiability of \( L^A \). In this model the probabilities of attack steps are attributes and are not needed to deduct a sequence of steps implementing a certain attack.

Alternatively, one could think of the attack steps as a state transition condition. That is, given a state consisting of the product of boolean variables representing all relevant capabilities (i.e. the capabilities appearing in some attack step) then each attack step defines a probability of going from some state \( S \) where \( L \) capabilities are true to a new state \( T \) where \( L' \) holds. In this setting, to find the sequence of attack paths leading to a set \( A \) of capabilities means querying for the reachability of states where \( L^A \) holds. This is formally equivalent to the satisfiability version discussed above, however it allows for more flexibility when describing more complex attack steps. It is for example reasonable to allow for more complex states including variables of integer type (within a certain finite range) that could model the amount of certain tokens available to the adversary, the amount of money required prior an attack etc.

Another advantage of this formalisation is that it is the natural input for modern model-checkers like nuSMV\(^6\), allowing checking more complex temporal logic propositions that go beyond simple reachability.

Although initially we plan to analyse collections of simple attack steps, an interesting and natural further step will be parameterising the capabilities to generalise them in contexts where similar assets/actors or other factors can be extracted. For example, the capability 'read password file' could be parametrised to represent this capability in a machine from a finite list: 'read password file(\( X \in \text{Machines} \))'. One could even go further and generalise this reading capability to a list of files: 'read \( Y \in \text{Files} \) (\( X \in \text{Machines} \))'. This would allow for defining also parametrised attacks of the form \( A(x_1, \ldots, x_n) \) where the variables are used to abstract from machines, users, etc.

It might be necessary for modelling complex attack scenarios to have a notion for the consumption of capabilities and resources. Examples are credentials for authentication, number of machines that are online, time for access, and money. If capabilities or resources can be gained and lost again along an attack path a state machine description of the steps of an attack would be needed. The probabilities associated with attack steps lead then to a Markov model of state transition.

### 7.4 Conclusion

In this chapter, we outlined our idea for a model-based risk analysis for web services. The proposed methodology is based on two central pillars: rule-based vulnerability assessments and capability-based threat modeling. A combination of both approaches yields flexible possibilities to perform risk analyses for systems of different purposes, varying sizes, and in different phases of the SDLC. For instance, in a global scenario, with many elements, a rather abstract threat modeling can produce valuable results for the overall risk analysis even though a detailed analysis of all software components is not possible. In case of a single software application, on the other hand, during the design or implementation phase, a

---

\(^5\)In general \( p \) could also be a vector of probabilities, this allows to distinguish between the attacker’s motivation and the technical probability of performing the attack for example.

\(^6\)http://nusmv.fbk.eu/
more detailed vulnerability assessment can cover significant risk factors although the later deployment environment and possible attacker profiles are not yet known. For an elaboration on future work we refer to Section D.5 of Appendix D.
8 Formal Foundation for Risk and Cost Aware SDLC

Risk analysis involves the process of understanding the nature of risks and determining the level of risk [31]. Risk modelling refers to techniques that are used to aid the process of identifying, documenting and estimating likelihoods and consequences of unwanted incidents. An unwanted incident is an event that harms or reduces the value of an asset (such as availability or confidentiality), and a risk is the likelihood of an unwanted incident and its consequence for a specific asset. In this chapter we present an approach to risk modelling referred to as risk graphs [9].

A formal foundation for risk management facilitates risk analysis by providing a rigorous basis for analysing and reasoning about risks. Risk graphs support various kinds of reasoning about risk, and in this chapter a calculus is introduced with rules for such reasoning, the soundness of which can be shown with respect to the underlying semantics. By providing a formal semantics for risk graphs, we are provided a formal foundation for reasoning about risk, and therefore also a formal foundation for risk management in general.

In Section 8.1 we present the syntax and semantics of risk graphs, and in Section 8.2 we present a calculus for reasoning about likelihoods. In Section 8.3 risk graphs are extended to so-called dependent risk graphs with support for modelling dependencies and for conducting modular risk analysis. Section 8.4 briefly discusess the instantiation of the risk graph formalisation and calculi in state-of-the-art risk modelling techniques. For a more detailed introduction to risk graphs and their formalisation, the reader is referred to Appendix E. The appendix moreover introduces the generalisation of risk graphs to deal with changing risks, and it demonstrates and exemplifies the instantiation of risk graphs in CORAS [40].

8.1 Risk Graphs

A risk model is a structured way of representing an unwanted incident and its causes and consequences by means of graphs, trees or block diagrams [50]. We introduce risk graphs as an aid for structuring events and scenarios leading to incidents, and for estimating likelihoods of incidents. There exist several modelling techniques that can be used for such structuring of scenarios and incidents, and for the reasoning about likelihoods of incidents, for example fault trees [27], event trees [29], attack trees [53], cause-consequence diagrams [44], Bayesian networks [6] and CORAS threat diagrams [40]. Risk graphs can be understood as a common abstraction of these modelling techniques [9]. By giving formal semantics to risk graphs, we thereby also provide a risk model semantics that can be used to explain and reason about several established approaches to risk modelling. Hence, the formalisation of risk graphs can serve as a formal foundation for risk management with wide flexibility in the chosen approach to risk and threat modelling.

A risk graph consists of vertices (representing threat scenarios) and a finite set of directed relations (representing the “leads to” relationship) between them. An example risk graph is shown in Figure 8.1. Each vertex in a risk graph is assigned a set of likelihood values representing the estimated likelihood for the scenario to occur. The assignment of several likelihood values, typically a likelihood interval, represents underspecification of the likelihood estimate. A relation from vertex \( v \) to vertex \( v' \) means that \( v \) may lead to \( v' \). Also the relations can be assigned likelihood sets. These are conditional likelihoods

\[
\begin{align*}
&v_1 \left[ P_1 \right] \xrightarrow{P_x} v_2 \left[ P_2 \right] \\
&v_2 \left[ P_3 \right] \xrightarrow{P_t} v_3 \left[ P_t \right] \xrightarrow{P_s} v_4 \left[ P_t \right] \\
&v_4 \left[ P_t \right] \xrightarrow{P_s} v_5 \left[ P_t \right] \xrightarrow{P_t} v_6 \left[ P_t \right] \\
&v_6 \left[ P_t \right] \xrightarrow{P_s} v_7 \left[ P_t \right] \\
&v_7 \left[ P_t \right] \xrightarrow{P_t} v_8 \left[ P_t \right] \\
&v_8 \left[ P_t \right] \xrightarrow{P_t} v_9 \left[ P_t \right] \\
&v_9 \left[ P_t \right]
\end{align*}
\]

Figure 8.1: Risk graph
that specify the likelihood for a scenario leading to another scenario when the former occurs. One threat scenario may lead to several other threat scenarios, so the probabilities on the relations leading from a threat scenario may add up to more than 1. A risk graph is furthermore allowed to be incomplete in the sense that a given threat scenario may lead to more scenarios than what is accounted for in the risk graph. The probabilities of the relations leading from a threat scenario may for this reason also add up to less than 1.

Formally, the syntax of a risk graph is a set \( D \) of elements \( v \). An element is a vertex \( v \) or a relation \( v \rightarrow v' \). Let \( P \subseteq [0, 1] \) denote a probability set. We then write \( v(P) \) to indicate that the probability set \( P \) is assigned to \( v \). Similarly, we write \( v \xrightarrow{P} v' \) to indicate that the probability set \( P \) is assigned to the relation from \( v \) to \( v' \). If no probability set is explicitly assigned, we assume the probability set assigned to the element to be \([0, 1]\), i.e. that the probability is fully underspecified.

Risk graphs are used for the purpose of documenting and reasoning about risks, particularly the documentation and analysis of threat scenarios and unwanted incidents and their likelihoods. The approach of [9] assumes that scenarios and their probabilities are semantically represented by a probability space [15] on traces of events. We let \( \mathcal{H} \) denote the set of all traces (both finite and infinite) and \( \mathcal{H}_0 \) the set of all finite traces. A probability space is a triple \( (\mathcal{H}, \mathcal{F}, \mu) \). \( \mathcal{H} \) is the sample space, i.e. the set of possible outcomes, which in our case is the set of all traces. \( \mathcal{F} \) is the set of measurable subsets of the sample space, and \( \mu \) is a measure that assigns a probability to each element in \( \mathcal{F} \). The semantics of a risk graph is statements about the probabilities of the trace sets that represent vertices or the composition of vertices. In other words, the semantics is a set of statements about the measure \( \mu \).

For composition of vertices, \( v \sqcap v' \) denotes the occurrence of both \( v \) and \( v' \) where the former occurs before the latter. We let \( v \sqcup v' \) denote the occurrence of at least one of \( v \) and \( v' \). A vertex is atomic if it is not of the form \( v \sqcap v' \) or \( v \sqcup v' \). We use lower case \( v \) as the naming convention for arbitrary vertices, and upper case \( V \) as the naming convention for the set of finite traces representing the vertex \( v \).

When defining the semantics of risk graphs we use the auxiliary function \( \text{tr}(\_\,) \) that yields a set of finite traces from an atomic or combined vertex. Intuitively, \( \text{tr}(v) \) is the set of all possible traces leading up to and through the vertex \( v \), without continuing further. The function is defined by

\[
\text{tr}(v) \overset{\text{def}}{=} \mathcal{H}_0 \ni v \quad \text{when } v \text{ is an atomic vertex}
\]

\[
\text{tr}(v \sqcap v') \overset{\text{def}}{=} \text{tr}(v) \ni \text{tr}(v')
\]

\[
\text{tr}(v \sqcup v') \overset{\text{def}}{=} \text{tr}(v) \cup \text{tr}(v')
\]

where \( \ni \) is the operator for sequential composition of trace sets, for example weak sequencing in UML sequence diagrams [24]. Notice that the definition of the composition \( v \sqcap v' \) does not require \( v \) to occur immediately before \( v' \). The definition implies that \( \text{tr}(v \sqcap v') \) includes traces from \( v \) to \( v' \) via finite detours.

A probability interval \( P \) assigned to \( v \), denoted \( v(P) \), means that the likelihood of going through \( v \) is a value \( p \in P \), independent of what happens before or after \( v \). The semantics of a vertex is defined by

\[
[v(P)] \overset{\text{def}}{=} \mu_v(\text{tr}(v)) \in P
\]

where the expression \( \mu_v(S) \) denotes the probability of any continuation of the trace set \( S \subseteq \mathcal{H} \), and is defined as

\[
\mu_v \overset{\text{def}}{=} \mu(S \ni \mathcal{H})
\]

A probability interval \( P \) assigned to a relation \( v \rightarrow v' \) means that the likelihood of \( v' \) occurring after an occurrence of \( v \) is a value in \( P \). This likelihood is referred to as the conditional likelihood. The semantics of a relation is defined by

\[
[v \xrightarrow{P} v'] \overset{\text{def}}{=} \mu_v(\text{tr}(v \sqcap v')) \in \mu_v(\text{tr}(v)) \cdot P
\]

Our definitions of interval arithmetic in the setting of risk graphs are given in Fig. E.2 of Appendix E.

The semantics \([D]\) of a risk graph is the conjunction of the expressions defined by the elements in \( D \), formally defined as

\[
[D] \overset{\text{def}}{=} \bigwedge_{e \in D}[e]
\]

A risk graph is said to be correct (with respect to the world or a specification of the relevant part of the world) if each of the conjuncts of \([D]\) is true. We say that \( D \) is inconsistent if it is possible to deduce \( \text{False} \) from \([D]\). Notice that \([0] = \text{True} \)
8.2 Reasoning about Likelihoods

In this section we introduce rules for calculating probabilities of vertices in risk graphs. The first rule is referred to as the relation rule, and captures the conditional likelihood semantics of a risk graph relation. For a vertex \( v \) that leads to \( v' \), the vertex \( v \oplus v' \) denotes the occurrences of \( v' \) that happen after an occurrence of \( v \).

**Rule 8.1 (Relation)** If there is a direct relation from \( v \) to \( v' \), we have:

\[
\frac{v(P)}{(v \cap v')(P \cdot P')}
\]

The second rule is referred to as the mutual exclusive vertices rule, and yields the probability of either \( v \) or \( v' \) occurring when the two vertices are mutually exclusive:

**Rule 8.2 (Mutually exclusive vertices)** If the vertices \( v \) and \( v' \) are mutually exclusive, we have:

\[
\frac{v(P)}{(v \sqcup v')(P + P')}
\]

The third rule is referred to as the statistically independent vertices rule, and yields the probability of either \( v \) or \( v' \) occurring when the two vertices are statistically independent:

**Rule 8.3 (Statistically independent vertices)** If vertices \( v \) and \( v' \) are statistically independent, we have:

\[
\frac{v(P)}{(v \sqcup v')(P + P' - P \cdot P')}
\]

Consistency checking of risk models is important, as it is a useful means for detecting errors or misunderstandings of the risk estimates that are documented during a risk analysis. The basis for the consistency checking is the likelihood values that are already assigned to the vertices and relations of a risk graph. The guidelines for consistency checking depend on whether the risk graph in question is complete, and whether the likelihoods are given as exact probabilities or as probability intervals. In complete diagrams all possible ways in which a scenario may arise are shown, such that for example the likelihood of vertex \( v_3 \) in Figure 8.1 can be completely determined from vertices \( v_1 \) and \( v_2 \). For incomplete diagrams, there may be further scenarios that are not accounted for, in which case we can only determine the lower bound of likelihoods based on the preceding vertices.

As a concrete example, consider the risk graph depicted in Figure 8.2. The diagram shows possible scenarios that may damage the integrity of electronic health records (EHRs). The upper part addresses the problem of network connection, the failure of which may lead to disruption in transmission of EHRs. Assuming that this part of the diagram is complete, the likelihood of the latter scenario can be obtained from the likelihood 0.2 of the former scenario and the conditional likelihood 0.4 that the former scenario leads to the latter when the former occurs. Specifically, by Rule 8.1 the likelihood is \( 0.2 \cdot 0.4 = 0.08 \).

![Figure 8.2: Example risk graph from eHealth domain](image-url)
The lower part of the diagram addresses the problem of irregular handling of EHRs by the general practitioners (GPs). The GPs are part of the administrative domain, but need to comply with policies and regulations. If the GPs have possibilities to not adhere to the policies, scenarios may arise that compromise the integrity of EHRs, for example that the records are stored and edited locally as shown in the risk graph.

Assume now that the scenarios Disruption in transmission of EHRs and GP stores and maintains EHRs on local computer are statistically independent. We may then use Rule 8.3 to calculate the likelihood of Integrity of EHRs are compromised. First, we use Rule 8.1 to calculate the contribution from the upper part which yields $0.08 \cdot 0.05 = 0.004$. This is then the likelihood of the occurrences of Integrity of EHRs are compromised that are due to Disruption in transmission of EHRs. Similarly, the occurrences that are due to GP stores and maintains EHRs on local computer is given by $0.05 \cdot 0.3 = 0.015$. By Rule 8.3 we aggregate the likelihoods and get $0.004 + 0.015 - 0.004 \cdot 0.015 = 0.01894$.

If the risk graph is complete it shows all possible ways in which the scenario Integrity of EHRs are compromised can arise. In that case the likelihood equals 0.01894. In Figure 8.2 the assigned likelihood is 0.04 which means that there are inconsistencies that must be resolved. If, on the other hand, the diagram is incomplete (which is more reasonable to assume), we only know that 0.01894 is the lower bound of the likelihood of the scenario in question. The assigned estimate of 0.04 is then consistent with the other estimates in the diagram.

The reader is referred to Appendix E for the concrete guidelines for likelihood estimation and consistency checking, and for examples of likelihood calculation and consistency checking of risk graphs.

### 8.3 Reasoning about Dependencies

When systems are mutually dependent, a threat towards one of them may realise threats towards the others [48, 49]. Rinaldi et al. [49] argue that mutually dependent infrastructures must be considered in a holistic manner. Within risk analysis, however, it is often not feasible to analyse all possible systems that affect the target of analysis as once. To remedy this, we may use a modular approach [9]. By modular risk analysis we refer to a process in which separate parts of a system is analysed independently, and in which there is support for combining separate analysis results into an overall picture for the whole system.

For this purpose we present in this section dependent risk graphs. In the context of dependent risk graphs, we refer to risks graphs as defined in Section 8.1 as basic risk graphs. A dependent risk graph is similar to a basic risk graph, except that its elements (vertices and relations) $D$ are divided into two disjunct sets, namely the assumption $A$ and the target $T$. The assumption describes the assumptions on which the risk estimates of the target depend. We denote a dependent risk graph by $A \vDash T$. The reader is referred to Appendix E for a number of syntactical constraints that apply to dependent risk graphs.

Before extending the semantics of basic risk graphs, we define the notion of interface between subgraphs. An interface is between sets of elements that may not fulfill the well-formedness requirement for basic risk graphs presented in Appendix E. Given two sub-graphs $D$ and $D'$, $i(D, D')$ denotes the interface of $D$ towards $D'$. This interface is obtained from $D$ by keeping only the vertices and relations that $D'$ depends on directly.

$$i(D, D') \overset{\text{def}}{=} \{ v \in D \mid \exists v' \in D' : v \rightarrow v' \in D \cup D' \} \cup \{ v \rightarrow v' \in D \mid v' \in D' \}$$

A dependent risk graph $A \vDash T$ means that all sub-graphs of $T$ that only depends on the parts of $A$’s interface towards $T$ that actually hold must also hold. Formally, the semantics is defined as follows:

$$[A \vDash T] \overset{\text{def}}{=} \forall T' \subseteq T : [i(A \cup T \setminus T', T')] \Rightarrow [T']$$

By the definition, if all of $A$ holds ($[A]$ is true), then all of $T$ must also hold. Note that if the assumption of a dependent risk graph $A \vDash T$ is empty, i.e. $A = \emptyset$, it means that we have the risk graph $T$. In other words, the semantics of $\emptyset \vDash T$ is the same as the semantics of $T$.

In order to proceed to the rules for dependency reasoning, we first precisely define what is meant by dependency. The relation $D \not\vDash D'$ means that $D'$ does not depend on any vertex or relation in $D$. This means that $D$ does not have any interface towards $D'$, and $D$ and $D'$ have no common elements. Note that $D \not\vDash D'$ does not imply $D' \not\vDash D$. 

NESSoS - 256980
\[ D \uparrow D' \overset{\text{def}}{=} D \cap D' = \emptyset \wedge i(D, D') = \emptyset \]

**Rule 8.4 (Assumption independence)** If we have deduced \( T \) assuming \( A \) and \( T \) is independent of \( A \), then we may deduce \( T \):

\[
\frac{A \uparrow T}{\Box T}
\]

From the second premise it follows that \( T \) does not depend on any element in \( A \). Since the first premise states that all sub-graphs of \( T \) hold that depend on the parts of \( A \)'s interface towards \( T \) that hold, we may deduce \( T \).

**Rule 8.5 (Assumption simplification)** Parts of the assumptions that are not connected to the rest can be removed:

\[
\frac{A \cup A' \uparrow T}{A' \uparrow T}
\]

The second premise implies that neither \( A' \) nor \( T \) depends on any element in \( A \). The validity of the first premise is therefore independent of \( A \), in which case also the conclusion is valid.

**Rule 8.6 (Target simplification)** Parts of the target can be removed as long as they are not situated in-between the assumption and the part of the target that we wish to keep:

\[
\frac{A \uparrow T \cup T' \quad T' \uparrow T}{A \uparrow T}
\]

The second premise implies that \( T \) does not depend on any element in \( T' \), and therefore does not depend on any element in \( A \) via \( T' \). Hence, the validity of the first premise implies the validity of the conclusion.

To make use of these rules when scenarios are composed, we also need a consequence rule.

**Rule 8.7 (Assumption consequence)**

\[
\frac{A \cup A' \uparrow T}{A' \uparrow T}
\]

By the latter rule, if all sub-graphs of \( T \) hold that depend on the parts of the interface of \( A \cup A' \) towards \( T \) that hold, and we can show \( A \), then it follows that \( T \).

### 8.4 Instantiations of Risk Graphs

As argued in Section 8.1, risk graphs can be understood as a common abstraction of several well-known approaches to risk modelling. By the formalisation of risk graphs presented in this chapter, we have therefore provided a formal foundation for risk management with a wide flexibility in terms of the preferred concrete techniques for risk modelling and reasoning about risk.

In Appendix E we explain and demonstrate how risk graphs can be instantiated in the CORAS approach [40] as a concrete example. The appendix moreover presents risk graphs generalised to the setting of changing and evolving systems in which also risk are changing and evolving and modelled as such. Also this formalisation of risks graphs with change is demonstrated with the CORAS instantiation.

### 8.5 Conclusion

In this chapter we have introduced risk graphs as a technique for the identification and documentation of risk to support risk analysis in particular and risk management in general. Specifically, risk graphs support the structuring of events and scenarios that lead to unwanted incidents, and they support the estimation and reasoning about likelihoods.
The formal foundation for risk management presented in this paper includes a formal semantics of risk graphs in terms of a probability space on traces of events. A set of rules for reasoning about likelihoods have been presented, the soundness of which can be shown with respect to the semantics. Moreover, the guidelines for consistency checking likelihood estimates assigned to diagram elements are based on the likelihood calculus.

Explicitly modelling and reasoning about assumptions and dependencies in risk analyses is supported by dependent risk graphs. Furthermore, dependent risk graphs supports a modular approach to risk modelling and analysis in which separate parts of a complex target of analysis with mutual dependencies can be analysed separately. The reasoning about dependencies is supported by the presented calculus.
9 Links With Other WPs

In this chapter we briefly outline the links with WP10 and the other technical work that is conducted in NESSoS. We focus on areas of potential integration between WPs that can be the topic of future research. The methodology to support a risk and cost aware SDLC that is developed in WP10 spans the orthogonal activities of WP6, WP7 and WP8. The risk and cost assessments will drive the SDLC by identifying potential security risks and by identifying options for risk mitigation that will ensure return on the investment in security.

Considering the Security Requirements for Services of WP6, the outcomes from security requirements analysis can be used as input to the identification of several threat and risk factors that later need to be managed during the SDLC. Moreover, during a security requirements analysis, one needs to consider the level of risk and cost that a system is exposed to, as well as cost efficient options for mitigating unacceptable risks. Finally, most (if not all) methods and languages that are developed in WP6 can be aligned with the overall process of the risk and cost aware SDLC that is being developed in WP10. By exploring the potential for security requirements engineering and risk and cost assessment to leverage on each other, we thereby also explore the potential for risk and cost assessment to indirectly affect all the SDLC phases that follow the security requirements elicitation.

Hence, the risk and cost assessments indirectly affect the Secure Service Architecture and Design of WP7 via the support for the identification of relevant security requirements that are later integrated in the design models. However, we also aim at investigating the direct links between WP10 and WP7 by conducting risk and cost assessments based on the architecture and design models. First, the model-driven approach to risk and cost assessment of WP10 is already to begin with well aligned with the UML-based approaches of WP7. Model-driven risk and cost assessment make systematic use of service architecture and design models in the identification of potential threats and vulnerabilities. In turn, the options for risk mitigation at justifiable cost that are identified during the risk assessment can drive the further architecture and design process. Second, considering the inherent evolutionary characteristic of FI and how this is addressed by the research topics of WP7 and WP10, an interesting topic to investigate is the support for assessing evolving risks and costs on the basis of the explicit modeling of evolving services that is conducted in WP7. With evolution and traceability of changes specified in the architecture and design models the methods for assessing and modeling changing and evolving risks may support the secure evolution of services in the architecture and design phase of the SDLC.

As to the Programming Environments for Secure and Composable Services of WP8, one interesting link is seen in the focus of WP8 on the composition of services demands (Task 8.2) and in the WP10 techniques for measuring the security level of the composed systems. Although the absolutely secure system is desirable, such condition is hardly achievable. This observation is even more relevant for a composed system, where cascade effects and dependencies must be properly taken into account both at design time and run time. Thus, next to the elimination of identified design flaws, the risk assessment of the composite service is required for the assurance that a high level of security is provided. Similar observation could be made also about the information-flow analysis, where risk assessment and design checking techniques supplement each other in order to build a trustworthy system.

Also run-time monitoring middleware developed in WP8 are related to the risk monitoring. In particular, risk can be integrated into the monitoring and enforcement infrastructure of continuous usage control if this infrastructure is not entirely trusted. There is also a reverse link between monitoring and risk assessment. Risk assessment requires the up-to-date knowledge about occurring events, e.g., in order to know the probability of such events. Therefore, monitoring system is essential for reliable continuous risk assessment. A further link is the enforcement techniques developed by WP8 that could be used by the cost-benefit analysis in order to minimise risk caused by certain threats paying minimal cost.

Finally, considering the Security Assurance for Services of WP9, one specific goal there is to articulate and improve security metrics to optimise the potential of quantifying assurance levels, based on such metrics. Thus, both WP9 and WP10 aim at determining the current "security status" of an application. While WP9 tries to achieve this by determining to what extent the application is secure ("ensure confidence about the security level"), the vulnerability assessment method of WP10 focuses on the how insecure the application is, or, to be more precise, it wants to "identify possible vulnerabilities of a given system".

Both methods want to give feedback at each stage of the software life cycle and therefore a common
prerequisite is to obtain the necessary data on which this determination can be based. There is most probably an overlap between the "metrics describing the overall security of the software" from WP9 and the "catalogue of most significant indicators" for application vulnerabilities from WP10.

A second link between the two work packages is use of testing methodologies, especially black- and white-box testing, penetration testing and code analysis to support the quantification of assurance (WP9) or risk (WP10) levels.
10 Conclusions

Traditionally, risk management is seen by industry as a process aiming at an efficient balance between realising opportunities for gains while minimising vulnerabilities and losses. It is an integral part of management practice and an essential element of good corporate governance. It is generally accepted that risk analysis, assessment and mitigation are part of risk management. Risk management is a recurrent activity that copes with the analysis, plan, implementation, control and monitoring of implemented measurements and the enforced security policy. In contrast, risk analysis is executed in certain time intervals and – up to the next execution – provides a temporary view of the risks. For the Future Internet applications, where dynamicity, context-awareness and non-deterministic system behaviour are more rule than exception, we must be able to identify and assess risks at all stages of a Software Development Life Cycle (SDLC). We argue therefore that risk management should be as flexible as the SDLC, following an incremental, modular and component-based approach. It should be based on continuous feedback from run-time risk monitoring and mitigation.

In this report we have provided an initial risk and cost aware SDLC that is our first attempt to integrate security risk management and security cost management into the phases of an iterative SDLC. When defining the risk and cost aware SDLC we do not assume any specific SDLC, risk management methodology or cost model as its starting point; we rather make the general assumption that the SDLC is iterative and that the risk management methodology conforms to the ISO 31000 risk management standard [31]. There is hence a number of existing approaches that can instantiate the risk and cost aware SDLC that we propose. We provide guidelines and examples on how to tailor the risk and cost aware SDLC to a specific organisation with specific choices of SDLC, risk management methodology, cost models, and so forth. For this task we look to the ISO/IEC 27001 [32] standard.

In conducting the security risk and cost assessment activities of the development lifecycle of FI software services and systems we leverage on existing risk management frameworks to the extent that they provide adequate methods to assess risks and identify treatments for unacceptable risks. Because the business oriented perspective and adequate methods for demonstrating return on investment in security is provided little support in existing frameworks we have proposed an initial systematic approach to assess costs and benefits of identified treatment options for unacceptable security risks. The proposed method has been designed to be embedded into ISO 31000 standard on risk management by extending the risk treatment phase.

In this report we have moreover highlighted the increasing number and importance of web applications and services and their relevance in the FI setting. More and more services and solutions are becoming web-based and there an urgent need to maintain their security. By observing that the overall structure of web applications and services is common and standard and that many security related issues in different instances of web applications and services are the same, we have proposed some initial solutions towards vulnerability assessment and capability-based threat modelling as cornerstones of a method for risk and cost assessment in the SDLC of web applications.

While accommodating the security risk and cost assessment to an iterative SDLC of FI software services and systems, there is still a need for support for specific challenges that are imposed by the nature of the FI. In this report we have presented several supporting methods and techniques to handle the heterogeneous, modular and dynamic nature of the FI in which security risks and costs rapidly evolve. This includes a method for the continuous assessment of risks in run-time, methods and modelling support for assessing changing and evolving risks, and methods for reasoning about risks in a modular way. Although these methods can be applied free standing to cope with specific risk and cost analysis needs, we will in future work investigate their orchestration in the risk and cost aware SDLC and evaluate them with respect to selected FI application scenarios. Future work also includes exploring the links with WP10 and the other technical WPs and how to use the security risk and cost assessment to support the various activities in engineering secure FI software services and systems.
References


A Toward a Risk and Cost Aware Software Development Life Cycle

Mass Soldal Lund  Bjørnar Solhaug

Abstract: This paper presents a first attempt toward a Risk and Cost aware Software Development Life Cycle (RC-SDLC). For system development on the one hand and security risk analysis on the other hand there already exist a number of established processes, methods and techniques. The challenge addressed in this paper is how to provide a RC-SDLC by integrating security risk management and security cost management into all phases of the SDLC. During a RC-SDLC, several specific methods and techniques for security risk analysis and security cost analysis can be utilised to support the various phases and tasks. This paper explains how a selection of specific methods for security risk and cost analysis can be instantiated in the overall RC-SDLC.

Keywords: Security risk, security cost, risk management, information security management, iterative system development

A.1 Introduction

In this paper we attempt the first steps toward a Risk and Cost aware Software Development Life Cycle (RC-SDLC) for the development of secure software services and systems. Both system development and risk analysis of existing systems are well-understood; the challenge in providing a risk and cost aware SDLC is to integrate security risk management and security cost management in all the phases of the SDLC in such a way that security risk and security cost considerations are drivers of the development and not something that is applied afterwards.

When providing a SDLC like the one we are aiming at, there are two pitfalls we should seek to avoid: 1) Making it so general that it does not provide any real guidance as to how to put it into practice, and 2) making it so specific that it is not applicable because of the assumptions and constraints placed upon potential adopters. Arriving at the right balance is a challenging task. To avoid the latter pitfall, we do not assume any specific SDLC, risk management methodology or cost model as the starting point of the RC-SDLC, but we make the general assumption that the SDLC is iterative and that the risk management methodology conforms to the ISO 31000 risk management standard. There is hence a number of existing approaches that can instantiate the RC-SDLC that we propose. In order to avoid the former pitfall, we provide guidelines and examples on how to tailor the RC-SDLC to a specific organisation with specific choices of SDLC, risk management methodology, cost models, etc. For this task we look to the ISO/IEC 27001 standard so that our guidelines are specialisations of a suitable subset of the requirements given in this standard.

The remainder of this paper is structured as follows: In Sections A.2 and A.3 we give brief introductions to the ISO 31000 and ISO/IEC 27001 standards, respectively. In Section A.4 we review iterative SDLCs. In Section A.5 we provide the overall process of our risk and cost aware SDLC. In Section A.6 we relate the overall process to system development methodologies, and in Section A.7 we demonstrate instantiations of the RC-SDLC with specific methods for risk and cost analysis methods. Finally, in Section A.8, we conclude.

A.2 Risk Management

Risk Management is guided by the ISO 31000 risk management standard [6]. Figure A.1 is adapted from this standard and illustrates the seven activities of the risk management process. The five activities in the middle constitute the core activities of a risk analysis, and are described as follows:
• **Establish 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, recognise 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 remaining two activities are continuous activities of the overall risk management process, and are described as follows:

• **Communicate and consult** are the continual and iterative processes an organisation conducts to provide, share or obtain information, and to engage in dialogue with stakeholders about risk management.

• **Monitoring** involves the continuous checking, supervising and critically observing the risk status in order to identify changes from the performance level required or expected, whereas **review** focuses on the activity undertaken to determine the suitability, adequacy and effectiveness of the subject matter necessary to achieve established objectives.

ISO 31000 comes with no guidance on concrete risk analysis techniques and must be operationalised by a risk analysis method implementing the standard (for example OCTAVE [1], CORAS [9] or CRAMM [4]) in order to be put into practice. While a risk analysis method provides methodological advice on how to carry out the various activities of risk management, a risk analysis technique is more narrow in the sense that it addresses only some aspects of the risk analysis process. A risk analysis method typically makes use of one or more risk analysis techniques. It follows that ISO 31000 moreover provides no guidance on risk analysis techniques in the context of a Software Development Life Cycle.

### A.3 Information Security Management

The ISO/IEC 27001 standard [7] defines an Information Security Management System (ISMS) as “the part of the overall management system, based on a business risk approach, to establish, implement, operate,
The standard describes a Plan-Do-Check-Act process model for ISMS process, as shown in Figure A.2, which is adapted from the standard. The stages of the process are described as follows:

- **Plan** is to establish ISMS policy, objectives, processes and procedures relevant to managing risk and improving information security to deliver results in accordance with an organisation's overall policies and objectives.
- **Do** is to implement and operate ISMS policy, controls, processes and procedures.
- **Check** is to assess and measure process performance against ISMS policy, objectives and practical experience.
- **Act** is to take corrective and preventive actions to achieve continual improvements of the ISMS.

In relation to ISO 31000 we find the risk analysis activities of *Establish the context, Risk identification, Risk estimation, Risk evaluation* and *Risk treatment* in the Plan/Establish stage; we find the activities of *Monitor and Review* in the Check/Monitor and Review stage; and we find the activity of *Communicate and consult* in the Act/Maintain and Improve stage.

### A.4 Iterative System Development

In the context of iterative system development we consider three classes of Software Development Life Cycles to be of relevance, ranging from what can be viewed as “heavy” to “light” methodologies:

1. Spiral model
2. Unified process
3. Agile methods

#### A.4.1 Spiral Model

The spiral model [3] evolved from the waterfall model as a system development process that assumed the benefits of conducting the system development in an incremental and iterative fashion, and that sought to reduce the (project) risks inherent in the application of the waterfall model.

The process (see Figure A.3) iterates through a number of activities in an outward spiral of increased efforts and costs:

- Commit to an approach for the next iteration.
• Determine objectives, alternatives and constraints.
• Identify and resolve risks, and evaluate alternatives.
• Develop and verify.
• Plan the next iteration.
• Review and partition.

The iterations around the spiral correspond to phases of the waterfall model:
• Round 0: Feasibility study.
• Round 1: Concept of operations.
• Round 2: Top-level requirements specification.
• Round 3: Software product design.
• Round 4: Detailed design and implementation.

From this we can see that the iterative approach of the spiral model mainly iterates with respect to planning, and not with respect to activities such as requirement specification, design and implementation. In the context of this paper it is important to notice that “risk analysis” in the spiral model refers to analysis of project risks, while in relation to the RC-SDLC we will in general understand “risk analysis” as analysis of security risks.

A.4.2 Unified Process

There are several variants of the Unified Process, including Agile Unified Process (AUP), Basic Unified Process (BUP), Enterprise Unified Process (EUP), Essential Unified Process (EssUP), Open Unified Process (OpenUP) and Rational Unified Process (RUP) [11]. We will not give a presentation of all these, but use Rational Unified Process (RUP) [8] – probably the best known and most used variant – as a representative of the “Unified Processes”.

The Rational Unified Process is structured around two dimensions (see Figure A.4). One dimension progresses along the lifespan for the system development, divided into the four phases Inception, Elaboration, Construction and Transition; the other dimension is activities, or disciplines, such as requirements, design, implementation and test, that run across all the phases. This structure comes from the recognition that iterative and incremental approaches reduce the (project) risks involved in a software development project, and the recognition that requirements and designs usually undergo changes and refinements as a project proceeds (“requirements creep”).

The process is highly iterative, with the possibility of several iterations within each of the phases. As can be seen from the figure, these iterations are iterations of the actual development, so each iteration includes all cross-cutting activities (as opposed to the spiral model where the iterations are performed with respect to planning and not development as such). The process emphasises component based development and continuous integration as a way of reducing (project) risk. This also leads to a natural partitioning of the work, since development of components can be associated with iterations.

A.4.3 Agile Methods

Agile system development is a collective name of a number of system development processes such as eXtreme Programming (XP) [5], Scrum [10] or Agile Model-Driven Development (AMDD) [2]. The basis is a highly iterative process with short iterations focused around building working code and developing functionality incrementally. The purpose is to be readily adaptive to changes in requirements, environment and technology. The processes often disregard heavy specification and documentation practices, and requirement capture and specification are concentrated around user stories, model sketches and use cases.
Simulations, models, benchmark
Risk analysis
Prototyping
Concept
Requirement and lifecycle plan
Development plan
Integration and test plan
Cumulative cost
Progress through steps
Evaluate alternatives, identify, resolve risks
Plan next phase
Determine objectives, alternatives, constraints
Commit
Review
Partition
Develop and verify
Evaluate alternatives, identify, resolve risks

Figure A.3: Spiral model

Inception
Elaboration
Construction
Transition

Business modeling
Requirements
Analysis & Design
Implementation
Test
Deployment

Figure A.4: Rational Unified Process
A.5 Overall Process of the Risk and Cost Aware SDLC

In a risk and cost aware SDLC there will necessarily be activities directly related to risk and cost, such as risk and cost analysis, as well as activities that are not directly related to risk and cost (requirement specification, design, test, etc.) but on which risk and cost will have an impact. Detailed discussions on activities only indirectly related to risk and cost is outside the scope of this work; in this paper the focus will be on the risk and cost activities, as well as the placement of such activities in relation to other SDLC activities.

Figure A.5 illustrates a generic iterative SDLC made risk and cost aware by introduction of the (generic) activities Risk analysis and Cost/benefit analysis. In this illustration we assume the general activities such as Requirements, Design, etc. to comprise also security related sub-activities so that for example the activity Requirements includes the definition of security requirements, and so fourth. Based on this assumption we have indicated in the figure where the work of the technical NESSoS work packages fits in:

- WP6: Security requirements for services in Requirements
- WP7: Secure service architectures and design in Design and Architecture
- WP8: Programming environments for secure and composable services in Implementation
- WP9: Security assurance for services in Test, validation and assurance
- WP10: Risk and cost aware SDLC in Risk analysis and Cost/benefit analysis

The motivation for the placement of the activity Risk analysis after the activity Requirements is based on the ISO/IEC 27001 which has information security policies and expectations as a prerequisite for the ISMS process while risk analysis is part of the Establish ISMS stage of the process. As it can be argued that security policies is a prerequisite of risk analysis it is also possible to argue that security policies can be an outcome of risk analysis (as a risk treatment option). We will however assume that there exist initial security requirements prior to risk analysis and that the security requirement may be revised either as part of the risk treatment step of risk analysis or in subsequent iterations of the SDLC.

Cost/benefit analysis follows Risk analysis as the outcome of risk analysis is important input to cost/benefit analysis; as is stated in Chapter B, the cost/benefit analysis method provided in this deliverable “can be considered as an extension to the ‘treatment evaluation’ phase of existing risk management frameworks”. Further, Cost/benefit analysis precedes Design, Architecture and the other development activities as the cost/benefit analysis in many case will influence decisions made in these subsequent activities.

The risk and cost aware SDLC is iterative, as illustrated in Figure A.6. It should be noted at this point that the iterations will not be identical even though each iteration has the same generic activities. For
example, *Risk analysis* will typically comprise of high-level risk analysis in the initial iteration, detailed risk analysis in subsequent iterations and maintenance and/or run-time risk analysis in the final iterations. Other activities may have different workload for the different iterations. For example the activity *Implementation* may have little or no workload in the initial iteration but heavy workload in the later iterations. How the various methods for risk and cost analysis relate to the different iterations are further discussed in Section A.7 below.

### A.6 Relating the Overall Process to System Development Methodologies

The generic risk and cost aware SDLC described in the previous section can in some sense be seen as a meta-SDLC from with we can have specific iterative system development methodologies as specialisations. In the following we illustrate this by explaining how the spiral model and the RUP may be seen as specialisations of the generic SDLC.

#### A.6.1 Spiral Model

Related to the Spiral Model, the iterations of the generic risk and cost aware SDLC correspond to the five rounds of the spiral. Since not all activities are present in all rounds of the spiral, some of the iterations of the risk and cost aware SDLC will have “empty” activities. For example, in Round 2 effort will be spent on *Requirements* but the *Design* activity will be “empty”, while in Round 3 there will be effort in *Design* and *Requirements* are empty. In addition the *Risk analysis* activity will encompass both security risks and project risks.

#### A.6.2 Rational Unified Process

Related to the Rational Unified Process (and other Unified Processes) the iterations of the generic risk and cost aware SDLC corresponds to the iterations of the RUP, i.e. Initial, Elaboration 1, Elaboration 2, . . . ,
Construction 1, Construction 2, etc. We should then see Risk & Cost Analysis as one of the cross-cutting activities together with Requirements, Analysis & Design, and so forth. This is illustrated in Figure A.7 which is modified from Figure A.4. As this enhancement of the RUP is currently theoretical, the graph showing the allocation of effort to the Risk & Cost Analysis activity is a rough estimate of what we expect in a “risk and cost aware RUP”. As can be seen we expect the activity to behave somewhat like testing, with some effort and a local maximum for each iteration, and a long tail in the Transition phase. Whether this is in fact the case will have to be validated empirically by application of the modified RUP in case studies.

A.7 Instantiation of the NESSoS Methods in the Overall Process

On the one hand, the aim of WP10 is to incept a risk and cost aware SDLC by adapting and specialising existing iterative SDLCs. On the other hand, WP10 aims at developing and providing methods and techniques to support specific phases or tasks of the overall process of such a risk and cost aware SDLC.

In this section we summarise the methods and techniques presented in Chapter 5 through Chapter 8 and explain how they can be instantiated in the risk and cost assessment of the RC-SDLC.

A.7.1 Method for Balancing Risk and Cost

The method for balancing risk and cost in the SDLC presented in Chapter 5 addresses the needs for a systematic approach to estimating and assessing the cost of identified risks as balanced against the costs and benefits of implementing identified options for risk treatment. In particular, the method targets the final phase of risk treatment in the risk management process of the ISO 31000 standard. This means that the proposed method can be instantiated during the final phases of an iteration of the risk and cost assessment process, within the wider iteration of the overall RC-SDLC.

The approach assumes that the risks have already been identified, documented and assessed, and that treatment options for unacceptable risks have been identified and documented. For this purpose, the
approach assumes that any established risk assessment methodology and risk modelling technique that comply with ISO 31000 can be applied.

Given the documentation of the identified risks and the treatment options, the method for balancing risk and cost uses techniques for identifying, categorising and estimating costs and benefits associated with the risks and treatments within a specified time frame. The identified treatments are assessed so as to select the most advantageous options for the further SDLC process.

In the context of the overall SDLC and the lifespan of the system development, the method for balancing risk and cost moreover identifies specific categories of risk and cost for the various development phases, for example the costs associated with the phases of Inception, Elaboration, Construction and Transition in the Rational Unified Process.

A.7.2 Method for Continuous Assessment of Risk and Cost

The method for continuous assessment of risk and cost presented in Chapter 6 focuses on the continuous assessment of risk at run-time. The method is an approach to support access and usage decisions when there may be uncertainties regarding satisfaction or violation of the relevant policies.

In the context of the ISO 31000 risk management process, the proposed method fits mostly into the risk monitoring of the continuous activity of Monitor and review. However, the method also utilises risk assessment techniques for risk identification, risk estimation and risk evaluation in the sense that the reference monitor makes continuous probability estimates that serve as the basis for risk and cost relevant decision making.

A risk is commonly defined as the combination of the likelihood of an unwanted incident and its consequence when it occurs. In the decision making in access and usage control, the incidents that need to be considered are the granting of access when the policy is violated and the revocation of access when the policy is satisfied. The decision making must moreover take into account the consequence (cost/benefit) of granting access when the policy is satisfied, as well as revoking access when the policy is violated. The likelihood estimation amounts to estimating the probability of policy satisfaction, which yields the (uncertain) likelihood of the relevant incidents and events to occur.

This approach hence uses risk assessment techniques that take cost and benefit into account in order to make the optimal decisions of access and usage during the continuous monitoring at run-time.

A.7.3 Method for Risk and Cost Assessment with Particular Focus on Web applications

Chapter 7 presents an approach to the management of risk and cost in the development life cycle of web applications and web services. The approach focuses in particular on extracting potential attacks based on identified vulnerabilities as well as the capabilities of attackers.

Related to the ISO 31000 standard, the method addresses mostly risk identification and risk estimation and aims at yielding formal risk models that document all sequences of actions that can lead to a given attack.

The method can supplement more traditional risk management frameworks by providing specialised support for mitigating security risks related to web applications and services. A core feature of the envisage approach is the automatic generation of risk models during risk identification that can help designers of security systems to accelerate recurring tasks and to prevent errors and omissions.

A.7.4 Formal Foundation

The formal foundation for a risk and cost aware SDLC presented in Chapter 8 aims at providing a rigorous basis for analysing and reasoning about risk in the risk management process. The formalisation of risk graphs and the calculi to support the reasoning about them provide techniques to support the ISO 31000 phases of risk identification, risk estimation and risk evaluation.

In the wider context of a risk and cost aware SDLC there will be several iterations of the security risk assessment process. While the development life cycle evolves there may be needs to assess how the risk picture will change in the face of, for example, competing design decisions. In such a setting, the support
for modelling and reasoning about changing and evolving risks as presented in Appendix E can be utilised in assessing the risks associated with each alternative.

Modular risk analysis may also serve as a useful technique in the setting of Future Internet services and systems as these are inherently heterogeneous and modular. The support for modelling and reasoning about dependencies in risk models presented in Chapter 8 may serve as useful techniques in this setting.

The proposed formal approach currently addresses the identification of threat scenarios and unwanted incidents, as well as the estimation and reasoning about their likelihoods. A direction for future work is to extend the formal foundation to provide explicit support for taking cost into account, and furthermore to explore risk graphs and selected instantiations of these as techniques (such as the Markov chains presented in Appendix C or the CORAS diagrams presented in Appendix E) for risk and cost assessment in the SDLC.

A.8 Conclusions

In this paper we have presented a first attempt toward a risk and cost aware SDLC. The approach assumes an iterative SDLC into which a process for security risk and security cost management is integrated. As a result, risk and cost analysis serves as one of the iterative activities of the SDLC together with the other activities such as requirement engineering, design, implementation and testing.

In the development life cycle of engineering secure software services and systems the aim is to address security concerns from the very beginning, thereby reducing vulnerabilities and maintaining security needs through the engineering process. Embedding security risk and security cost management into the SDLC contributes to make these concerns drivers of the development. The RC-SDLC moreover aims to ensure the value of security solutions and their return of investment.

The proposed RC-SDLC process makes use of various methods and techniques for conducting the various phases and tasks. In this paper we have explained how the methods for a risk and cost aware SDLC proposed in the NESSoS context can be instantiated in the overall process to fulfil specific analysis needs.

References


B  Method for Balancing Risk and Costs in the SDLC

Stefan Eicker  Annett Nagel  Karsten Radatz

Abstract: Current incidents according to data losses, security breaches etc. can be found very often nowadays (e.g. Sony Playstation Network, MasterCard). So risk management is still an important aspect that needs to be taken in account. There are lots of risk management frameworks that provide methods to assess risks and help identifying treatments for unacceptable risks. But it is not only about the loss of security, but also about the loss of money; so this topic not only interests the IT department but also the business department. But at this point the risk management frameworks do have a problem: They do not satisfy the needs of decision makers as they lack elaborate methods, techniques and tools for assessing and weighting treatment costs and the resulting benefits.

In this paper, we propose a method, which can be considered as an extension of the treatment evaluation conducted during the risk treatment phase of existing risk management frameworks. The method therefore expects results from previous steps such as risk models, risk estimations and options for risk treatment. The results of the method provide an improved foundation for further decisions in comparison to the results of a cost/benefit analysis. It has been designed with cost-efficiency in mind and is divided into three different steps.

Keywords: Risk management, ISO 31000, costs, benefits, total list of types of costs, total list of types of benefits.

B.1 Introduction

Existing risk management frameworks, such as OCTAVE [2], CRAMM [4], the NIST Risk Management Guide for Information Technology Systems [19] and CORAS [16], provide methods to assess risks and identify treatments for unacceptable risks, but lack elaborate methods, techniques and tools to assess and weight treatment costs and the resulting benefits. The ISO 31000 risk management standard [12] mentions “balancing the costs and efforts of implementation against the benefits derived”, but gives no advice how this could be achieved. (From ISO 31000, Section 5.5.2) This lack of guidance motivates the development of a method for balancing risk and cost. Since the assessment of costs and benefits in this context is a complex endeavour, it cannot satisfy the needs of decision makers by providing the results of a cost/benefit analysis only. Instead, we argue in favour of a systematic approach, which explicitly incorporates the experience of experts in an iterative way.

In this paper, we propose such a method, which can be considered as an extension of the treatment evaluation conducted during the risk treatment phase of existing risk management frameworks. (See Figure 4.1.) The method therefore expects results from previous steps such as risk models, risk estimations and options for risk treatment. The results of the method provide an improved foundation for further decisions in comparison to the results of a cost/benefit analysis. It has been designed with cost-efficiency in mind and is divided into three different steps.

The paper is structured as follows. In section B.2, the relation and interfaces to existing risk analysis methods is considered, taking the CORAS approach as a concrete example. Section B.3 introduces our concept of a “total list of types of costs” and a “total list of types of benefits” which is the basis for our method. The method itself is described in Section B.4. The paper finishes with Section B.5 which gives a short conclusion and an outlook for further work.

B.2 Integration with Existing Risk Analysis Methods

The method has been designed to be embedded into the risk management process of the ISO 31000 standard, and extends the 5th phase “Treat Risk”. This ensures generality and allows the end-user to
apply the method in the setting of any risk management framework that operationalises the ISO 31000 process. The CORAS model-driven risk analysis method called is used in this paper to give a concrete example how it can be used by existing ISO 31000-based risk analysis methods. CORAS is a risk analysis method which is divided into eight main steps. These are:

1. Preparations for the analysis
2. Customer presentation of the target
3. Refining the target using asset diagrams
4. Approval of the target description
5. Risk identification using threat diagrams
6. Risk estimation using threat diagrams
7. Risk evaluation using risk diagrams
8. Risk treatment using treatment diagrams

The first four steps correspond to the context establishment illustrated in Figure 4.1, whereas Step 5 to Step 8 correspond to the steps following the context establishment.

In order to conduct the eight steps of the CORAS process, CORAS provides a customised language for modelling and assessing risks and threats. The CORAS diagrams are inspired by UML and CORAS also offers a computerised tool for documentation, maintenance and reporting.

The method for balancing risk and cost is presented in details in Section B.4, but an overview of how it is embedded into the ISO 31000 process is illustrated in Figure B.1.

![Diagram](image)

**Figure B.1: How the proposed method fits into the ISO 31000 standard**

In the following sections an overview of the proposed method is given and demonstrated along the running example also used in conjunction with the CORAS approach [5]. The overview does not only illustrate
the implementation of our method but also demonstrates by example how risk analysis approaches can be extended by the new proposed method. Throughout this contribution italic font is used for the running example. The basic setting of the case study is described as follows.

In one region of the country, an experimental telemedicine system has been set up. A dedicated network between the regional hospital and several primary healthcare centres (PHCCs) allows a general practitioner (GP) to conduct a cardiological examination of a patient (at the PHCC) in co-operation with a cardiologist located at the hospital. During an examination both medical doctors have access to the patient’s health record and all data from the examination is streamed to the cardiologist’s computer.

The National Ministry of Health is concerned whether the patient privacy is sufficiently protected, and hires a security analysis consultancy company to do a security analysis of the cardiology system. The consultancy company appoints a security analysis leader and a security analysis secretary to do the job (referred to as “the analysts”). In co-operation with the ministry, they organise the meetings and workshops of which the security analysis is comprised.

B.3 Total Lists of Types of Costs and Benefits

The total list of types of costs and the total list of types of benefits are the foundation of our method. They contain domain knowledge regarding costs and benefits specified for the development, introduction and use of software and benefits specific for the development, introduction and use of software and will be used as guidance to build the project specific customized lists. With this concept of total lists we aim at putting all elements of possible types of costs and benefits into one list in order to derive concrete company, project and treatment specific lists from this. The total lists are derived from fields such as IT service management and business studies taking into account theory as well as practice. Hence, for the total list of types of costs e.g. several cost plans from literature were used; the total list of types of benefits was derived from risk lists. In the following subsections the development of the lists will be described in more detail.

B.3.1 Basic Principles

In order to get a consistent and reliable total list of types of costs at first basic principles have to be defined how to build this list. From literature four principles can be identified [9]:

Cleanliness and clearness. This principle means that costs can only be assigned to one type of costs. One example which violates this principle is personnel costs for reparation which can be assigned to personnel costs or to reparation costs. Another type of costs which violates this principle is typically the category miscellaneous.

Consistency and no overlapping. This means that allocation of costs based on the available documents can be done consistently and without delay. Also the same costs always have to be added to the same type of costs for each accounting period.

Completeness. This principle means that all costs which fulfil the properties of a chosen type of cost must be represented in the total list of types of costs.

Cost effectiveness. This principle is about having costs in mind when enforcing the other three principles.

The first two principles are both aimed at providing a clean list of types of costs. In addition, Haberstock states that the cleanliness and consistency principles work against cost effectiveness because a higher level of cleanliness and consistency will always result in higher costs [9]. Therefore Rüth tolerates a mixed type of costs in the case that its value does not matter [18]. Zilahi-Szabo’s demand for a minimum value for a separate cost category is also introduced because of cost effectiveness [21]. Cost effectiveness can therefore be seen as an encompassing principle. Hence, Rüth does not see cost effectiveness as a separate principle itself [18].
B.3.2 Total List of Types of Costs

In this sub-section the total list of types of costs will be derived from combining the business economics view, IT cost lists and the IT management view. At last the resulting total list of types of costs will be put into combination with the software development phases in order to get an SDLC-specific overview of the costs.

Business Economics View

From the business economics view different criteria to group the total list of types of costs can be identified. These criteria are used to build cost lists from the IT management view. Elsässer states that all IT costs have to be measured, classified and allocated regarding business economics principles (basic principles) [6]. Hence, this perspective is also relevant for IT costs. Table B.1 shows the identified criteria which can be used to build and structure the total list of types of costs.

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Criteria</th>
<th>Value</th>
</tr>
</thead>
</table>
| 1   | Type of Production Factor | - Cost of Materials   
|     |                            | - Personnel Costs         
|     |                            | - Asset Costs             
|     |                            | - Service Costs           
|     |                            | - Capital Charges         
|     |                            | - etc.                   |
| 2   | Type of Goods             | - Expandable Goods        
|     |                            | - Durable Goods           |
| 3   | Accountability            | - Direct Costs            
|     |                            | - Indirect Costs          |
| 4   | Type of Cost              | - Variable Costs          
|     |                            | - Fixed Costs             |
| 5   | Origin of Costs           | - Primary Costs           
|     |                            | - Secondary Costs         |
| 6   | Type of Cost Accounting   | - Imputed Costs           
|     |                            | - Basic Costs             |
| 7   | Function                  | - Procurement Costs       
|     |                            | - Manufacturing Costs     
|     |                            | - Distribution Costs      
|     |                            | - Administrative Costs    
|     |                            | - etc.                   |

Table B.1: Common Criteria

Haberstock concludes that a classification is usually done by the type of the production factor. On a lower level another criterion can be used to structure the list [9]. Following Hummel and Männel [11], a practicable classification for types of costs uses different criteria. This means that the structure of the resulting total list of types of costs is a combination of the seven criteria from Table B.1.

IT Cost Lists and IT Management View

Various cost lists can be found in literature, e.g. IT cost lists from Elsener [7] shown in Table B.2, or Gadatsch and Mayer[8] shown in Table B.3. The total list of types of costs proposed in this method is among others (such as the cost plan according to Jaspersen [13]) built upon these cost lists.

Besides these classifications, ITIL (published by the Office of Government Commerce (OGC)) offers a classification of costs which is part of the Financial Management for IT Services. [17] This classification is shown in table B.4. ITIL uses three criteria for grouping costs. Two of them have already been mentioned
### Personnel
- Salary
- Overtime & Stand-by Duty
- Fringe Benefits
- Allowable Expenses
- Training
- External Personnel

### Hardware
- Purchase of Hardware
- Amortization of Hardware
- Leasing of Hardware
- Hardware Maintenance

### Communication
- Phone / Fax
- Carrier

### External Services
- Outsourcing Services
- Consulting Services

### Software
- Purchase of Software
- Amortization of Software
- Leasing of Software
- Software Maintenance

### Miscellaneous
- Office Equipment
- Literature
- Occupancy Costs
- Overhead Costs

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Main Type of Cost / Type of Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IT Material</td>
</tr>
<tr>
<td>1.1</td>
<td>Paper, Ink, Toner</td>
</tr>
<tr>
<td>1.2</td>
<td>Media (Disks, CDs, Tapes, HDDs)</td>
</tr>
<tr>
<td>1.3</td>
<td>…</td>
</tr>
<tr>
<td>2</td>
<td>IT Development</td>
</tr>
<tr>
<td>2.1</td>
<td>Software Development, Customizing of Standard Software</td>
</tr>
<tr>
<td>2.2</td>
<td>Software Support</td>
</tr>
<tr>
<td>2.3</td>
<td>…</td>
</tr>
<tr>
<td>3</td>
<td>IT Operation</td>
</tr>
<tr>
<td>3.1</td>
<td>Hosting, Backup</td>
</tr>
<tr>
<td>3.2</td>
<td>Network Services (Firewall, Antivirus, etc.)</td>
</tr>
<tr>
<td>3.3</td>
<td>…</td>
</tr>
<tr>
<td>4</td>
<td>Amortization, Rent, Leasing, Imputed Costs</td>
</tr>
<tr>
<td>4.1</td>
<td>Amortization Hardware</td>
</tr>
<tr>
<td>4.2</td>
<td>Amortization Software</td>
</tr>
<tr>
<td>4.3</td>
<td>…</td>
</tr>
</tbody>
</table>

Table B.2: Types of Costs, according to Elsener [7]

Table B.3: Types of Costs, shortened illustration according to Gadatsch and Mayer [8]
in table B.1, namely “accountability” and “type of cost”. These are completed by “operating costs and investment costs”.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Accommodation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUs</td>
<td>Offices</td>
</tr>
<tr>
<td>LANs</td>
<td>Storage</td>
</tr>
<tr>
<td>Disk storage</td>
<td>Secure areas</td>
</tr>
<tr>
<td>Peripherals</td>
<td>Utilities</td>
</tr>
<tr>
<td>WANs</td>
<td></td>
</tr>
<tr>
<td>PCs</td>
<td></td>
</tr>
<tr>
<td>Portables</td>
<td></td>
</tr>
<tr>
<td>Local Servers</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software</th>
<th>External Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating systems</td>
<td>Security services</td>
</tr>
<tr>
<td>Scheduling tools</td>
<td>Disaster recovery services</td>
</tr>
<tr>
<td>Applications</td>
<td>Outsourcing services</td>
</tr>
<tr>
<td>Databases</td>
<td>Human resources overhead</td>
</tr>
<tr>
<td>Personal productivity tools</td>
<td></td>
</tr>
<tr>
<td>Monitoring tools</td>
<td></td>
</tr>
<tr>
<td>Analysis packages</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>People</th>
<th>Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payroll costs</td>
<td>Internal charges from other cost centers within the organization</td>
</tr>
<tr>
<td>Benefit cars</td>
<td></td>
</tr>
<tr>
<td>Re-location costs</td>
<td></td>
</tr>
<tr>
<td>Expenses</td>
<td></td>
</tr>
<tr>
<td>Overtime</td>
<td></td>
</tr>
<tr>
<td>Consultancy</td>
<td></td>
</tr>
</tbody>
</table>

Table B.4: Types of Costs according to ITIL

SDLC-specific View

Figure B.2 shows an exemplary model of a total list of types of costs, which is derived from the mentioned literature but not yet SDLC-specific. In order to get a complete overview of the costs of a software project Henrich proposes to use the categories from IT cost lists in combination with phases from the software development process. At last Henrich states that it should be differentiated between one-time costs and running costs [10]. For the SDLC-specific view in Table B.5 development phases according to the Rational Unified Process (RUP) [14] are used. See Chapter 4 and Figure 4.3 for a brief overview of RUP.

B.3.3 Total List of Types of Benefits

The total list of types of benefits has been derived from risk management processes and from literature dealing with IT security and IT project management.

German Federal Office for Information Security

The German federal office for information security (Bundesamt für Sicherheit in der Informationstechnik (BSI)) identifies two main types of benefits of risk treatments as illustrated in Figure B.3, namely qualitative benefits and quantitative benefits. Quantitative benefits are financial benefits which can directly be measured, e.g. savings from higher productivity by using state of the art technology rather than out of date products. Qualitative benefits are grouped into the three different categories of availability, integrity and confidentiality. Figure B.2 gives an overview of the identified types of benefits. The BSI states that
Figure B.2: Total List of Types of Costs

Table B.5: Example of SDLC-specific total List of Types of Costs
increasing availability, integrity and confidentiality will primarily reduce the costs of incidents with a high probability [20].

Further on there are some benefits which will also result from treating risks like image improvement, improvement of cost effectiveness, minimisation of unwanted incidents, minimisation of loss of knowledge and improvement of system stability [20].

![Types of Benefits according to BSI](image)

**Figure B.3: Types of Benefits according to BSI [20]**

**IT Project Management View**

From IT project management literature we can also identify several types of benefits from treating risks. Aichele [1] identifies three categories of benefits, direct benefit, indirect benefit and elusory benefits. Direct benefit is defined as saving of current costs; this means e.g. saving of material costs or personnel costs. Indirect benefit is defined as saving of upcoming costs like e.g. reduce of bad debts. Elusory benefit is defined as a qualitative factor like, for example, enhanced data security.

The total list of types of benefits shown in Figure B.4 has been derived from these different views.

**B.4 Method Description**

Figure B.5 depicts an overview of the steps of the method which will be described in more detail in the following sub-sections.

**B.4.1 Step One – Customising**

This step is organised as a workshop with experts from the department of the enterprise and/or external consultants. As a prerequisite it has to be explicitly determined by the experts of the enterprise for which timeframe the costs and benefits have to be evaluated. After that the relevant costs and benefits for the enterprise will be identified. Hence, the goal of this workshop is to prepare a customised cost and treatment list in regard to the timeframe given by the risk horizon. The required input for phase one is the total cost list (e.g. hardware, software, human resources, etc.) and a list of possible types of benefits (e.g. image, finance, etc.) given by the proposed method and in addition the knowledge of the company’s experts.
The total list of types of benefits as well as the total list of types of costs is customisable. The enterprise value for instance might be of interest for a stock corporation but not for a private limited company.

The health ministry in our example has identified that the costs for operating systems are not relevant as only free available open source operating systems are in use. Therefore this point is deleted from the total cost list as shown in Table B.6. Besides this point some others are also deleted from the list because they are not relevant for the ministry. The reason why has to be documented in the new customised list in order to make this decision traceable. In addition the total list of types of benefits is shortened because the benefit of preserving the enterprise’s value is not of interest for the ministry. This decision has also to be documented in the new customised list of types of benefits.

As a result of this workshop there are two individually customised lists which fit the company's needs in regard to the given project. In our example the ministry has setup a customised total list of types of costs for an extended user training program in order to avoid corrupted patient records. Because the training program takes place during the transition phase only costs for this phase are taken into account.

Step One – Summary
Input:
- Total List of types of costs
- Total List of types of benefits

Tasks:
- Determine relevant timeframe (Step 1a)
- Customise total list of types of costs (Step 1b)
Step 1: Customizing

- Step 1a: Determine relevant Timeframe
- Step 1b: Customize Total Cost List
- Step 1c: Customize Total List of Benefits

Figure B.5: Three Steps

- Customise total list of types of benefits (Step 1c)

Attendees (external consultants as well as internal employees):

- Analysis leader
- Analysis secretary
- Representatives of the client:
  - Decision makers
  - Finance
  - Project Manager (from the software development project)

Diagram:

- The changes have to be documented traceably structured as a new costs and benefits list.

Output:

- Customised total list of types of costs
- Customised total list of types of benefits
- Timeframe

B.4.2 Step Two – Estimating

Step two is also organised as a workshop with representatives from the client. It requires the customised lists developed during phase one as well as risk treatments, risk assessments and the risks themselves - all given by the used risk analysis method (e.g. CORAS).

During the workshop the costs of the risks will be estimated on the one hand and on the other hand the treatment costs for the risks will be appraised. The estimation process is designed as an iterative process which will be repeated until the experts (e.g. project managers) come to a common conclusion. For structuring the process any useful method such as a Delphi study [15] or planning poker known from Scrum [3] can be used.

The output of this step is the quantified benefit as a delta between costs for untreated and treated risks and the quantified costs of the treatments where the costs are estimated in regard to the given timeframe from step one.
### Table B.6: Example of a SDLC-specific total list of types of costs for the extended user training program

<table>
<thead>
<tr>
<th>Types of Costs</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personnel</td>
<td>One-Time</td>
<td>-</td>
<td>-</td>
<td>Trouble-shooting, Administration, Maintenance, Training, Installation, Testing, Configuration</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Material</td>
<td>One-Time</td>
<td>-</td>
<td>-</td>
<td>OS, Apps, DBs, PCs, Notebooks, Server, Network, Spare parts, Mass storage, Peripherals, Data media, Toner/Ink, Literature</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>External Services</td>
<td>One-Time</td>
<td>-</td>
<td>-</td>
<td>Room usage, Room setup, Building setup, Energy, Capital costs</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Overhead</td>
<td>One-Time</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

### Table B.7: SDLC-specific total list of types of costs with quantified costs

<table>
<thead>
<tr>
<th>Types of Costs</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personnel</td>
<td>One-Time</td>
<td>-</td>
<td>-</td>
<td>Training (800€), Installation (100€), Configuration (100€)</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Material</td>
<td>One-Time</td>
<td>-</td>
<td>-</td>
<td>Literature (50€)</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>External Services</td>
<td>One-Time</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Overhead</td>
<td>One-Time</td>
<td>-</td>
<td>-</td>
<td>Room usage (50€), Room setup (50€), Energy (50€)</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
The experts of the health ministry now estimate costs and benefit of different treatments in this workshop in an iterative way from past experiences. They come to the conclusion that an extended training program two days will cost 1200€. The benefit of this treatment will be that recovery costs will be halved because the risk of compromised health records will be reduced by 50%. This can be assumed because former incidents have shown that about half of the incidents are due to a lack of training and all of them could have been avoided with sufficient training. That means the initial frequency of once per year will drop to 0.5 per year. This also means that the frequency of this incident will drop to “unlikely”. Having a look at the risk evaluation matrix an unlikely incident with moderate consequences is “acceptable” for the company.

From past experiences the experts know that restoring a compromised health record from the backup will cost 400€ per item. For estimating the corresponding costs the experts used a method called planning poker which they already knew from other software development methods like Scrum.

Step Two – Summary

Input:
- Output step one
- Risk treatments
- Risk assessments
- Risks

Tasks:
- Estimating costs
- Estimating benefits

Attendees:
- Analysis leader
- Analysis secretary
- Representatives of the client:
  - Decision makers
  - Project Managers

Finance is not part of the experts group as it would not be effective to verify the own results in the next step.

Diagram:
- Table of the costs and benefits

Output:
- Table with quantified costs and benefits

B.4.3 Step Three – Verifying

As well as the first two phases phase three is also organised as a workshop. The needed input is provided by step two. In this phase the estimations made in step two will be verified by other experts of the company. So the output of the method will be a verified cost and benefit analysis which can then be used for further decision support methods like for instance ROI, ROSI or ROISI to evaluate the outcome with respect to return on investment in security.
The input from step two is a list with the benefit of an extended training program and the costs of restoring a compromised health record. The finance department which is part of step three does not agree with the experts involved in step two that restoring a single health record costs 400€. Having a look at the invoices from past incidents the finance department comes to the conclusion that restoring a health record results in costs of 1200€ in average. The project managers have just estimated the costs for external service providers (restoring backup) but missed the internal costs for human resources (providing backup tapes, troubleshooting, health record not available etc.).

<table>
<thead>
<tr>
<th>Types of Costs</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personnel</td>
<td>One-Time</td>
<td></td>
<td></td>
<td>Training (200€), Installation (50€), Configuration (50€)</td>
</tr>
<tr>
<td>Material</td>
<td>One-Time</td>
<td></td>
<td></td>
<td>Literature (50€)</td>
</tr>
<tr>
<td>Overhead</td>
<td>One-Time</td>
<td></td>
<td></td>
<td>Room usage, Room setup, Energy (50€)</td>
</tr>
<tr>
<td>Sum</td>
<td>Total</td>
<td></td>
<td></td>
<td>400€</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Types of Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finance</td>
</tr>
<tr>
<td>Reduce of recovery costs (1200€ * 50 / 100 = 600€)</td>
</tr>
<tr>
<td>Sum</td>
</tr>
</tbody>
</table>

Table B.8: Verified total list of costs and benefits

Step Three – Summary

Input:
- Output step two

Tasks:
- Verify with finance experts the output of step two

Attendees:
- Analysis leader
- Analysis secretary
- Representatives of the client:
  - Decision makers
  - Finance
  - Project Managers
B.5 Conclusion and Future Work

In this paper we have elaborated a new method of how to measure costs and benefits of risk treatments. The new proposed method can be embedded into any ISO 31000 based risk management methods and extends the last phase treat risks. Furthermore the method does not limit the user to a given decision model like ROL or ROISI, but gives the opportunity to freely choose a decision support method of preference.

Future work has to be done on the evaluation of the total list of types of cost and the total list of types of benefit. It has to be determined whether the both lists are really a complete repository for types of costs and types of benefits or if they have to be expanded.

A tool supporting the workflow of the different steps should also be made available. This tool could e.g. take into account dependencies between different treatments. For our example a plausibility check which prevents the users from entering false data could make the second day of the training program obsolete.

References


Abstract: Usage Control (UCON) model is a successor of access control model. The main difference between the models is that UCON checks the access policy not only before the access is granted, but also during the access. Usage sessions may last for a long time and the attributes, required for the access decision, may change. Similar to the access control model, the correct access control decision in UCON can be made only if the required attributes are up-to-date. Monitoring of every attribute though desirable is often impossible or impractical. Therefore, fresh values of some attributes can be available only in some points of time, while the decisions about continuing or revoking access must be performed at any time.

In this paper we propose an approach which helps to make a reasonable decision even if there are some uncertainties about the not fresh value of an attribute. The approach exploits Markov chains to predict the current value of attributes and uses risk assessment to weight possible decisions (to continue or to revoke access). We provide two versions of the approach which are applicable depending on the available information. We implemented our model in Java and checked its performance and accuracy of prediction.

Keywords: Usage control model; risk assessment; freshness of attributes; Markov chains.

C.1 Introduction

Access control is based on the assumption that the required attributes do not change during the access. This assumption is valid if usage sessions are short with respect to the frequency of changes of attributes. Wider spread of new technologies (such as Web Services, Grid, and Cloud computing) made usage sessions longer. With longer sessions we cannot rely on the assumption that attributes do not change during the session. This problem has been solved with a new paradigm – usage control model (UCON) introduced by Sandhu and Park [9, 7, 8]. The most important features of the UCON model are mutability of attributes and continuity of control. Mutability of attributes means that attributes required for making an access decision can change during a usage session. Therefore, it is not enough to evaluate attributes once before the access is granted, but continuous evaluation is required when the usage session is active.

Unfortunately, continuous usage control is not always feasible, because of a number of uncertainties which exist in the system. Some uncertainties are unintentional and generated by natural causes [2]. These uncertainties cause a difference between a real attribute value and a value used for making the decision. First, there are inevitable delays of information delivery in any distributed system. Second, some attributes cannot be checked frequently because it is either impossible, or impractical, or too costly. Thus, there is a trade of between the cost of an attribute check and a possible cost if attribute is not checked. Third, there are the attributes which require considerable time to be processed before they can be sent to the decision making engine. Hence, frequently the decision about usage session must be made when the precise values of attributes are not available.

We exploit risk for making a access and usage decisions when some uncertainties do exist. The idea is to assess possible losses and benefits of taking a decision. In most cases risk is used to judge if a system is secure enough or some additional controls should be installed. Uncertainties in these cases are expressed as average probability that a threat occurs. The well-known formula is used for computation of risk [3, 10, 1] when the probability of an unwanted event is multiplied by the impact of the unwanted event. The event is that the attribute value changed and an access should be revoked according to that value.
We propose several models for cost effective enforcement of usage control policies [4, 5]. The main results of our work are

- methods for computation of the probability of a policy failure exploiting discrete-time and continuous-time Markov chains;
- a method for decision making based on comparison of possible risks and benefits of continuing and revoking the usage session;
- experimental justification of accuracy of prediction for a location-based policy with one attribute and one threshold;

C.2 Intentional and Unintentional Uncertainties

Let \( H_t \) be an event specifying that a real attribute value \((a_{\text{real}}, t)\) does satisfy a security policy, while \( \overline{H}_t \) specifies the opposite. The reference monitor operates only with observed attributes and can compute \( H_{t}^{\text{obs}} \), i.e. the policy holds for observed attributes at a given time \( t \).

Usually, the reference monitor possesses uncertain knowledge about real attribute values. Assume, the reference monitor can measure this uncertainty by computing the conditional probability \( \Pr[H_t | H_{t}^{\text{obs}}] \) that the policy really holds at \( t \) knowing that observed attributes satisfy the policy at \( t \). Let \( H_{[t_{\text{perm}}:t_{\text{now}}]} \) specifies that the sequence of real attribute values satisfies the policy \( p_{\text{on}} \) starting from \( t_{\text{perm}} \) till \( t_{\text{now}} \).

In usage control the policy is evaluated every time when attribute changes. In case of the attribute pulling some values might be missed. Thus, the reference monitor has less information to prove that \( H_{[t_{\text{perm}}:t_{\text{now}}]} \) holds. Assume, that knowledge of the reference monitor about the predicate satisfaction in this interval is probabilistic and

\[
\Pr[H_{[t_{\text{perm}}:t_{\text{now}}]} | H_{t_{a1}}^{\text{obs}} \cdot H_{t_{a2}}^{\text{obs}} \cdot \ldots \cdot H_{t_{ak}}^{\text{obs}}]
\]

specifies the probability that the policy really holds by \( t_{\text{now}} \) knowing that observed attributes satisfy the policy at time of issuing.

Freshness of Attributes is unintentional uncertainty occurring due to attributes mutability. We launched three types of freshness uncertainties.

Freshness I. Corresponds to the scenarios where only a part of attribute changes can be detected. As an example, assume the network of sensors provides the current location of the user. Sensors have limited resources (power, bandwidth, memory), and the reference monitor pulls the location attribute only once per hour. Even if the attribute does not satisfy the policy during this hour, the reference monitor will make the incorrect access decision and continue the access. There always exists a possibility of the policy violation in-between despite that all pulled attributes satisfy the policy.

Freshness II. Implies that an attribute may change during inevitable time delays \( \Delta t_{\text{proc}} = \tilde{t} - t > 0 \) needed for the delivery (due to a network latency) and decision making (evaluation of logical predicates).

Freshness III. corresponds to scenarios where the current attribute value is uncertain since some update queries are pending and may not be committed by the time of the policy re-evaluation.

As an example, assume a policy which allows users with a "normal" reputation to submit a huge number of applications for execution in Grid environment. The reputation is updated only when the execution is ended and the system receives feedback from a resource provider. Applications can run concurrently and each single execution can be long-lived and lasts days. The access decision to submit a new job is based on the reputation value dated by the last registered feedback and on the number of applications currently running on the user's behalf. Indeed, the ongoing applications can be malicious but this fact can be discovered afterwards. The only way to obtain the fresh reputation value is to block the access until all running applications terminate. Instead, the system has to be set up to make an access decision with some uncertainty on the current reputation of the user.
Correctness of Attributes. Is affected by additive noises that usually exist in case of non-accurate measurements. For example, the location attribute can be sensed only with the given precision.

Trustworthiness of Attributes. Appears as a result of altering attributes by the attribute provider or as the result of attacks occurred during attributes delivery, storing, etc. Current approaches guarantee only integrity of an attribute by validating a signature of the entity which signs the attribute, but this does not guarantee trustworthiness.

C.3 Cost-effective Policy Enforcement Model

The reference monitor chooses between two alternatives (grant access and deny/revoke access) only one, which is as good as possible. Good means that the reference monitor grants access to legitimate users and the policy holds, and forbids the access to unauthorised entities otherwise. In the presence of uncertain attributes, the reference monitor is unable to infer accurately whether the policy holds, and, consequently, to choose a good alternative. There are four scenarios how the reference monitor acts processing uncertain attributes:

- **true positive**: continue access and the policy is satisfied;
- **false negative**: continue access and the policy is failed;
- **false positive**: revoke access and the policy is satisfied;
- **true negative**: revoke access and the policy is failed.

True positive and true negative are correctly selected alternatives, while false negative and false positive are erroneous. Each scenario has a monetary outcome, cost, the reference monitor loses/gains choosing this scenario. These numbers could be represented as a table (see Table C.1).

<table>
<thead>
<tr>
<th>Decision</th>
<th>Satisfied policy</th>
<th>Failed policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continue access</td>
<td>$C_{TP}$</td>
<td>$C_{TN}$</td>
</tr>
<tr>
<td>Revoke access</td>
<td>$C_{FP}$</td>
<td>$C_{FN}$</td>
</tr>
</tbody>
</table>

Table C.1: Decision matrix

The probability that the received data is correct could be found from the precision of the measuring method and tools. There are also a number of work on determining the trustworthiness of a data provider (see the paper on reputation [6]). Thus, we assume that this value also could be found. Therefore, we pay more attention to finding the probability of fresh data, i.e., the probability that no changes occurred since the last check we were able to observe.

We propose to use Markov Chains in order to find this probability. The available information about a change of the attribute allows using one of the following two models:

![Markov Chain](image)

1Here we assume that the three compounds are independent.
1. We model an attribute using a discrete-time Markov chain (DTMC) if we know the number of attribute changes occurred since the last check.

2. We use a continuous-time Markov chain (CTMC) to model an attribute when we do not know the number of changes but know the time passed from the since check.

**Example C.1** Consider an auction where the policy allows selling goods if the rating of a seller is greater than 1. Currently, we know that a seller has rating 3, but this information is old, because two more deals have been accomplished since the last update of the rating. The customers have not uploaded the ratings of the sellers simply because the customers have not got their goods yet (delivery takes some time). The auction has to know if the seller should be able to sell goods or, maybe, his activity should be suspended. The Discrete-Time Markov chain for the example is shown in Figure C.1. The nodes of the chain denote the values of the rating, and the edges show the possible changes of the rating. For a particular seller we can assume that transition probabilities $\mu$, $\eta$, and $\omega$ are the same for all states. The grey circles denote the states where the policy is violated (bad states).

**Example C.2** As another example, we consider a location of a user in a building as an example of the attribute. The building contains six rooms: two laboratories (Lab1 (1) and Lab2 (3)), the office (2), the corridor (4), the bar (5), and WC (6). The security policy allows the access to the resource from rooms 1 and 2, and forbids the access from other rooms. The Markov chain representing the building contains six states (Figure C.2a). The nodes represent the rooms, and the edges represent possible transitions (doors) between rooms. The modified Markov chain is presented in Figure C.2b. Since states 3, 4, 5, and 6 are forbidden, they are replaced with a single absorbing state $a$. The following one-step transition probabilities ($P$) and rates ($V$) have been determined according to the experimental observations.

Naturally, in order to apply Markov chain theory we need to identify the required transition and/or rate parameters. On the other hand, these values usually do not depend on time and could be computed in advance. Probably, additional resources could be needed to find these values, but such analysis has to be done only once during a short period.
Now we are able to define the probability that no changes occurred since the last check we were able to observe. Here we skip the mathematical details.

When we know the probability of policy violation $\Pr_{RM}$ according to the ALE analysis the possible benefit of allowing further access is $(1 - \Pr_{RM}) \times C^{tp}$. On the other hand, allowing access we also suffer some losses: $\Pr_{RM} \times C^{tn}$. The same logic can be applied to another alternative (to revoke access). Thus, the access should be allowed if:

$$(1 - \Pr_{RM}) \times C^{tp} + \Pr_{RM} \times C^{tn} > \Pr_{RM} \times C^{fn} + (1 - \Pr_{RM}) \times C^{fp}$$  \hspace{1cm} (C.1)

C.4 Risk of Violation of Policy of Several Attributes

Frequently, a policy consists of a number of complex usage rules. A complex rule contains several atomic rules, that constrain different attributes of a subject, an object, and an environment. In our paper we consider only the following three operators for aggregation of rules: conjunction (AND), disjunction (OR), and negation (NOT).

We assume that the attributes are statistically independent. This property can be guaranteed by the policy designer that should choose attributes in a proper way. There are two possibilities for assigning costs to a complex rule.

1. The first one is when four costs for the decision matrix (see Table C.1) are assigned for the whole complex policy. This situation is applicable if the costs do not depend on the cause of policy failure. Thus, it does not matter which atomic rule fails, because we suffer the same amount of losses. This situation is easy for policy-makers, because only 4 costs are required for computations. The risk-aware decision about a usage session for the complex rule is done in the same way as for a policy of an atomic rule. The only difference is that probabilities have to be computed using the probability theory.

2. The second possibility is applicable when a more fine-grained analysis is required. In such case we need to distinguish between losses caused by a failure of one attribute or another one. Such situation usually happens when satisfaction of one rule is much more important for us than the satisfaction of another one. In this case we aggregate possible losses and benefits.

C.5 Experiment

As a proof of concept we implemented our approach as a software prototype in Java. In this experiment we considered only the uncertainties of Freshness I. During the experiment we analysed how a researcher moves around the scientific institute. We gathered the statistical information about the changes of location of a person. The structure of the institute is shown in Figure C.2a in Section C.3. We collected 373 states visited by a researcher and times he spent in these states. We used first 250 to compute the transition rates for CTMC. Then we used other 123 to perform the test of the model.

First, we found the average time that a person spends in each rooms and where it heads after leaving this room. These parameters were used in order to determine the required parameters for the Continuous-Time Markov Chain. We used other 123 to perform the test of the model.

Second, we tested accuracy of prediction and performance of our model.

To test the accuracy of prediction we generated a random moment for the check, and if the location at this moment was Lab1, we inspected if the researcher entered the forbidden states during the interval until the next check. Then we compared this practical frequency of violation with theoretical probability computed using our model. The results of the experiment are presented in Figure C.3a. In the figure theoretical results are shown as a solid line while the practically found values are depicted as squares. We evaluated the ratio error of practical frequency. The deviation of practical and theoretical data is less than 5%. In addition, we measured the accuracy of prediction of probability that policy is violated at the current moment. Again, we randomly generated the moment for the start of the session, and then computed the probability of policy violation at the end of interval. The results of the experiment are shown in Figure C.3b. The ratio error does not exceed 10.
We analysed the performance of the model computing the time required for making 1000 decisions in case of different number of states in the chain (Figure C.4). We have found that the time of execution grows polynomially with number of states in the Markov chain ($n^3$). Note, that the existing tool has not been built with performance requirements in mind. Therefore, we think that performance could be significantly improved by more advanced programming. On the other hand, even now we can see that the tool works very well for 20 states. Moreover, even higher number of states could be used if several seconds for making a decision is acceptable time.

![Figure C.3: Accuracy of prediction](image)

![Figure C.4: The time required for making 1000 decision in case of CTMCs of different size](image)

### C.6 Conclusion

In this work we presented an approach which helps to make decisions even if values of attributes are not up-to-date. Two situations were considered: first, when we know the number of changes of attributes; second, when we know only the time passed after the last update. In both cases Markov Chains allow
us predicting the probability of policy failure and risk assessment helps to make a rational decision about allowing or denying further access. We also proposed two way of making a decision for a complex policy, depending on the type of cost available to us. We made a prototype of the proposed usage control model. The results of testing showed, that the model gives accurate predictions, and the performance of the model is good enough for application in dynamic systems.

As the future work we would like to make the approach more effective by using the values (probabilities) found during the previous access check rather than recomputing the values from the beginning. Such on-line approach should significantly reduce the computational cost. Another possible direction of the model improvement is to consider cases of dependent attributes. This issue requires complex mathematical models for implementing correlated Markov chains.

References


**D Method for Risk and Costs Assessment with Particular Focus on Web Applications**

Jorge Cuellar       Bernd Grobauer       Bernd Meyer
Martin Ochoa-Ronderos       Jan Stijohann

**Abstract:** This paper reports work-in-progress aimed at the design of a generic tool for the generation of formal risk and threat models of web applications. The tool shall rely on a database of available known common vulnerabilities and threat scenarios and assist the user in the construction of a risk model for a target system, with a parameter instantiation of probabilities, costs, etc. Appropriate filter functions can be used for the presentation of the results of the analysis and for writing documentation. Therefore, the planned tool will be extending state-of-the-art methods and techniques for model-driven risk analysis.

The tool generates a risk model from a description of attack steps. Attack steps are simple rules formulated in an enhanced propositional logic style. The formulation has a clearly defined semantic and is powerful enough for the description of general reusable components. A specific threat or risk is an ordered sequence of possible attack steps. This approach reflects directly the heterogeneous and compositional structure of the original security problem.

**Keywords:** Web Applications, Risk Management, Vulnerabilities, SDLC,

**D.1 Motivation**

Although the range of IT systems is very wide, and web applications are only one of many possible types of IT systems, there is an urgent need to secure web applications and services. This is due to the following:

1. The number and importance of web applications and services are steadily increasing. Many other services and solutions are becoming web-based.

2. The overall structure of web applications and services is common and standard; many security related issues in different instances of web applications and services are the same.

3. They are arguably the most important sources of insecurities regarding ITC systems. The list of insecurities are being propagated in the Internet rapidly, and most attackers have information about how to try to penetrate such systems.

There is a continuous trend to transfer applications and services to servers in the Internet. This enables service providers to present and sell their offerings and their computing resources to many customers simultaneously, with benefits for all parties involved. Customers and end-users can realise a significant cost reduction since they do not need to invest in their own computing equipment, in installing or developing software, and in the administration of complex IT applications and infrastructures. Service and infrastructure providers benefit from higher return on investment for hard- and software. This process will even be more important for Future Internet scenarios due to their increasingly heterogeneous and compositional nature.

The adequate protection of data, business processes, and applications is a fundamental requirement for the usability of web applications and services. Customers expect guaranteed confidentiality and integrity of their data, availability and quality of the services, and controlled access to the applications. In order to reach an adequate level of security for the corresponding applications a risk aware approach to the SDLC is needed that enables a modular design of complex models for formal risk analysis.

The assurance that the potential threats do not yield unacceptable risks is a requirement in a security certification above a certain level of strength. Many of the formal tools for risk modelling were developed
to fulfil the requirements of certification processes. Although there are several general purpose tools for risk modelling, risk evaluation, and treatment, there is still a gap between the theoretical concepts and their practical usability:

• There exist vast lists with potential risks, threats, and vulnerabilities for web based applications. But the currently available risks analysis methods and tools are missing a language to consistently represent or reason about them, instantiate them with parameters or values. In order to make use of generic lists of web application threats, risks, and vulnerabilities, an enhanced logic-style notation would be helpful. This language can be used to represent attacks as sequences of steps, each one with the necessary prerequisites, resources, capabilities, and consequences.

• The methods for model-driven risk analysis can lead to highly complex risk models, even for case studies of moderate size. The descriptions of the target systems quickly become huge and unwieldy, making the method prone to errors and omissions. Also the interpretation of the results of an analysis is often difficult since mechanisms for filtering of the threat scenarios conflict with the generality of the approach.

• A large amount of work has resulted in generic models for a large class of web applications and web services. But in order to use them in a concrete system, many parameters need to be adjusted: number and type of interacting components, network topology, skills and capabilities of attackers, assets to protect, targeted level of security and more. The adaptation of the parameters of an generic model is a painful work that could be automated, reducing the probability of errors during the construction phase of the model.

We envision a generic tool for the generation of formal risk and threat models of web applications. The tool shall rely on a database of available known common vulnerabilities and threat scenarios and assist the user in the construction of a risk model and a parametrisation for a target system. Appropriate filter functions can be used for the presentation of the results of the analysis and for writing documentation. Therefore, the planned tool will be extending state-of-the-art methods and techniques for model-driven risk analysis.

The tool generates a risk model from a description of attack steps. Attack steps are simple rules formulated in an enhanced propositional logic style. The formulation has a clearly defined semantic and is powerful enough for the description of general reusable components. A specific threat or risk is an ordered sequence of possible attack steps. This approach reflects directly the heterogeneous and compositional structure of the original security problem.

We do not expect that such a tool will reach the expertise and know how of a security specialist. Also, tuning of the results will still be necessary. But the tool can help the designer of a security system to accelerate cumbersome recurring tasks when modelling more complicate scenarios and to increase the overall level of security by avoiding errors and omissions in the analysis.

D.2 Main Concepts: Vulnerability, Loss Event, Attack, Step, and Capability

The notions of vulnerability [9, 4] and loss event are central to the analysis of risk in IT systems. For instance, ISO 27005 [6] defines risk as “the potential that a given threat will exploit vulnerabilities of an asset or group of assets and thereby cause harm to the organisation”. (The events that cause harm to the organisation are, of course, the loss events). But the relation between those two central concepts – vulnerability and loss event – is not straightforward; several concepts are needed to link them.

That is because the mere presence of a vulnerability does not ensure the existence of a possible loss event, and does not determine the probability or cost of such a loss event. They all depend, in particular, on the possibility of the attacker to perform an attack, a series of different steps that lead to a loss event. It is not surprising that in the literature it is common to see an attack as the path of attack steps. This is done for instance in attack patterns, see for instance [2, 1, 3].
In other words, exploiting a vulnerability does not always imply directly a loss event, but can be a useful step in that direction. Each attack step may exploit one or several vulnerabilities, but in general the attacker may need some preconditions to access the vulnerability. We call them capabilities. For instance, to perform a step, the attacker may require certain credentials that he acquired in a previous step, or physical access to certain parts of a system. This notion of capability is often implicit in existing formalisations or representations of attacks or attack steps. A step that does not result in a loss event is performed by the attacker with the purpose of obtaining new capabilities that can be used in a later step. A step may provide him some knowledge or access to a resource in a way that then he is able to read certain data or to execute some code. Concisely, we will use capabilities to model the states of a transition system: the attack steps modify the capabilities of the attacker. This explicit use of capabilities is useful to derive attack paths: an attacker carries out a series of attack steps in order to gain certain capabilities that allow him to perform further steps that finally allow him to reach his aim, e.g., to read or modify certain data, sabotage a certain system, etc.

Other factors that determine if a vulnerability gives rise to an attack step are

1. the frequency with which threat agents try to exploit a vulnerability and
2. the success rate, influenced by the difference between the threat agent's attack strength and the strength of the system to resist the attack.

The attack frequency, in turn, is determined by the threat agents motivation to carry out an attack action (“What can he gain with an attack?”, “How much effort does it take?”, “What is the risk for him?”, etc.) and, again by the basic capabilities of a generic attacker. (If the attacker notices that he is unable to obtain basic capabilities, he will desist more rapidly from the attack).

Regarding the strength of the system to resist the attack, notice that the Open Group’s Risk Taxonomy [10] states:

**Vulnerability is the probability that an asset will be unable to resist the actions of a threat agent. Vulnerability exists when there is a difference between the force being applied by the threat agent, and an object’s ability to resist that force.**

In other words, even if a vulnerability is present, there may be also supplementary mechanisms in the system that resist the exploitation of the vulnerability.

### D.3 Abstraction Levels, Vulnerability Assessments and Threat Modelling

Risk analysis of web applications can be performed at different abstraction levels and in different stages of the development life cycle.

Consider first, for instance, a directory and the collaboration services of a large enterprise with several hundreds of thousands of users and their devices. Here, it is impossible to analyse in detail the possible vulnerabilities of each involved component. It may be true that the single elements have vulnerabilities, but the enterprise has a set of independent security controls that are used to resist the attackers trying to access those vulnerabilities. Thus, it is necessary to abstract away from the single software vulnerabilities and, instead, use rough estimations of the resistance that the security controls offer. On the other hand, a rather detailed threat modelling is possible, including the attacker’s motivation and resources, his possible attack steps, and the needed capabilities. This is reasonable since the deployment environment is known and the possible threat agent profiles can be anticipated. We say that in a global scenario, with many elements, an abstract threat modelling dominates the risk analysis.

---

1 In most cases, the step of one agent modifies only his own capabilities, but sometimes, it may change the global state of the system, changing thereby the capabilities of all attackers. For instance, the attacker may publish a list of passwords in a blackboard, a memory location that is visible by many attackers, or directly in a forum in Internet.

2 We talk of an “attacker” if we have one particular in mind, say performing a series of steps or attempting to achieve a certain goal. But we refer to “threat agents”, collectively, a the set of attackers, each one following perhaps a different strategy, etc. The main point, formally, is that threat agents are independent of each other, and each one has different knowledge and capabilities.
Now consider the case of a single software application, say in the earlier phases of the SDLC, during the design or implementation, prior to the release. The development team has a lot of information regarding the software design decisions, the architecture, the actions taken during the SDLC, etc. In this case, a more concrete and detailed vulnerability assessment becomes easier to obtain and dominates the risk analysis.

Recognising that there exist standard lists of best-practices, attack patterns, and common vulnerabilities, and that the fact that they offer a high degree of reusability, we propose a methodology for supporting model-based risk analysis for web services based on two central pillars:

**Rule-based vulnerability assessments.** It is assumed that following the best-practices regarding the activities to be performed during the SDLC, software artifacts\(^3\), and architectures of web services, the most common well known vulnerabilities can be avoided. During a typical system development many reasons (time pressure, the education and skills of the implementors, performance reasons, etc) may lead to deviations from the best practices. The type and amount of this deviation may be used to estimate the probabilities of possible vulnerabilities in the system. We propose to obtain rules that help us precisely in describing this dependency.

**Capability-based threat modelling.** Based on a collection of attack patterns and steps, for which capabilities, exploited vulnerabilities, and required attacker strength (expertise, resources, etc.) have been made explicit, threat modelling is to be supported.

In fact, both approaches, threat modelling and vulnerability assessment, depend on and complement each other. A combination yields flexible possibilities to perform risk analyses for systems of different purposes, varying sizes, and in different phases of the SDLC.

### D.3.1 Vulnerability Assessment

The goals of the intended vulnerability assessment method are:

1. to identify possible vulnerabilities of a given system, together with an estimation of their probabilities. This also highlights the components and design aspects that entail an increased vulnerability risk,

2. to improve the prediction of the general security level of a given system, and

3. to understand the available options with respect to their risk and cost effects, and to determine the most important next steps required to secure the system, from an economic point of view.

Obviously, to achieve those goals, we will need "measurements" of the current target of analysis, that quantify how complex the application is, what steps were done in the SDLC, the basic architecture, which security controls it has and how these are designed and implemented, what assets must be protected, etc.

To achieve goals (2) and (3), we will create risk models. They will be based on a large set of action rules, say of the form:

\[
vul_1(p_1, p_2, \ldots) \& cap_1(\ldots) \& att(a_1, a_2, \ldots) = \cdots
\]

where \(vul_1\) is a given vulnerability, with parameters \(p_1, p_2, \ldots\), (say, the probability of the existence of the vulnerability, and other relevant properties of the vulnerability); \(cap_1\) is a capability of the attacker (with some parameters); and \(att\) are attributes of the attacker, such as his motivation or skills. Of course, there can be several vulnerabilities, capabilities or attributes in one such rule.

What we need is to inspect ("measure") the current target application, including the deployment (real or intended), and the environment, to deduce values for the parameters of the possible vulnerabilities and attackers (and of the initial capabilities of the attackers) \(p_1, \ldots, a_1\) etc.

Thus we will have:

---

\(^3\)Software artifact is a tangible by-product created during the development of software and possibly documenting the activities of the SDLC. A SW artifact can be for instance a requirements document.
Measurement variables, or “indicators”. The properties of the target application that we want to measure,

Measurement points. The places where we can obtain the concrete data or information to assign values to the measurement variables. Notice that sometimes one variable can be measured at two different points (perhaps with different accuracy, cost, etc).

Measurement sources. The general documents and approaches that give access to the measurement points (see below for examples).

Measurement values, or simply “measurements”. The concrete values that are observed for a target application.

Parameters. The consolidated vulnerability information that has the suitable format to be used in a rule.

The first step to achieve the afore-noted goals is to obtain the measurements which form the base of our method. We identified three major sources that can provide access to the belonging measurement points:

1. Software development artifacts (including source code)
2. Questionnaires and check lists to be filled out by the developers and software architects
3. Testing results
4. Statistics of vulnerabilities, including publications such as vulnerability data bases, OWASP, etc.

Since an indicator may be obtained from more than one of these sources, one must choose which source will be most convenient or reliable for which indicator. For instance, a questionnaire could ask if the developers use best practice methods to handle SQL queries. It would be more reliable, though, to look directly into the source code because a developer may not have the current state of the entire source code in his mind. In addition, reviewing source code can be too expensive or even impossible. A questionnaire, on the other hand, may be more convenient for capturing aspects such as the degree of experience of the development team, their secure coding skills and security awareness.

Leveraging all of the different sources, we will be able to cover indicators from different abstraction levels and from different phases of the SDLC, e.g.:

General procedural aspects. Does an intrusion detection concept exist? Is the target system certified, e.g. ISO 27001? Do secure coding guidelines exist?

Requirements engineering. Is the necessity to validate all input data contained in the requirement documents? Is sensitive data identified and the need for its confidentiality and integrity considered? Are trust boundaries described, and if yes, are they correct?

Architecture & design. Does the architecture reflect the security requirements, e.g. the trust boundaries? What security controls do exist? How are they designed? Which third party components are used? How many interfaces do exist?

Implementation. Are temp files created in a secure way? How many lines of code does the application have?

Deployment. Are there any relevant firewalls? Is the application connected to the internet? Is it in a DMZ?

Testing. How is the resistance to fuzzing and penetration tests?

In a second step, the collected measurements are used to calculate the parameters for the rules, as explained above. For this purpose, the measurements are further processed. Depending on the kind of indicator this may be
A simple classification of quantitative values (e.g. lines of code, number of interfaces, number of users)

A comparison to a reference value (e.g. number of roles, resistance to fuzzing tests)

The determination of the deviation from accepted best practices, in terms of the secure development life cycle activities, the security architecture chosen, and the software artifacts present, especially regarding the security mechanisms used

For our analysis, a necessary prerequisite is a catalogue of threats and vulnerabilities and a categorisation of the core problems to be addressed.

**Challenges of an Indicator-based Vulnerability Assessment**

One challenge in our approach is to find a catalogue of most significant indicators and a methodology for choosing an appropriate set of them for a given target application.

This set should cover the most significant factors of influence, but the effort in measuring the values should still be efficiently manageable. Unfortunately, in software security, tiny details can make a difference. Consequently, one has to make sure that the chosen measurements cover as many of such details as possible.

**D.3.2 Capability-based Threat Modelling**

As foundation for the methodology, a collection of attack patterns/steps has to be created. For each attack step, the following is made explicit:

1. Which (access) capabilities are required for carrying out the attack step?
2. Which capabilities the successful execution of the attack step yields?
3. Which vulnerabilities are exploited by the attack step?
4. What attack strength (expertise, resources, etc.) is required to carry out the attack step?

We envision the following steps in capability-based threat modelling:

- A model of the system under consideration is created that shows how each component of the system provides access to other parts of the system (including information assets)
- The system model is enriched with information about likely vulnerabilities as elicited by template-based vulnerability assessment (Section D.3.1)
- The attackers are modelled by defining their initial capabilities, motivations (in terms of capabilities they want to gain), and attack strengths.
- From the system owner's view, loss events are modelled in terms of attackers gaining certain capabilities
- Based on this information (i.e., the system model, information about likely system vulnerabilities, the attackers, and loss events as identified by the system owner) attack paths and success probabilities for these attack paths are then derived

To turn our vision into reality, the following steps must be undertaken:

- Definition of a suitable risk ontology (see Section D.4.1 for preliminary results) that contains all required concepts and puts them into relation
- Definition of languages for expressing (1) attack steps and capabilities (see Section D.4.2 for preliminary results); (2) attacker strength, vulnerabilities, and success probabilities; (3) system architecture and properties relevant for the application of attack steps to system components
• Design of methods and algorithms that automate the derivation of attack paths and calculation of probabilities

• Implementation of tool support for (1) creation and maintenance of a dictionary of attack steps, (2) the modelling of system, attackers, and loss events, (3) automated derivation of relevant attack paths and their success probabilities, and (4) interactive analysis of the results as part of the overall risk-analysis process

D.4 Preliminary Results

In this section we introduce preliminary results that go in the intended direction described above. On the one hand, we propose a new suitable risk ontology that is fine-grained enough to explicitly model attack steps. On the other hand we discuss possible formalisations that would allow to automatically check for paths leading to an attack given a catalogue of atomic attack steps.

D.4.1 A Suitable Risk Ontology

In order to support the methodology envisioned above in Section D.2, we introduce a suitable risk ontology. Section D.4.1 motivates, why we create a new ontology rather than using one of the existing ones; Section D.4.1 describes the ontology.

Motivation for a New Risk Ontology

Several proposals for ontologies regarding information risk exist [8, 5], but do not adequately address certain concepts that are central to the methodology as envisioned above:

1. Attacks usually encompass a series of actions that step by step lead an attacker toward his aim. With each step, the attacker gains certain capabilities (e.g., access to user credentials, access to the login page of an application, increased user rights, etc.), until he finally has gained enough capabilities to carry out his aim.

   While the concept of attacks progressing in stages is central to many risk ontologies [8, 5], the concept of capabilities is at best implicit. The ontology presented below makes the concept of capabilities explicit.

2. In order to achieve the high degree of reusability as envisioned above, the ontology must be centred around the most basic and general concepts of risk. The existing extensive collections of attack patterns [2, 1, 3] and weaknesses [9, 4] suggest that attack patterns (more generally, attack actions) and weaknesses/vulnerabilities are hot candidates for central, reusable concepts of a risk ontology.

   As the Open Group's risk taxonomy [10] convincingly argues, a vulnerability exists when there is a difference between the force being applied by the threat agent, and an object's ability to resist that force. In other words: a vulnerability on one hand and the attacker's action on the other hand are two sides of the same coin and must always be examined together. The ontology presented below is centred around the notion of attacker actions. In this respect, the proposed ontology is very similar to that of Elahi et al. [5].

3. The success of an action (i.e., the exploitation of a certain vulnerability in a certain way with a certain aim) depends on the resources and expertise of an attacker. Experience shows that a population of (potential) attackers can usually not be generalised. For example, attacks on a web application may be carried out by complete novices (who do little more than try around with changing URL parameters), “script kiddies” who follow attack recipes for attacks such as SQL injection and cross-site scripting, and professional attackers with state-of-the-art expertise in penetrating web applications. The FAIR methodology [7], for example, models the attack strength of an attacker population as a probability distribution.

   The ontology presented below includes the concept of attacker resources/expertise.
The Ontology

Figure D.1 shows a UML class diagram of the proposed risk ontology. In the following we treat "attacker" as synonymous to "threat agent", as well as we treat "step" and "action" as synonymous. For example we can include non-malicious threat agents such as normal users who, through flawed behaviour may cause information risk. Also, not all actions carried out by an attacker must necessarily exploit a vulnerability, but may be actions that would be completely legitimate when carried out by an authorised user: for example, logging into an application with user credentials does not exploit a vulnerability (although, a vulnerability may have been exploited at a previous step in order to obtain user credentials).

As elaborated above, central to the ontology is the concept of action with its dependence on capabilities, vulnerabilities, and resources/expertise of the attacker: necessary precondition for the execution of an action by an attacker against an object (more about objects below) is that the attacker has certain capabilities. Further, an action may also require certain resources/expertise on part of the attacker and the presence of certain vulnerabilities in the targeted object, but whether resources/expertise/vulnerabilities constitute necessary preconditions or are treated as factors that influence the success probability of an action is not predetermined by the ontology: there are numerous ways to reason about success pro-

But we use the terms mostly in different contexts: an ‘attacker’ performs a series of ‘steps’, following a strategy and trying to achieve a goal. A ‘threat agent’ performs an ‘action’, independently of a particular context. When we discuss attacks, we talk about ‘attacker’ and ‘attack steps’, when we discuss possible vulnerabilities, their pre-conditions, their results, etc., we prefer to use the terms ‘threat agent’ and ‘action’.

Figure D.1: Ontology for capability-based risk analysis

NESSoS - 256980
babilities and many methodologies will treat resources/expertise and presence of vulnerabilities as factors influencing probabilities rather than "binary" conditions that are or are not present.

As stated above, actions act on objects, where "object" must be understood in its most general interpretation: for the purpose of this ontology, an object is everything that may provide access in some way either to some other object or to an information asset.

For an example regarding the concepts introduced so far, imagine a building that contains a database server on which valuable information is kept. A thief is then an attacker that performs an action “break in” on the building, pitting his resources/expertise in housebreaking against the resistance strength of the building. Since no building is 100% safe against burglary, it has some vulnerability which the thief may or may not be able to exploit. If he does, he gains capabilities regarding access to the application server contained in the building. Further actions may finally give him capabilities to access the information assets on the database server.

Continuing with the ontology, we turn to the threat scenario. A threat scenario is tying everything together: an attacker performs a series of actions that may lead to one or more unwanted events, i.e., occurrences that have a certain negative impact for the information owner of the affected information asset. Here, a complication for modelling risk is that unwanted event and associated impact are regarded from the information owner’s perspective, while the concepts around the attacker are described from the attacker’s perspective. This shift of perspective is necessary, because an attacker’s aim may be quite different from the unwanted events and associated impact his actions generate. The thief from the example given above may, for example, be content with accessing a single data set, but the fact that the attacker had capabilities to access many data sets (even though he did not access them) may have an impact, e.g., because of regulatory reasons or because of image loss. For this reason, our understanding is that an unwanted event occurs always, when a certain attacker has certain capabilities. What the impact of an unwanted event is, then depends on numerous factors, many of which are necessarily external to this ontology, as its intended use is to model how unwanted events may happen rather than what they might cost – there exist ontologies that treat the concept of impact in more detail [7]. Factors that relate to concepts present in our ontology are those factors that pertain to the attacker, namely his motivation/mindset as well as his resources/expertise – this time not regarding his ability to exploit vulnerabilities but to use the gained capabilities in a way that causes actual loss.

D.4.2 A First Language Proposal for Attack Transitions

There exist a well documented set of vulnerabilities and attacks against web applications and services in catalogues like OWASP. As argued above, many of these attacks are a composition of small steps (actions), which gradually increase an attacker’s capabilities. In this section we discuss a formal model for these actions that allows for automatically computation all paths (sequences of steps) that lead to a given attack.

D.4.3 Attack Steps

Following the previously discussed ontology, in order to attack a given asset, an attacker must possess a set of capabilities (e.g., reading or writing permissions in a given machine). That is, he needs a set $L^A$ of capabilities that lead to attack $A$. This process is often decomposable in many steps, that gradually allow the attacker to perform the attack by gaining intermediate capabilities, should exploitable vulnerabilities exist. In its simplest form, an action is a triple consisting of a list of capabilities, the success probability of the action and a new list of capabilities gained after the realisation of the step. We can express this formally as:

$$a : L \xrightarrow{p} L'$$

where $a$ is a label for the attack step, $L$ is the original list of capabilities, $p$ is the probability that this particular attack step takes place, and $L'$ is the new list of capabilities.

---

5https://www.owasp.org/

6In general $p$ could also be a vector of probabilities, this allows to distinguish between the attacker’s motivation and the technical probability of performing the attack for example.
It is reasonable to assume that if performing action \( a \) an attacker gains capabilities \( L' \) and performing \( b \) he gains capabilities \( L'' \), then by performing \( a \) followed by \( b \) he gains \( L' \cup L'' \). We denote the final attack step as

\[ L^A \models p A \]

where \( p \) is 1 if we only consider the technical probability of performing the attack, that is, \( L^A \) is the minimum amount of capabilities such that an attacker would certainly be able to perform attack \( A \). As noted before, \( p \) could be also a vector containing two or more values describing other probabilities, such as the motivation to actually perform the attack. To deduce what sequence of actions lead to \( A \), one could see the list \( L \) of capabilities as a conjunction \( \ell_1 \land \cdots \land \ell_n \) of atomic capabilities and then embed the attack steps in propositional logic (initially omitting the probabilities). Constructing a proof of the derivation of \( L^A \) would imply the existence of a sequence of steps leading to \( A \), thus it is equivalent to the satisfiability of \( L^A \). In this model the probabilities of attack steps are attributes and are not needed to deduce a sequence of steps implementing a certain attack. Moreover, the description in propositional logic implies that capabilities acquired during steps of an attack will persist in order to have well-defined truth values for these capabilities. This can be however problematic if one wishes to model attacks where an adversary can also lose some of its capabilities/resources after performing the attack step, and therefore a model using propositional logic would only approximate attacks, because some attacks paths would no longer be reasonable (if for example in reality some capabilities/resources are actually lost due to some actions included in the automatically founded attack step).

### D.4.4 Further Properties

#### Complex states.

Alternatively, one could think of the attack steps as a state transition relation. That is, given a state consisting of the product of boolean variables representing all relevant capabilities (i.e. the capabilities appearing in some attack step) then each attack step defines a probability of going from some state \( S \) where \( L \) capabilities are true to a new state \( T \) where \( L' \) holds. In this setting, to find the sequence of attack paths leading to a set \( A \) of capabilities means querying for the reachability of states where \( L^A \) holds. This is formally equivalent to the satisfiability version discussed above, however it allows for more flexibility when describing more complex attack steps. It is for example reasonable to allow for more complex states including variables of integer type (within a certain finite range) that could model the amount of certain tokens available to the adversary, the amount of money required prior to an attack etc.

Another advantage of this formalisation is that it is the natural input for modern model-checkers like nuSMV\(^7\), allowing checking more complex temporal logic propositions that go beyond simple reachability.

#### Parameterised attack steps.

Although initially we plan to analyse collections of simple attack steps, an interesting and natural further step will be parameterising the capabilities to generalise them in contexts where similar assets/actors or other factors can be extracted. For example, the capability ‘read password file’ could be parametrised to represent this capability in a machine from a finite list: ‘read password file(\( X \in \text{Machines} \))’. One could even go further and generalise this reading capability to a list of files: ‘read \( Y \in \text{Files} \ (X \in \text{Machines})\)’. This would allow for defining also parametrised attacks of the form \( A(x_1, \ldots, x_n) \) where the variables are used to abstract from machines, users, etc.

#### Consumption of capabilities and resources.

It might be necessary for modelling complex attack scenarios to have a notion for the consumption of capabilities and resources. Examples are credentials for authentication, number of machines that are on-line, time for access, and money. If capabilities or resources can be gained and lost again along an attack path a state machine description of the steps of an attack would be needed. The probabilities associated with attack steps lead then to a Markov model of state transition.

---

\(^7\)http://nusmv.fbk.eu/
D.4.5 Verification Techniques and Tools

To automatically search for attack sequences given a repository of attack steps there are different possibilities that depend on the formal interpretation of the attack steps as discussed above. In its simplest form, where the attack steps can be embedded in propositional logic, a natural choice is to check for sequences using for example Prolog or automatic theorem provers that allow for querying all proof derivation trees of the goal. For the state transformation approach, it would be however more reasonable to use a model checker like the aforementioned nuSMV or probabilistic model checkers to account for the probabilities. Another possibility would be to ad-hoc evaluate reachability in a graph using a functional language like Haskell, that allows flexible and efficient graph filtering. We plan to compare the different implementation approaches and report on them in future deliverables.

D.5 Conclusion

In this paper, we outlined our idea for a model-based risk analysis for web services. The proposed methodology is based on two central pillars: rule-based vulnerability assessments and capability-based threat modeling. A combination of both approaches yields flexible possibilities to perform risk analyses for systems of different purposes, varying sizes, and in different phases of the SDLC. For instance, in a global scenario, with many elements, a rather abstract threat modeling can produce valuable results for the overall risk analysis even though a detailed analysis of all software components is not possible. In case of a single software application, on the other hand, during the design or implementation phase, a more detailed vulnerability assessment can cover significant risk factors although the later deployment environment and possible attacker profiles are not yet known.

Future Work  As far as the vulnerability assessment is concerned, we need to find and evaluate significant “indicators”, i.e. suitable properties of a target application which influence or correlate to a vulnerability. For the most abstract category, general procedural aspects, we plan to study existing proven secure development processes such as the Microsoft SDL in order to obtain relevant indicators. In the case of requirements engineering, architecture, and design the choice of the specific indicators is highly dependent on the actual target application. A good threat model of a target application requires a systematic consideration of dynamic indicators such as trust boundaries, (non-) existing security controls, their design, integrated third party components, the number and kind of interfaces present. For category implementation, we want to find out to what extend fuzzing and penetration testing approaches can be used to obtain reliable indicators. Especially a possibly (semi-) standardized fuzzing may be well-suited to efficiently measure certain implementation indicators, such as memory corruption bugs, in contrast to a rather time-consuming and complex code-review. Having a first set of indicators for each category, we need to evaluate them with real-world target applications.

For category implementation, we want to find out to what extent fuzzing and penetration testing approaches can be used to obtain reliable indicators. Especially a possibly (semi-) standardized fuzzing may be well-suited to efficiently measure certain implementation indicators, such as memory corruption bugs, in contrast to a rather time-consuming and complex code-review. Better scalable approaches to gather source / binary code related or even run-time indicators are fully or semi-automated tools such as code analyzers, scriptable debuggers and dynamic analysis tools (e.g. from sysinternals, www.sysinternals.com). Having a first set of indicators for each category, we need to evaluate them with real-world target applications.

References


E Formal Foundations for Risk Management

Mass Souldal Lund Bjørnar Solhaug

Abstract: A formal foundation for risk management facilitates risk analysis by providing a rigorous basis for analysing and reasoning about risks. This paper presents risk graphs as a means for modelling and documenting unwanted incidents along with their likelihoods and the scenarios that lead to them. A formal semantics is provided that precisely describes the intended interpretation of risk graphs. Risk graphs support various kinds of reasoning about risk. A calculus is introduced with rules for such reasoning, the soundness of which can be shown with respect to the underlying semantics. Risk modelling is an important technique to aid the process of identifying and documenting risks, as well as the process of estimating likelihoods and consequences. The formal foundation of risk graphs therefore provides important formal foundation for risk management. An advantage of risk graphs is that they can be understood as a common abstraction of several established risk modelling techniques, and their formalisation can therefore be used to explain and reason about these.

Keywords: Risk management, risk modelling, risk graphs, formal semantics, calculus.

E.1 Introduction

Risk analysis involves the process of understanding the nature of risks and determining the level of risk [7]. Risk modelling refers to techniques that are used to aid the process of identifying, documenting and estimating likelihoods and consequences of unwanted incidents. An unwanted incident is an event that harms or reduces the value of an asset (such as availability or confidentiality), and a risk is the likelihood of an unwanted incident and its consequence for a specific asset. In this paper we present an approach to risk modelling referred to as risk graphs [2]. By providing a formal semantics for risk graphs, we are provided a formal foundation for reasoning about risk, and therefore also a formal foundation for risk management in general.

We introduce the syntax and semantics of risk graphs in Section E.2. In Section E.3 we provide a calculus for analysing likelihoods in risk graphs. In Sect. E.4 we introduce dependent risk graphs and a calculus for reasoning about assumptions in risk models. In Section E.5 we generalise risk graphs to enable modelling and reasoning about changing risks, and in Section E.6 provide a means for relating risk graphs to target models. In Section E.7 we demonstrate how risk graphs can be instantiated with the CORAS risk modelling language.

E.2 Risk Graphs

A risk model is a structured way of representing an unwanted incident and its causes and consequences by means of graphs, trees or block diagrams [12]. We introduce risk graphs as an aid for structuring events and scenarios leading to incidents, and for estimating likelihoods of incidents. There exist several modelling techniques that can be used for such structuring of scenarios and incidents, and for the reasoning about likelihoods of incidents, for example fault trees [5], event trees [6], attack trees [13], cause-consequence diagrams [9], Bayesian networks [1] and CORAS threat diagrams [8]. Risk graphs can be understood as a common abstraction of these modelling techniques [2]. By giving formal semantics to risk graphs, we thereby also provide a risk model semantics that can be used to explain and reason about several established approaches to risk modelling. Hence, the formalisation of risk graphs can serve as a formal foundation for risk management with wide flexibility in the chosen approach to risk and threat modelling.
A risk graph consists of vertices (representing threat scenarios) and a finite set of directed relations (representing the “leads to” relationship) between them. An example risk graph is shown in Fig. E.1. Each vertex in a risk graph is assigned a set of likelihood values representing the estimated likelihood for the scenario to occur. The assignment of several likelihood values, typically a likelihood interval, represents underspecification of the likelihood estimate. A relation from vertex \( v \) to vertex \( v' \) means that \( v \) may lead to \( v' \). Also the relations can be assigned likelihood sets. These are conditional likelihoods that specify the likelihood for a scenario leading to another scenario when the former occurs. One threat scenario may lead to several other threat scenarios, so the probabilities on the relations leading from a threat scenario may add up to more than 1. A risk graph is furthermore allowed to be incomplete in the sense that a given threat scenario may lead to more scenarios than what is accounted for in the risk graph. The probabilities of the relations leading from a threat scenario may for this reason also add up to less than 1.

### E.2.1 The Syntax of Risk Graphs

Formally a risk graph is a set \( D \) of elements \( e \). An element is a vertex \( v \) or a relation \( v \rightarrow v' \). Let \( P \subseteq [0,1] \) denote a probability set. We then write \( v(P) \) to indicate that the probability set \( P \) is assigned to \( v \). Similarly, we write \( v \xrightarrow{P} v' \) to indicate that the probability set \( P \) is assigned to the relation from \( v \) to \( v' \). If no probability set is explicitly assigned, we assume the probability set assigned to the element to be \([0,1]\), i.e. that the probability is fully underspecified. Using this textual notation, the risk graph shown in Figure E.1 can be represented by

\[
D = \{ v_1(P_1), v_2(P_2), v_3(P_3), v_4(P_4), v_5(P_5), v_6(P_6), v_7(P_7), \}
\]

\[
v_1 \xrightarrow{P_{1}} v_3, v_2 \xrightarrow{P_{2}} v_3, v_3 \xrightarrow{P_{3}} v_4, v_4 \xrightarrow{P_{4}} v_7, v_5 \xrightarrow{P_{5}} v_6, v_6 \xrightarrow{P_{6}} v_7 \}
\]

### E.2.2 The Semantics of Risk Graphs

Risk graphs are used for the purpose of documenting and reasoning about risks, particularly the documentation and analysis of threat scenarios and unwanted incidents and their likelihoods. The approach of [2] assumes that scenarios and their probabilities are represented by a probability space [3] on traces of events. We let \( \mathcal{H} \) denote the set of all traces (both finite and infinite) and \( \mathcal{H}_f \) the set of all finite traces. A probability space is a triple \((\mathcal{H}, \mathcal{F}, \mu)\). \( \mathcal{H} \) is the sample space, i.e. the set of possible outcomes, which in our case is the set of all traces. \( \mathcal{F} \) is the set of measurable subsets of the sample space, and \( \mu \) is a measure that assigns a probability to each element in \( \mathcal{F} \). The semantics of a risk graph is statements about the probabilities of the trace sets that represent vertices or the composition of vertices. In other words, the semantics is a set of statements about the measure \( \mu \).

For composition of vertices, \( v \sqcap v' \) denotes the occurrence of both \( v \) and \( v' \) where the former occurs before the latter. We let \( v \sqcup v' \) denote the occurrence of at least one of \( v \) and \( v' \). A vertex is atomic if it is not of the form \( v \sqcap v' \) or \( v \sqcup v' \). We use lower case \( v \) as the naming convention for arbitrary vertices, and upper case \( V \) as the naming convention for the set of finite traces representing the vertex \( v \).

When defining the semantics of risk graphs we use the auxiliary function \( tr(\_\_) \) that yields a set of finite traces from an atomic or combined vertex. Intuitively, \( tr(v) \) is the set of all possible traces leading up to
and through the vertex \( v \), without continuing further. The function is defined by

\[
\begin{align*}
\text{tr}(v) \&= H \text{ where } v \text{ is an atomic vertex} \\
\text{tr}(v \sqcap v') \&= \text{tr}(v) \sqcap \text{tr}(v') \\
\text{tr}(v \sqcup v') \&= \text{tr}(v) \sqcup \text{tr}(v')
\end{align*}
\]

where \( \sqcup \) is the operator for sequential composition of trace sets, for example weak sequencing in UML sequence diagrams [4]. Notice that the definition of the composition \( v \sqcap v' \) does not require \( v \) to occur immediately before \( v' \). The definition implies that \( \text{tr}(v \sqcap v') \) includes traces from \( v \) to \( v' \) via finite detours.

A probability interval \( P \) assigned to \( v \), denoted \( v(P) \), means that the likelihood of going through \( v \) is a value \( p \in P \), independent of what happens before or after \( v \). The semantics of a vertex is defined by

\[
[v(P)] \&= \mu_c(\text{tr}(v)) \in P
\]

where the expression \( \mu_c(S) \) denotes the probability of any continuation of the trace set \( S \subseteq H \), and is defined as

\[
\mu_c(S) \&= \mu(S \sqcap H)
\]

A probability interval \( P \) assigned to a relation \( v \rightarrow v' \) means that the likelihood of \( v' \) occurring after an occurrence of \( v \) is a value in \( P \). This likelihood is referred to as the conditional likelihood. The semantics of a relation is defined by

\[
[v \xrightarrow{P} v'] \&= \mu_c(\text{tr}(v \sqcap v')) \in \mu_c(\text{tr}(v)) \cdot P
\]

Our definitions of interval arithmetic in the setting of risk graphs are given in Fig. E.2.

The semantics \( [D] \) of a risk graph is the conjunction of the expressions defined by the elements in \( D \), formally defined as

\[
[D] \&= \Lambda_{e \in D}[e]
\]

A risk graph is said to be correct (with respect to the world or a specification of the relevant part of the world) if each of the conjuncts of \( [D] \) is true. We say that \( D \) is inconsistent if it is possible to deduce \( \text{False} \) from \( [D] \). Notice that \( [\emptyset] = \text{True} \).

**E.3 Reasoning about Likelihoods in Risk Graphs**

In this section we introduce rules for calculating probabilities of vertices in risk graphs, and we provide guidelines for consistency checking probabilities that are assigned to risk graphs.

The first rule is referred to as the *relation rule*, and captures the conditional likelihood semantics of a risk graph relation. For a vertex \( v \) that leads to \( v' \), the vertex \( v \sqcap v' \) denotes the occurrences of \( v' \) that happen after an occurrence of \( v \).
How to check consistency of likelihoods in risk graphs

**Exact values in complete diagrams**
- Assigned value: \( v(p) \)
- Calculated value: \( v(p') \)
- Consistency check: \( p = p' \)

**Exact values in incomplete diagrams**
- Assigned value: \( v(p) \)
- Calculated value: \( v(p') \)
- Consistency check: \( p \geq p' \)

**Intervals in complete diagrams**
- Assigned interval: \( [p_i, p_j] \)
- Calculated interval: \( [p'_i, p'_j] \)
- Consistency check: \( [p_i, p'_i] \subseteq [p_i, p_j] \) or, equivalently, \( p_i \leq p'_i \) and \( p_j \geq p'_j \)

**Intervals in incomplete diagrams**
- Assigned interval: \( [p_i, p_j] \)
- Calculated interval: \( [p'_i, p'_j] \)
- Consistency check: \( p_j \geq p'_j \)

<table>
<thead>
<tr>
<th>Table E.1: Guidelines for consistency checking likelihoods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rule E.1 (Relation)</strong> If there is a direct relation from ( v ) to ( v' ), we have:</td>
</tr>
</tbody>
</table>
| \[
| v(P) \quad \overset{v \leftarrow v'}{\longrightarrow} \quad v' \quad \quad (v \parallel v')(P \cdot P')
|\]

The second rule is referred to as the **mutually exclusive vertices rule**, and yields the probability of either \( v \) or \( v' \) occurring when the two vertices are mutually exclusive:

**Rule E.2 (Mutually exclusive vertices)** If the vertices \( v \) and \( v' \) are mutually exclusive, we have:

\[
\frac{v(P)}{v(P)} \quad \frac{v'(P')}{(v \parallel v')(P + P')}
\]

The third rule is referred to as the **statistically independent vertices rule**, and yields the probability of either \( v \) or \( v' \) occurring when the two vertices are statistically independent:

**Rule E.3 (Statistically independent vertices)** If vertices \( v \) and \( v' \) are statistically independent, we have:

\[
\frac{v(P)}{v(P)} \quad \frac{v'(P')}{(v \parallel v')(P + P' - P \cdot P')}
\]

As a small example of probability calculation consider the risk graph in Fig. E.1 and assume we want to calculate the probability of \( v_3 \) from \( v_1 \) and \( v_2 \). By Rule E.1 we calculate \((v_1 \cap v_2)(P_1 \cdot P_2)\) and \((v_2 \cap v_3)(P_2 \cdot P_3)\). Assuming that \( v_1 \) and \( v_2 \), as well as \( v_1 \cap v_3 \) and \( v_2 \cap v_3 \), are statistically independent, we use Rule E.3 to calculate \((v_1 \cap v_3 \cup (v_2 \cap v_3))(P_1 \cdot P_2 + P_2 \cdot P_3 - P_1 \cdot P_2 \cdot P_3)\).

Assuming that the likelihood estimates in Fig. E.1 are correct, there is still one issue to consider before we can conclude about the likelihood of the vertex \( v_3 \). The issue is whether or not the risk graph is complete. If the risk graph is complete, the graph shows all the possible ways in which \( v_3 \) may occur. In that case we have that \( v_3 = (v_1 \cap v_3) \cup (v_2 \cap v_3) \) and that \( P_3 = P_1 \cdot P_2 + P_2 \cdot P_3 - P_1 \cdot P_2 \cdot P_3 \) is the correct likelihood of this vertex. If the risk graph is incomplete, there may be further scenarios that can lead to \( v_3 \). In that case we only know that \( P_3 \) is the lower bound of the probability of \( v_3 \).

Consistency checking of risk models is important, as it is a useful means for detecting errors or misunderstandings of the risk estimates that are documented during a risk analysis. The basis for the consistency checking is the likelihood values that are already assigned to the vertices and relations of a risk

NESSoS - 256980 112
graph. The guidelines for consistency checking depend on whether the risk graph in question is complete, and whether the likelihoods are given as exact probabilities or as probability intervals. The guidelines are given in Table E.1.

As an example of consistency checking, consider the risk graph in Fig. E.1, assuming first that the graph is complete. By the above example, we know that the probability of the vertex \( v_3 \) is \( P_3' = P_1 \cdot P_a + P_2 \cdot P_b - P_1 \cdot P_a \cdot P_2 \cdot P_b \) given the vertices and relations that lead to this vertex. The assigned probability \( P_3 \) must therefore equal the calculated probability \( P_3' \) in order to be consistent with the preceding probability estimates if we are working with exact values. If the \( P \)'s are intervals, we must have that \( P_3' \subseteq P_3 \) for the risk graph to be consistent.

Discarding the assumption of the completeness of the graph gives the consistency requirement that the assigned probability \( P_3 \) must be greater than or equal to the calculated probability \( P_3' \), i.e. that \( P_3 \geq P_3' \), if we have exact values. On the other hand, if \( P_3 \) and \( P_3' \) are intervals \( P_3 = [p_i, p_j] \) and \( P_3' = [p'_i, p'_j] \), the requirement is that \( p_j \geq p'_j \).

E.4 Reasoning about Dependencies in Risk Graphs

When systems are mutually dependent, a threat toward one of them may realise threats toward the others [10, 11]. Rinaldi et al. [11] argue that mutually dependent infrastructures must be considered in a holistic manner. Within risk analysis, however, it is often not feasible to analyse all possible systems that affect the target of analysis as once. To remedy this, we may use a modular approach [2]. By modular risk analysis we refer to a process in which separate parts of a system is analysed independently, and in which there is support for combining separate analysis results into an overall picture for the whole system.

For this purpose we present in this section dependent risk graphs. Such a risk graph is divided into two parts, an assumption and a target. The assumption describes the assumptions on which the risk estimates of the target depend.

E.4.1 The Syntax of Dependent Risk Graphs

In the context of dependent risk graphs, we refer to risks graphs as defined in Section E.2 as basic risk graphs. For a basic risk graph \( D \) to be well-formed, we require that if a relation is contained in \( D \), then its source and destination vertices are also contained in \( D \):

\[
v \rightarrow v' \in D \Rightarrow v \in D \land v' \in D \quad (E.1)
\]

A dependent risk graph is similar to a basic risk graph, except that its elements (vertices and relations) are divided into two disjunct sets, \( A \) and \( T \). The former represents the elements of the assumption, and the latter represents the elements of the target. We denote a dependent risk graph by \( A \triangleright T \). For a dependent risk graph to be well-formed we have the following requirements:

\[
v \rightarrow v' \in A \Rightarrow v \in A \land v' \in A \cup T \quad (E.2)
\]

\[
v \rightarrow v' \in T \Rightarrow v' \in T \land v \in A \cup T \quad (E.3)
\]

\[
v \rightarrow v' \in A \cup T \Rightarrow v \in A \cup T \land v' \in A \cup T \quad (E.4)
\]

\[
A \cap T = \emptyset \quad (E.5)
\]

Note that (E.4) is implied by (E.2) and (E.3). This means that if \( A \triangleright T \) is a well-formed dependent risk graph, then \( A \cup T \) is a well-formed basic risk graph.
E.4.2 The Semantics of Dependent Risk Graphs

Before extending the semantics of basic risk graphs as defined in Section E.2, we define the notion of interface between sub-graphs. An interface is between sets of elements that may not fulfil the well-formedness requirement for basic risk graphs presented above.

Given two sub-graphs \( D \) and \( D' \), \( i(D, D') \) denotes the interface of \( D \) toward \( D' \). This interface is obtained from \( D \) by keeping only the vertices and relations that \( D' \) depends on directly.

\[
i(D, D') = \{ v \in D | \exists v' \in D' : v \rightarrow v' \in D \cup D' \} \cup \{ v \rightarrow v' \in D | v' \in D' \}
\]

A dependent risk graph \( A \lhd T \) means that all sub-graphs of \( T \) that only depends on the parts of \( A \)'s interface toward \( T \) that actually hold must also hold. Formally, the semantics is defined as follows:

\[
[A \lhd T] \overset{\text{def}}{=} \forall T' \subseteq T : [i(i(A \cup T \setminus T', T'))] \Rightarrow [T']
\]

Note that \( \setminus \) is assumed to bind stronger than \( \cup \) and \( \cap \). By the definition, if all of \( A \) holds ([\( A \)] is true), then all of \( T \) must also hold. Note also that the definition of the semantics of basic risk graphs applies to all sets of vertices and relations, irrespective of any well-formedness criteria.

Observe that if the assumption of a dependent risk graph \( A \lhd T \) is empty, i.e. \( A = \emptyset \), it means that we have the risk graph \( T \). In other words, the semantics of \( \emptyset \lhd T \) is the same as the semantics of \( T \).

E.4.3 Rules for Dependency Reasoning

In the following we define a set of rules for reasoning about dependencies. The calculus can be used, for example, to argue that an overall threat scenario captured by a dependent risk graph \( D \) follows from \( n \) dependent risk graphs \( D_1, \ldots, D_n \) describing mutually dependent sub-scenarios.

In order to proceed to the rules, we first precisely define what is meant by dependency. The relation \( D \dashv D' \) means that \( D' \) does not depend on any vertex or relation in \( D \). This means that \( D \) does not have any interface toward \( D' \), and \( D \) and \( D' \) have no common elements.

\[
D \dashv D' \overset{\text{def}}{=} D \cap D' = \emptyset \wedge i(D, D') = \emptyset
\]

Note that \( D \dashv D' \) does not imply \( D' \dashv D \).

**Rule E.4 (Assumption independence)** If we have deduced \( T \) assuming \( A \) and \( T \) is independent of \( A \), then we may deduce \( T \):

\[
A \lhd T \quad A \dashv T \quad \Rightarrow \quad T
\]

From the second premise it follows that \( T \) does not depend on any element in \( A \). Since the first premise states that all sub-graphs of \( T \) hold that depend on the parts of \( A \)'s interface toward \( T \) that hold, we may deduce \( T \).

**Rule E.5 (Assumption simplification)** Parts of the assumptions that are not connected to the rest can be removed:

\[
A \cup A' \lhd T \quad A \dashv A' \cup T \quad \Rightarrow \quad A' \lhd T
\]

The second premise implies that neither \( A' \) nor \( T \) depends on any element in \( A \). The validity of the first premise is therefore independent of \( A \), in which case also the conclusion is valid.

**Rule E.6 (Target simplification)** Parts of the target can be removed as long as they are not situated in-between the assumption and the part of the target that we wish to keep:

\[
A \lhd T \cup T' \quad T' \dashv T \quad \Rightarrow \quad A \lhd T
\]
Table E.2: Naming conventions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Diagram construct</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>Vertex before-after</td>
</tr>
<tr>
<td>vb</td>
<td>Vertex before</td>
</tr>
<tr>
<td>va</td>
<td>Vertex after</td>
</tr>
<tr>
<td>v</td>
<td>Vertex</td>
</tr>
<tr>
<td>v → v'</td>
<td>Relation before-after</td>
</tr>
<tr>
<td>v →b v'</td>
<td>Relation before</td>
</tr>
<tr>
<td>v →a v'</td>
<td>Relation after</td>
</tr>
</tbody>
</table>

The second premise implies that $T$ does not depend on any element in $T'$, and therefore does not depend on any element in $A$ via $T'$. Hence, the validity of the first premise implies the validity of the conclusion.

To make use of these rules when scenarios are composed, we also need a consequence rule.

**Rule E.7 (Assumption consequence)**

$$A \cup A' \triangleright T \quad \triangleright A' \triangleright T$$

By the latter rule, if all sub-graphs of $T$ hold that depend on the parts of the interface of $A \cup A'$ toward $T$ that hold, and we can show $A$, then it follows that $T$.

We refer to Section E.7 for examples of dependent diagrams in the form of CORAS instantiations.

### E.5 Risk Graphs for Changing Risks

For systems that change and evolve over time, also the risks change and evolve and should be analysed and documented as such. In order to support the modelling of changing risks we need to generalise risk graphs to allow the simultaneous modelling of risks both before and after the implementation of some given changes. For this purpose we extend the risk graph notation to three kinds of vertices and three kinds of relations, namely before, after and before-after. When an element (vertex or relation) is of kind before it represents risk information before the changes, when it is of kind after it represents risk information after the changes, and when it is of kind before-after it represents risk information that holds both before and after the changes.

#### E.5.1 The Syntax of Risk Graphs with Change

A risk graph with change is represented by a pair $(D_b, D_a)$ of sets of elements, the former consisting of the vertices and relations of kind before and the latter consisting of vertices and relations of kind after. Table E.2 gives an overview of the language constructs and the naming conventions we use for referring to them. The symbols written in sans serif and the arrows denote specific language constructs, whereas $v$ denotes an arbitrary vertex of any kind. Table E.3 gives an overview of the various ways of specifying likelihoods. Recall that any of the likelihoods can be undefined, in which case they are completely underspecified.

Figure E.3 shows an example of the visual representation of a risk graph with change. Solid lines, like on the vertex $v_1$ and the relation from $v_1$ to $v_3$, indicate elements that only exists before, while dashed lines indicates elements that exists after. The vertices with a white shadow, like $v_2$, are those that exist both before and after, while those with black shadows, like $v_5$, exist only after. The dashed relations with a single probability set, like the relation from $v_5$ to $v_6$, exist only after, while those with double probability sets, like the relation from $v_1$ to $v_4$, exist both before and after.

Since we are operating with vertices and relations of kind before-after as language element of their own, we also allow the representation of risk graphs with change as a single set $D$ of vertices and relations, where each element is of one of the kinds before, after or before-after. This single set of elements is then
Likelihood spec. | Interpretation
---|---
v(P P') | v occurs with likelihood P before, and v occurs with likelihood P' after
vb(P) | vb occurs with likelihood P before
va(P) | va occurs with likelihood P after
v −→ v' | v leads to v' with conditional likelihood P before, and v leads to v' with conditional likelihood P' after
v −→b v' | v leads to v' with conditional likelihood P before
v −→a v' | v leads to v' with conditional likelihood P after

Table E.3: Denoting likelihoods

![Risk graph for changing risks](image)

syntactic sugar for the equivalent representation of a pair of sets of elements. For such a combined representation D we use the functions before(\_\_) and after(\_\_) to filter the combined risk graph with respect to the elements of kind before and after, respectively. The following define the function before(\_\_) for singleton sets of elements.

\[
\text{before}(\{v(P, P')\}) = \{vb(P)\}
\]
\[
\text{before}(\{vb(P)\}) = \{vb(P)\}
\]
\[
\text{before}(\{va(P)\}) = \emptyset
\]
\[
\text{before}(\{v \rightarrow_{P'} v'\}) = \{v \rightarrow_{P_b} v'\}
\]
\[
\text{before}(\{v \rightarrow_{a} v'\}) = \emptyset
\]

The filtering of a risk graph with change D with respect to the before elements is then defined as

\[
\text{before}(D) = \bigcup_{e \in D} \text{before}(\{e\})
\]

The definition of the function after(\_\_) is symmetric. For a risk graph with change D of elements of the three different kinds, the representation as a pair of elements of kind before and elements of kind after is then given by (before(D), after(D)). Figure E.4 shows the graphical representation of the two risk graph before(D) (top) and after(D) (bottom) where D is the risk graph shown in Fig. E.3.

### E.5.2 The Semantics of Risk Graphs with Change

Given the syntax of risk graphs with change as defined above, we can define the semantics as a straightforward generalisation of the semantics of regular risk graphs as defined in Section E.2. The semantics \[[D_b, D_a]\] of a risk graph with change is defined as

\[
[[D_b, D_a]] = [D_b] \land [D_a]
\]
For a combined representation $D$ of a risk graph with change, the semantics is defined as

$$[D] = \text{before}(D), \text{after}(D)$$

### E.6 Relating Risk Model to Target Description

Risk analysis of changing systems requires means for identifying the parts of a risk picture that are affected by changes to a specific part of the target (and therefore need to be reassessed), as well as identifying the parts of the risk picture that are not affected (and therefore valid also after the changes). Thus we need techniques for identifying and documenting the relation between the target description and the risk models in a way that gives us traceability between target elements (elements of the target description) and risk model elements.

Two key artifacts in a risk analysis are the target model and the risk model. The target model is the core part of the overall target description and documents the events, scenarios and actors that are the subject to the risk analysis. Given these two artifacts we introduce a third artifact in order to establish the relation between the former two, namely a trace model.

The trace model can be represented in a table format that allows the tracing from target model elements to risk model elements, and vice versa. Initially we can think of the trace model as a set of pairs $(u_{id}, v_{id})$ of target model identifiers $u_{id}$ and risk model identifiers $v_{id}$ representing the rows of the table. This of course requires that each of the elements have a unique identifier. In the following we assume that we already have a target model and a risk model of elements with unique identifiers, since obtaining such models by indexing the elements is a trivial task.

From a pragmatic point of view, there are two obvious shortcomings of the table format given above. To make efficient use of the trace model it should convey information about the relations in an intuitive way; the use of possibly tool generated indexes for the model elements is not intuitively informative. Furthermore, in many cases several target model elements are logically understood as a whole. Without some means of grouping several rows of the table into one compound relation, such structures of the target model will be obscured.

To mitigate this we introduce a third column in the table for tagging the target model element/risk model element pairs. The grouping of pairs is then conducted by inserting the same tag on several rows. The name of the tag should be chosen by the end-user, and should be a unique name that conveys intuitive information about the grouping. More formally, the trace model is now a set of tuples $(u_{id}, v_{id}, t)$ of a target model identifier, a risk model identifier and a tag.

We extend the risk graph notation with a language construct for explicitly specifying the relation to the target model. The construct is used for annotating risk graphs with the tags of the trace model. We
understand this construct as a mere visualisation of the trace model in the risk graphs, and not as part of the semantics of risk graphs. An example of the visualisation of a trace model in a risk graph with change is shown in Fig. E.5. As with the other elements of risk models with change, the target model relations can be specified as existing before the change only, (for example $t_1$), after the change only (for example $t_3$), or both before and after the change (for example $t_2$).

E.7 Instantiation of CORAS

In a CORAS risk analysis, threat diagrams are used intensively to facilitate risk identification and risk estimation. The diagrams are furthermore used as a part of the documentation and reporting of the analysis results. In this section we explain and demonstrate how risk graphs can be instantiated in CORAS. In this way the semantics and calculi of risk graphs can be applied if CORAS is the chosen approach to risk management and risk analysis. As argued in Section E.2, risk graphs can also be instantiated in other concrete and well-established approaches.

Figure E.6 depicts an example of a threat diagram. In fact, this threat diagram shows the scenarios that are modelled by means of the risk graph in Fig. E.1. Generally, CORAS threat diagrams describe how threats may exploit vulnerabilities to initiate threat scenarios, how threat scenarios may lead to unwanted incidents or other threat scenarios, and the assets harmed by the unwanted incidents. The language constructs are threats (deliberate, accidental and non-human), vulnerabilities, threat scenarios, unwanted incidents and assets. Only threat scenarios and unwanted incidents may be assigned likelihoods.

There are furthermore three kinds of relations in threat diagrams, namely initiates relations, leads-to relations and impacts relations. An initiates relation has a threat as source and a threat scenario or unwanted incidents as target. It can be annotated with a likelihood that describes the likelihood for the threat to initiate the related scenario or incident. A leads-to relation has a threat scenario or unwanted incident as both source and target. It can be annotated with a conditional likelihood. An impacts relation has an unwanted incident as source and an asset as target, and can be annotated with a consequence value that describes the harm of the incident on the asset when the incident occurs.

While all scenarios and relations in Fig. E.1 are present in Fig. E.6, there are some significant differences between the two diagrams. The threat diagram explicitly shows the initiating threats, distinguishes vertex $v_7$ from the other scenarios as an unwanted incident, and explicitly shows the asset that is harmed.

The differences between threat diagrams and risk graphs are summarised as follows:

- Initiate relations and leads-to relations in threat diagrams can be annotated with vulnerabilities, while the relations in risk graphs cannot.
- Threat diagrams distinguish between four kinds of vertices, namely threats, threat scenarios, unwanted incidents and assets, while risk graphs only have scenarios.
Figure E.6: Instantiation of risk graphs in CORAS
• Threat diagrams distinguish between three kinds of relations, namely initiates relations, leads-to relations and impacts relations, while risk graphs only have leads-to relations.

Given the differences between threat diagrams and risk graphs, the techniques for reasoning about likelihoods nevertheless carry over to the CORAS instantiation. The vulnerabilities are mere annotations on relations, and can be ignored in the formal representation of the diagrams. Moreover, the various vertices and relations of threat diagrams can be interpreted as special instances of the risk graph vertex and relation:

• An unwanted incident of a threat diagram is interpreted as a scenario of a risk graph.

• A set of threats \( r_1, \ldots, r_n \) with initiates relations to the same threat scenario \( s \) is interpreted as follows: The threat scenario \( s \) is decomposed into \( n \) parts, where each resulting sub-scenario \( s_j, j \in \{1, \ldots, n\} \), corresponds to the part of \( s \) that is initiated by threat \( r_j \). The threat \( r_j \) with initiates relation of likelihood \( l_j \) to sub-scenario \( s_j \) is then combined into the risk graph scenario \( \text{Threat } r_j \text{ initiates } s_j \) and the scenario is assigned likelihood \( l_j \).

• An impacts relation from unwanted incident \( u \) to asset \( a \) with consequence \( c \) in a threat diagram is interpreted as follows: The impacts relation is interpreted as a risk graph relation with likelihood 1; the asset \( a \) is interpreted as the risk graph scenario \( \text{Incident } u \text{ harms asset } a \text{ with consequence } c \).

With this interpretation, we refer to Sect. E.3 for the techniques for reasoning about likelihoods in CORAS threat diagrams. However, notice that Rule E.1 (Relation) applies to the CORAS leads-to relations only and that Rule E.2 (Mutually exclusive vertices) and Rule E.3 (Independent vertices) apply to the CORAS threat scenarios and unwanted incidents. In order to allow all likelihood reasoning to be conducted directly in CORAS diagrams, we introduce a separate rule for the initiates relation. We let \( r \) denote a threat, \( v \) denote a vertex (threat scenario or unwanted incident), \( r \rightarrow v \) denote the initiates relation from threat \( r \) to vertex \( v \), and \( r \sqcap v \) denote the occurrences of vertex \( v \) that are initiated by the threat \( r \).

**Rule E.8 (Initiates)** If there is an initiates relation from threat \( r \) to vertex \( v \), we have:

\[
\frac{r \rightarrow v}{(r \sqcap v)(P)}
\]

### E.7.1 Dependent CORAS

Dependent CORAS is the instantiation of dependent risk graphs in CORAS. By the same principle of dependent risk graphs, a dependent CORAS diagram is a set of elements \( D \) that is divided into two disjunct subsets \( A \) and \( T \) representing the assumption and the target, respectively.

In order to support the specification of assumptions in CORAS threat diagram, the syntax is extended with the language construct of a border that is depicted as a rounded rectangle as depicted in Fig. E.7. Everything inside the border belongs to the target; everything on the border, like the leads-to relation from threat scenario \( v_2 \) to unwanted incident \( v_4 \) also belongs to the target. The remaining elements, i.e. everything completely outside the border, belongs to the assumptions.

With respect to the border, the following constraints apply:

• Relations cannot cross the border line from the inside to the outside.

• Assets can occur only on the inside of the border line.

• Impacts relations can occur only on the inside of the border line.

• Threats cannot occur on the border line. They may, however, occur both on the inside and on the outside of the border line.

• Initiates relations cannot cross the border line. This means that initiates relations are either fully inside or fully outside the border line.
In Fig. E.7, for example, threat $r_1$, vulnerability $vn_1$ and threat scenario $v_1$ belong to the target and do not depend on any of the assumptions. The occurrence of unwanted incident $v_4$ depends not only on these, but also the assumption. Hence, the validity of the likelihood estimate $P_4$ depends on the assumptions about the likelihoods of $v_2$ and $v_3$.

Assumptions are something we take as granted or accept as true in a risk analysis. In some cases we make assumptions that do not hold in the general case, for example that the power supply or network connection is stable; consequently, this yields analysis results that are valid only under these assumptions. In other cases we make assumptions about the environment of the target on which the target depends. With the modular approach to risk analysis that is supported by dependent CORAS, parts of the environment may then serve as the target in a separate analysis. By using the calculus for reasoning about dependencies, the results of the latter can serve as premises in the reasoning about the former.

As an illustrative example, consider the dependent threat diagram of Fig. E.8. Also this diagram models the unwanted incident $v_4$. However, this time the threat scenario $v_2$ is part of the target, while $v_1$ is part of the assumptions. Common to the two dependent diagrams is the part of the assumption that consist of the threat $r_3$, the vulnerability $vn_3$ and the threat scenario $v_3$.

By using the rules for reasoning about dependencies, the two separate analysis results can be com-

Figure E.7: Dependent CORAS diagram

Figure E.8: Dependent CORAS diagram with different assumption and target
bined into one. The result is shown by the dependent CORAS diagram in Fig. E.9. The elements that serve as assumptions in both analyses remain as assumptions in the combined diagram. The remaining parts can be included as part of the target.

**E.7.2 CORAS with Change**

The instantiation of risk graphs with change in CORAS yields a generalisation of CORAS diagrams with the expressiveness to model risks that occur before changes only, risk that occur after changes only, and risk that occur both before and after changes. Moreover, for the risks that occur both before and after, the diagrams can express how risks evolve by changes to likelihoods and consequences. This generalisation is by the same principles as the generalisation of risk graphs in Section E.5.

The CORAS threat diagram in Fig. E.10 exemplifies the instantiation. In particular, this diagram shows the modelling in CORAS of the scenarios that are modelled as a risk graph with change in Fig. E.5. In the same way as for risk graphs, a CORAS diagram $D$ with change can be understood as syntactic sugar for a pair of CORAS diagrams $(D_b, D_a)$, representing the risk picture before and after changes, respectively.

**E.8 Conclusions**

In this paper we have introduced risk graphs as a technique for the identification and documentation of risk to support risk analysis in particular and risk management in general. Specifically, risk graphs support the structuring of events and scenarios that lead to unwanted incidents, and they support the estimation and reasoning about likelihoods.

Risk graphs model risks at a quite abstract level and are not really intended for being used in the real and practical setting of risk identification and risk management. However, risk graphs can be understood as a common abstraction of several concrete risk modelling languages that may serve as the chosen risk modelling technique in specific risk analyses. Consequently, the formal foundation and rules for reasoning about risk graphs that we have presented carry over to existing risk modelling techniques such as fault trees, event trees, attack trees, cause-consequence diagrams, Bayesian networks and CORAS threat diagrams. The formalisation of risk graphs and their calculi therefore provides a formal foundation for risk management with a wide flexibility for the end-user in terms of which concrete approach to choose.

The formal foundation for risk management presented in this paper includes a formal semantics of risk graphs in terms of a probability space on traces of events. A set of rules for reasoning about likelihoods have been presented, the soundness of which can be shown with respect to the semantics. Moreover, the guidelines for consistency checking likelihood estimates assigned to diagram elements are based on the likelihood calculus.
Figure E.10: Instantiation of risk graphs with change in CORAS
Explicitly modelling and reasoning about assumptions and dependencies in risk analyses is supported by dependent risk graphs. Furthermore, dependent risk graphs supports a modular approach to risk modelling and analysis in which separate parts of a complex target of analysis with mutual dependencies can be analysed separately. The reasoning about dependencies is supported by the presented calculus.

Finally, this paper has presented a generalisation of risk graphs to the setting of changing and evolving systems with the support for modelling and reasoning about changing and evolving risks.

References


