Secure and Trustworthy Composite Services

Title: Consolidation of existing work

Lead beneficiary (partner):
Tecnalia

Internally reviewed by (name/partner):
Dr. Ilias Spais/ATC
Jimmy McGibney/TSSG

Approved by:
Executive board

Abstract:
Aniketos is about establishing and maintaining trustworthiness and secure behaviour in a constantly changing service environment. The project aligns existing and develops new technology, methods, tools and security services that support the design-time creation and run-time dynamic behaviour of composite services, addressing service developers, service providers and service end users.

Therefore, the Aniketos project covers many research areas, and there are differences between the state of the art and real implementations in the market. This deliverable collects technologies used by industry and unifies this knowledge into a single document, providing a starting point for the project. The goal is to have a faster integration of technologies and a holistic view by all the consortium of the platform. The deliverable thus serves to reduce overlap of knowledge and increase reuse of past projects' results.

<table>
<thead>
<tr>
<th>Dissemination level</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
<td>Public</td>
</tr>
<tr>
<td>CO</td>
<td>Confidential, only for members of the consortium (including Commission Services)</td>
</tr>
</tbody>
</table>

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 257930.
Aniketos consortium

Aniketos (Contract No. FP7-257930) is an Integrated Project (IP) within the 7th Framework Programme, Call 5, Priority 1.4 (Trustworthy ICT). The consortium members are:

**SINTEF ICT**
NO-7465 Trondheim
Norway
www.sintef.com

**Project manager:** Richard T. Sanders
richard.sanders@sintef.no
+47 73 59 30 06

**Technical manager:** Per Håkon Meland
per.h.meland@sintef.no +47 73 59 29 41

**Tecnalia Research & Innovation**
(TECNALIA)
E-48160 Derio
Bizkaia, Spain
www.tecnalia.com/en

Contact: Erkuden Rios Velasco
erkuden.rios@tecnalia.com

**Consiglio Nazionale delle Ricerche**
(CNR)
00185 Roma, Italy
www.cnr.it

Contact: Fabio Martinelli
Fabio.Martinelli@iit.cnr.it

**Thales Services SAS (THALES)**
78140 Velizy-Villacoublay, France
www.thalesgroup.com

Contact: Edith Félix
edith.felix@thalesgroup.com

**Liverpool John Moores University**
(LJMU)
Liverpool, L3 5UX, United Kingdom
www.ljmu.ac.uk/cmp

Contact: Madjid Merabti
m.merabti@ljmu.ac.uk

**Elsag Datamat S.P.A. (ELSAG)**
16154 Genova, Italy
www.elsagdatamat.com

Contact: Pucci Paolo
Paolo.Pucci@elsagdatamat.com

**SEARCH-LAB Ltd. (SEARCH)**
Budapest 1117, Hungary
www.search-lab.hu

Contact: Zoltán Hornák
zoltan.hornak@search-lab.hu

**Atos Origin (ATOS)**
28037 Madrid, Spain
www.atc.gr

Contact: Pedro Soria-Rodriguez
pedro.soria@atosresearch.eu
# Table of contents

Aniketos consortium........................................................................................................................................ iii  
Table of contents ........................................................................................................................................ v  
List of figures ............................................................................................................................................... ix  
List of tables ............................................................................................................................................... ix  
Executive summary ...................................................................................................................................... 1  
1 Introduction ........................................................................................................................................... 3  
1.1 Aniketos motivation and background .................................................................................................. 3  
1.2 Summary ........................................................................................................................................... 3  
1.3 Structure of this document ...................................................................................................................... 4  
1.4 Relationships with other deliverables .................................................................................................. 4  
1.5 Contributors ...................................................................................................................................... 5  
1.6 Acronyms and abbreviations .................................................................................................................. 5  
2 Overview of the Aniketos platform........................................................................................................... 6  
3 Classification of works .............................................................................................................................. 8  
4 Aniketos platform modules ......................................................................................................................... 12  
4.1 Socio-technical security modelling tool ................................................................................................... 12  
4.1.1 SeaMonster – Security Modelling Software (SINTEF)....................................................................... 12  
4.1.2 TECNALIA Security Framework (TECNALIA) ............................................................................... 14  
4.1.3 Modelling of Evolutionary Requirements (UNITN) ....................................................................... 15  
4.1.4 MASTER Design Toolkit (UNITN) ................................................................................................ 16  
4.1.5 Goal Risk Model (UNITN) .......................................................................................................... 17  
4.1.6 Secure Tropos (UNITN) .............................................................................................................. 18  
4.1.7 Si* tool (UNITN) ............................................................................................................................ 19  
4.1.8 Organizational S&D Patterns (UNITN) .......................................................................................... 20  
4.1.9 Contextual Requirements Modelling and Analysis (UNITN) ............................................................ 22  
4.1.10 Modelling Interaction between Agents via Commitments (UNITN) ................................................. 23  
4.1.11 Thales Risk Assessment Domain Specific Modelling Language (THALES) ..................................... 24  
4.1.12 Thales SOA Modelling Suite (THALES) ...................................................................................... 25  
4.1.13 Property and Policy Tools (LJMU) ............................................................................................... 26  
4.1.14 Other related work to Socio-technical security modelling tool ..................................................... 27  
4.2 Model transformation module ................................................................................................................ 29  
4.2.1 Related work to Model transformation module ................................................................................ 29  
4.3 Trustworthiness prediction module ...................................................................................................... 30  
4.3.1 Role-based Trust Management with quantitative notion of trust (CNR) .............................................. 30  
4.3.2 Trust Manager (TSSG) ............................................................................................................... 31  
4.3.3 STACS: Scalability Testbed for Access Control performance of Services (TSSG) ......................... 31  
4.3.4 Other related work to Trustworthiness prediction module .............................................................. 31  
4.4 Monitor trustworthiness module ........................................................................................................... 32  
4.4.1 Role-based Trust Management with quantitative notion of trust (CNR) .............................................. 32  
4.4.2 xESB (UNITN) ............................................................................................................................. 32  
4.4.3 Trust Manager (TSSG) ............................................................................................................... 33  
4.4.4 Other related work to Monitor trustworthiness module .................................................................... 33  
4.5 Contract negotiation module .................................................................................................................. 34  
4.5.1 MUSIC studio and middleware (SINTEF) ....................................................................................... 34  
4.5.2 Security by Contract with Trust for mobile systems (CNR) .............................................................. 34  
4.5.3 SLA Management framework (ATOS) ........................................................................................... 36  
4.5.4 STACS: Scalability Testbed for Access Control performance of Services (TSSG) ......................... 37  
4.5.5 Other related work to Contract negotiation module .......................................................................... 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6</td>
<td>Security property determination module.</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Modular plans for secure service composition (CNR)</td>
</tr>
<tr>
<td>4.6.3</td>
<td>Property and Policy Tools (LJMU)</td>
</tr>
<tr>
<td>4.6.4</td>
<td>Platform for Run-time reconfigurability of security (ATOS)</td>
</tr>
<tr>
<td>4.6.5</td>
<td>STACS: Scalability Testbed for Access Control performance of Services (TSSG)</td>
</tr>
<tr>
<td>4.6.6</td>
<td>Universal Service Description Language (SAP)</td>
</tr>
<tr>
<td>4.6.7</td>
<td>Avantssar platform for validating trust and security properties (SAP)</td>
</tr>
<tr>
<td>4.6.8</td>
<td>Other related work to Security property determination module</td>
</tr>
<tr>
<td>4.7</td>
<td>Security verification module</td>
</tr>
<tr>
<td>4.7.1</td>
<td>Usage Control Authorization System (CNR)</td>
</tr>
<tr>
<td>4.7.2</td>
<td>MATTs (Mobile Agent Topology Test System) (LJMU)</td>
</tr>
<tr>
<td>4.7.3</td>
<td>Property and Policy Tools (LJMU)</td>
</tr>
<tr>
<td>4.7.4</td>
<td>MASTER Real-time Compliance Manager for SOA (ATOS)</td>
</tr>
<tr>
<td>4.7.5</td>
<td>Avantssar platform for validating trust and security properties (SAP)</td>
</tr>
<tr>
<td>4.7.6</td>
<td>Rodin Platform (SAP)</td>
</tr>
<tr>
<td>4.7.7</td>
<td>STACS: Scalability Testbed for Access Control performance of Services (TSSG)</td>
</tr>
<tr>
<td>4.7.8</td>
<td>Other related work to security verification module</td>
</tr>
<tr>
<td>4.8</td>
<td>Security policy monitoring module</td>
</tr>
<tr>
<td>4.8.1</td>
<td>Risk-aware Usage Decision Making in Highly Dynamic Systems (CNR)</td>
</tr>
<tr>
<td>4.8.2</td>
<td>MATTs (Mobile Agent Topology Test System) (LJMU)</td>
</tr>
<tr>
<td>4.8.3</td>
<td>Platform for Run-time reconfigurability of security (ATOS)</td>
</tr>
<tr>
<td>4.8.4</td>
<td>Model Driven Security/OpenPMF</td>
</tr>
<tr>
<td>4.8.5</td>
<td>Other related work to Security policy monitoring module</td>
</tr>
<tr>
<td>4.9</td>
<td>Threat response recommendation module</td>
</tr>
<tr>
<td>4.9.1</td>
<td>MUSIC studio and middleware (SINTEF)</td>
</tr>
<tr>
<td>4.9.2</td>
<td>SeaMonster – Security Modelling Software (SINTEF)</td>
</tr>
<tr>
<td>4.9.3</td>
<td>Risk-aware Usage Decision Making in Highly Dynamic Systems (CNR)</td>
</tr>
<tr>
<td>4.9.4</td>
<td>Goal Risk Model (UNITN)</td>
</tr>
<tr>
<td>4.9.5</td>
<td>Architecture for Self-Adaptive Socio-Technical Systems (UNITN)</td>
</tr>
<tr>
<td>4.9.6</td>
<td>Other related work to Threat response recommendation</td>
</tr>
<tr>
<td>4.10</td>
<td>Service threat monitoring module</td>
</tr>
<tr>
<td>4.10.1</td>
<td>Platform for Run-time reconfigurability of security (ATOS)</td>
</tr>
<tr>
<td>4.10.2</td>
<td>MASTER Design Toolkit (UNITN)</td>
</tr>
<tr>
<td>4.10.3</td>
<td>Other related work to threat response recommendation</td>
</tr>
<tr>
<td>4.11</td>
<td>Notification module</td>
</tr>
<tr>
<td>4.11.1</td>
<td>MATTs (Mobile Agent Topology Test System) (LJMU)</td>
</tr>
<tr>
<td>4.11.2</td>
<td>Platform for Run-time reconfigurability of security (ATOS)</td>
</tr>
<tr>
<td>4.11.3</td>
<td>Other related work to Notification module</td>
</tr>
<tr>
<td>4.12</td>
<td>Community support module</td>
</tr>
<tr>
<td>4.12.1</td>
<td>Related work to Community support module</td>
</tr>
<tr>
<td>4.13</td>
<td>Threat repository module</td>
</tr>
<tr>
<td>4.13.1</td>
<td>Security Vulnerability Repository Service (SEARCH)</td>
</tr>
<tr>
<td>4.13.2</td>
<td>MASTER Design Toolkit (UNITN)</td>
</tr>
<tr>
<td>4.13.3</td>
<td>Other related work to Threat repository module</td>
</tr>
<tr>
<td>4.14</td>
<td>Marketplace</td>
</tr>
<tr>
<td>4.14.1</td>
<td>Composition Repository (ATC)</td>
</tr>
<tr>
<td>4.14.2</td>
<td>Other related work to Marketplace</td>
</tr>
<tr>
<td>4.15</td>
<td>Training material module</td>
</tr>
<tr>
<td>4.15.1</td>
<td>Automated Training Material Generator (SEARCH)</td>
</tr>
<tr>
<td>4.15.2</td>
<td>Other related work to Training material module</td>
</tr>
</tbody>
</table>
D1.1: Consolidation of existing work

5 Environment modules ........................................................................................................................................... 67
   5.1 Integrated environments .......................................................................................................................... 67
   5.1.1 Oracle SOA Suite ................................................................................................................................ 67
   5.1.2 WSO2 SOA Platform .......................................................................................................................... 68
   5.1.3 SAP Enterprise Services Architecture ................................................................................................. 69
   5.2 Service composition framework ............................................................................................................ 70
   5.2.1 Service specification/planning mechanism .......................................................................................... 70
   5.2.2 Service validation mechanism ............................................................................................................ 79
   5.2.3 Service discovery mechanism ............................................................................................................ 79
   5.3 Service runtime environment ................................................................................................................ 79
   5.3.1 Service execution mechanism ............................................................................................................ 79
   5.3.2 Service adaptation mechanism ........................................................................................................ 87
   5.3.3 Service recomposition mechanism .................................................................................................. 88
   5.3.4 Service monitoring module .............................................................................................................. 88
   5.3.5 Context sensor .................................................................................................................................... 89
   5.4 Service registry ....................................................................................................................................... 89
   5.4.1 Composition Repository (ATC) ......................................................................................................... 89
   5.4.2 Other related work to Service registry .............................................................................................. 90
   5.5 Aniketos compliant service ...................................................................................................................... 90
   5.6 Identity management service .................................................................................................................. 91
   5.6.1 ED DIAMETER AAA (ELSAG) ......................................................................................................... 91
   5.6.2 Other related work to Identity management service .......................................................................... 92
   6 Usability and acceptance of the platform .................................................................................................. 94
   6.1 User-centred Design and Interaction Design (PLUS, DBL) ................................................................. 94
   6.2 Other related work to usability and acceptance .................................................................................. 98
   7 Background for the case studies .............................................................................................................. 99
   7.1 Case study A: “Future telecommunication services” .............................................................................. 99
   7.1.1 Italtel VoIP Suite (ITALTEL) ........................................................................................................... 99
   7.1.2 WindChester (WIND) ...................................................................................................................... 100
   7.2 Case study B: “The emerging European Air Traffic Management systems” ................................ 101
   7.2.1 Operational Validation (DBL) ......................................................................................................... 101
   7.2.2 Air Traffic Management (ATM) domain expertise (DBL) .............................................................. 104
   7.3 Case study C: “Land-buying and eGovernance” ................................................................................... 106
   7.3.1 Poleod (DAEM) .................................................................................................................................. 106
   7.3.2 OPS (Integrated Information System) of Municipality of Athens (DAEM) .................................. 107
   8 Recommendations for usability and acceptance of the platform ............................................................. 108
   9 Conclusion .................................................................................................................................................. 109

References ......................................................................................................................................................... 110

Appendix A. More detailed descriptions of works .......................................................................................... 125

1 Socio-technical security modelling tool ........................................................................................................ 125
   1.1 SeaMonster (SINTEF) .......................................................................................................................... 125
   1.2 TECNALIA Security Framework (TECNALIA) .................................................................................. 128
   1.3 Modelling Evolutionary Requirements (UNITN) ............................................................................... 132
   1.4 Goal Risk Model (UNITN) ................................................................................................................... 134
   1.5 Secure Tropos (UNITN) ....................................................................................................................... 136
   1.6 Si* tool (UNITN) .................................................................................................................................... 141
   1.7 Organizational S&D Patterns (UNITN) ................................................................................................. 143
   1.8 Contextual Requirements Modelling and Analysis (UNITN) .............................................................. 145
   1.9 Modelling Interaction between Agents via Commitments (UNITN) .................................................... 147
   1.10 Thales Risk Assessment Domain Specific Modelling Language (THALES) .................................... 150
   1.11 Thales SOA Modelling Suite (SMS) ................................................................................................... 152
   1.12 Property and Policy Tools (LJMU) ...................................................................................................... 152
D1.1: Consolidation of existing work

2 Trustworthiness prediction module ................................................................. 156
2.1 Role-based Trust Management with quantitative notion of trust (CNR) ............ 156
3 Monitor trustworthiness module .................................................................... 159
3.1 xESB (UNITN) ............................................................................................ 159
3.2 Trust Manager (TSSG) ................................................................................ 160
4 Contract negotiation module ............................................................................ 161
4.1 Security by Contract with Trust for mobile systems (CNR) ......................... 161
4.2 SLA Management framework (ATOS) ........................................................ 164
4.3 Security-Oriented Service Composition and Evolution ............................... 165
5 Security property determination module ....................................................... 168
5.1 Modular plans for secure service composition (CNR) ................................ 168
5.2 A Framework for Quantitative Security Analysis of Complex Business Systems (CNR) 170
5.3 Platform for Run-time reconfigurability of security (ATOS) ..................... 174
5.4 STACS / Scalability Testbed for Access Control performance of Services (TSSG) 176
5.5 Composing Security-Aware Software ....................................................... 177
5.6 Enforcing Secure Service Composition .................................................... 179
6 Security verification module .......................................................................... 181
6.1 Usage Control Authorization System (CNR) .............................................. 181
6.2 MATTs (Mobile Agent Topology Test System) (LJMU) .............................. 183
6.3 Avantssar platform for validating trust and security properties (SAP) ............ 188
6.4 Static Verification Framework Tool ............................................................. 189
7 Security policy monitoring module ............................................................ 191
7.1 Risk-aware Usage Decision Making in Highly Dynamic Systems (CNR) .... 191
7.2 Model Driven Security/OpenPMF .............................................................. 195
8 Threat response recommendation module .................................................. 199
8.1 Architecture for Self-Adaptive Socio-Technical Systems (UNITN) ............... 199
9 Service threat monitoring module .................................................................. 205
9.1 MASTER Design Toolkit (UNITN) ............................................................. 205
10 Community support module ....................................................................... 208
10.1 OAuth ...................................................................................................... 208
11 Threat repository module ............................................................................. 210
11.1 Security Vulnerability Repository Service (SEARCH) ............................... 210
12 Marketplace ............................................................................................... 213
12.1 Composition Repository (ATC) ............................................................... 213
13 Training material module ............................................................................ 214
13.1 Automated Training Material Generator (SEARCH) ............................... 214
14 Service composition Framework .................................................................. 215
14.1 Internet of Services: USDL/SDF .............................................................. 215
15 Service Runtime Environment .................................................................... 216
15.1 MUSIC studio and middleware (SINTEF) ............................................... 216
15.2 Dynamic and Static Aspect Weaving ..................................................... 219
16 Identity management service ...................................................................... 221
16.1 ED DIAMETER AAA (ELSAG) .............................................................. 221
17 Usability and acceptance of the platform .................................................. 224
17.1 User-centred Design and Interaction Design (PLUS, DBL) ....................... 224
18 Case Studies ............................................................................................... 227
18.1 Italtel VoIP Suite (ITALTEL) ................................................................. 227
18.2 WindChester (WIND) ............................................................................. 229
18.3 Poleod (DAEM) ..................................................................................... 234
18.4 OPS (Integrated Information System) of Municipality of Athens (DAEM) .... 235
List of figures

Figure 1. Goal: establish and maintain security and trustworthiness in composite services .................. 3
Figure 2. Overview of the Aniketos Platform. ................................................................................. 6
Figure 3. Allocation of existing work within Aniketos Platform and Environment components ...... 11
Figure 4. Example Policy file........................................................................................................... 27
Figure 5. High-level view of how products in Oracle SOA Suite work together (Source: Oracle [190]) .................................................................................................................................................. 68
Figure 6. WSO2 Enterprise middleware. (Source: WSO2 [192])................................................................................. 69
Figure 7. Overall SOA4All architecture. (Source: SOA4All Studio [206])............................................ 77
Figure 8. NEXOF Reference Architecture. (Source: NEXOF-RA project [215])................................. 78
Figure 9. Mule ESB architecture....................................................................................................... 84
Figure 10. WSO2 WSF for PHP Architecture ................................................................................... 86
Figure 11. Misuse case model excerpt of a model for an online travel agency service. ....................... 126
Figure 12. Attack tree excerpt of a network man-in-the-middle attack.............................................. 126
Figure 13. The Warehouse data management example service............................................................ 127
Figure 14. WS-SecurityPolicy typical scenario................................................................................... 128
Figure 15. High level design of the tool ............................................................................................. 129
Figure 16. Policy design workflow .................................................................................................. 130
Figure 17. Example of a security policy edition (i) ........................................................................... 130
Figure 18. Example of a security policy edition (ii). ......................................................................... 131
Figure 19. Example of a security policy edition (iii) ........................................................................ 131
Figure 20. Example of a security policy edition (iv) ........................................................................ 132
Figure 21. A continuous process to maintain the requirements ............................................................ 133
Figure 22. An example of application of GRM .................................................................................. 135
Figure 23. Basic elements of Secure Tropos metamodel .................................................................. 136
Figure 24. Trust and supervision relations in Secure Tropos ............................................................. 137
Figure 25. Dependencies and delegations in Secure Tropos ................................................................ 137
Figure 26. Secure Tropos methodology overview ........................................................................... 137
Figure 27. Requirements Acquisition within Secure Tropos methodology ......................................... 138
Figure 28. Basic properties list .......................................................................................................... 139
Figure 29. Example model using Secure Tropos .............................................................................. 140
Figure 30. Goal execution delegation diagram .................................................................................. 142
Figure 31. Goal execution delegation result diagram ........................................................................... 142
Figure 32. Access control pattern .................................................................................................... 144
Figure 33. An example of the context analysis .................................................................................. 146
Figure 34. An example model of a service oriented setting .................................................................. 149
Figure 35. An example of goal and sub-goals.................................................................................... 149
Figure 36. Risk management process followed by the tool ................................................................. 151
Figure 37. An example risk analysis model made with the tool ............................................................ 152
Figure 38. Property Interface for defining service properties ............................................................. 154
Figure 39. Example Property Set file .................................................................................................. 154
Figure 40. CompositionClient property selection interface ................................................................. 155
Figure 41. Example property definition file ....................................................................................... 156
Figure 42. Trust Manager .................................................................................................................. 161
Figure 43. Enforcement of security informed by trust ....................................................................... 162
Figure 44. The BEinGRID SLA Life Cycle ......................................................................................... 164
Figure 45. The BEinGRID SLA components..................................................................................... 165
Figure 46. Framework for security-oriented system composition and evolution................................. 166
Figure 47. A simple example of defining and specifying security objectives and properties ............. 167
Figure 48. An example of a service network for travel booking ............................................................ 169
List of tables

Table 1 Descriptions of components in the Aniketos platform ............................................................... 8
Table 2 Descriptions of environment components .................................................................................. 10
Table 3 Decision matrix with costs ..................................................................................................... 192
Table 4 Example of the decision matrix .............................................................................................. 194
Executive summary

As the Future Internet will need to be more responsive to threats and changes in services, the main objective of Aniketos is to help establish and maintain trustworthiness and secure behaviour in a constantly changing service environment.

The main result of the project will be the Aniketos platform that complements existing state-of-the-art Service Oriented Architecture (SOA) frameworks by connecting emerging technological solutions with the human practices that are needed to create and maintain secure and trusted composite services.

The platform provides support in three main areas:

- **design-time support** that consists of methodologies and tools that define and evaluate trustworthiness and risk-based security properties over and between external service components.

- **runtime support** to monitor and evaluate the trustworthiness and security violations of service components, also considering contextual information such as change in operation conditions and users’ behaviour.

- **community support** that includes Aniketos reference architecture and design patterns, example services, demonstration material, tutorials, development patterns and guidelines.

The purpose of this deliverable is to provide an integrated vision and alignment of the existing techniques (methods) and tools, in those three areas, that are candidates for development of the Aniketos platform to be based upon. The document also provides a set of recommendations to assist when choosing between the candidates in order to ensure usability and acceptance of the platform.

Note that this is the only release of this deliverable. As this deliverable describes our initial collection of background technology and existing work, we will update throughout the project an internal version where we will add further entries as new technology emerge and we perform additional analysis.

Other results of the consolidation task in Aniketos, such as the common glossary of core terms for the project, are planned to be delivered within D1.2 in month 12 (July 2011).
1 Introduction

1.1 Aniketos motivation and background

The Future Internet will provide an environment in which a diverse range of services are offered by a diverse range of suppliers, and users are likely to unknowingly invoke underlying services in a dynamic and ad hoc manner. Moving from today’s static services, we will see service consumers that transparently mix and match service components depending on service availability, quality, price and security attributes. Thus, the applications end users see may be composed of multiple services from many different providers, and the end user may have little in the way of guarantee that a particular service or service supplier will actually offer the security claimed.

Aniketos is about establishing and maintaining trustworthiness and secure behaviour in a constantly changing service environment. The project aligns existing and develops new technology, methods, tools and security services that support the design-time creation and run-time dynamic behaviour of composite services, addressing service developers, service providers and service end users. Aniketos provides methods for analysing, solving, and sharing information on how new threats and vulnerabilities can be mitigated. The project constructs a platform for creating and maintaining secure and trusted composite services. Specifications, best practices, standards and certification work related to security and trust of composite services are promoted for inclusion in European reference architectures. Our approach to achieving trustworthiness and security of adaptive services takes account of socio-technical aspects as well as basic technical issues.

1.2 Summary

This document corresponds to the deliverable D1.1 Consolidation of existing work and its role is to provide a consolidated description of the existing work related to the technologies and tools within the scope of Aniketos.

The document serves to share between Aniketos partners (who have participated in a number of previous related projects, both local and at European level) the background brought into the project by
D1.1: Consolidation of existing work

Each of them, as well as to analyze other existing relevant work from previous and on-going projects. The deliverable thus helps to have less overlap of knowledge and more reuse of past projects' results. The deliverable consolidates all the inputs in order to have an integrated vision of the relevant available work. This consolidation is aimed at a faster start of the selection of the baseline techniques and tools that will be the basis for the technical work in specific work packages. This will lead to a faster integration of technologies within the platform.

The work has resulted in a collection of descriptions of existing techniques (methods) and tools that are candidates for development of the Aniketos platform to be based upon. The descriptions include the initial thoughts on how the techniques and tools could be made part of the platform or its environment. The document also provides a set of recommendations to assist when choosing between the candidates in order to ensure usability and acceptance of the platform.

Note that this is the only release of this deliverable. As this deliverable describes our initial collection of background technology and existing work, we will update throughout the project an internal version where we will add further entries as new technology emerge and we perform additional analysis.

Other results of the consolidation task in Aniketos, such as the common glossary of core terms for the project, are planned to be delivered within D1.2 in month 12 (July 2011).

The intended readers of this document are software developers and security engineers interested in the development of trustworthy and secure services. They will profit from the integrated collection of techniques and tools descriptions gathered herein and will have a clearer idea of what the Aniketos project will work on. The document is specially intended for those developers participating in the Aniketos project, because they will be the primary beneficiaries, as they can use it as a reference document for a holistic view of the platform and for a quick guide to the available techniques and tools.

1.3 Structure of this document

The remainder of this document is organised as follows:

- Section 2 briefly explains the Aniketos platform and the areas of research to develop it.
- Section 3 explains the classification made of the work and how they are described.
- Section 4 provides the descriptions of the work related to the components within the Aniketos platform.
- Section 5 provides the descriptions of the work related to the Environmental components, i.e. components that can be considered to be outside the Aniketos platform.
- In Section 6, we describe other existing work relevant to ensuring usability and user acceptance of the Aniketos platform.
- Section 7 describes the consortium background useful for the case studies.
- Section 8 describes initial thoughts and recommendations for the selection of the baseline techniques and tools in order to guarantee the usability and acceptance of the Aniketos platform.
- Finally, Section 9 provides some conclusions for the document.

Additionally, where this is considered to be useful, we have included in Appendix A more detailed descriptions for the work described in Section 2, including a detailed description of basic operation and a usage example.

1.4 Relationships with other deliverables

This project deliverable (D1.1) relates to the following deliverables that are currently under construction:

- D1.2 First Aniketos architecture and requirements specification: Initial overall architecture of Aniketos platform together with the usage scenarios and requirements (month 12, July 2011). The...
D1.1: Consolidation of existing work

The classification of the existing work presented in this document has been made according to the current component view of the architecture that is under development in D1.2.

- D6.1 Initial analysis of the industrial case studies: Description of the scope and objectives of the industrial case studies, and the specific requirements for the industrial realisations in terms of scenario descriptions, technology domains and development processes (month 10, May 2011). The description of existing knowledge and background specific to each use case and recommendations and/or decisions on the baseline for each of them can be found in D6.1, and not in this document.

1.5 Contributors

All Aniketos consortium partners have contributed to this deliverable.

1.6 Acronyms and abbreviations

Following we provide a list of the most significant acronyms and abbreviations used along the document.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>Authentication, Authorization &amp; Accounting</td>
</tr>
<tr>
<td>ATM</td>
<td>Air Traffic Management</td>
</tr>
<tr>
<td>CS</td>
<td>Community support</td>
</tr>
<tr>
<td>DS</td>
<td>Design time support</td>
</tr>
<tr>
<td>DSL</td>
<td>Domain Specific Language</td>
</tr>
<tr>
<td>ESB</td>
<td>Enterprise Service Bus</td>
</tr>
<tr>
<td>MDD</td>
<td>Model-Driven Development</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure-as-a-Service</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IoS</td>
<td>Internet of Services</td>
</tr>
<tr>
<td>PEP</td>
<td>Policy Enforcement Point</td>
</tr>
<tr>
<td>RS</td>
<td>Run time support</td>
</tr>
<tr>
<td>S&amp;D</td>
<td>Security and Dependability</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software-as-a-Service</td>
</tr>
<tr>
<td>SDF</td>
<td>Service Delivery Framework</td>
</tr>
<tr>
<td>SME</td>
<td>Small Medium Enterprise</td>
</tr>
<tr>
<td>SOA</td>
<td>Service-oriented architecture</td>
</tr>
<tr>
<td>SoaML</td>
<td>Service-oriented architecture Modelling Language</td>
</tr>
<tr>
<td>SWIM</td>
<td>System Wide Information Management</td>
</tr>
<tr>
<td>UCD</td>
<td>User-centred design</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
<tr>
<td>USDL</td>
<td>Unified Service Description Language</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
</tbody>
</table>
2 Overview of the Aniketos platform

As expressed in the Aniketos Description of Work, “The main result of the project will be the Aniketos platform shown in Figure 2. The platform complements existing state-of-the-art Service Oriented Architecture (SOA) frameworks by connecting emerging technological solutions with the human practices that are needed to create and maintain secure and trusted composite services. As shown in the figure, the platform provides support in three main areas; design-time support, runtime support and community support, which are explained in the text below.

![Figure 2. Overview of the Aniketos Platform.](image)

**Design-time support:** Consists of methodologies and tools that define and evaluate trustworthiness and risk-based security properties over and between external service components. This allows a service developer to perform service discovery and composition based on security properties and metrics, not just functional descriptors. He/she will also be able to choose service providers and service components by trustworthiness aspects for service composites. Composite services are analysed and prepared through automated on-line mechanisms that gather data concerning both individual components and service compositions as a whole, and the developer is informed about known threats to these through the threat notification from the community. An important part of the work will be not only to ensure that trust and security are maintained, but also that it has the least impact on other functional elements of the system. Mechanisms for performance estimations are used to ensure that the required security properties do not impact on the effectiveness of the system as a whole. The design-time support modules are installed locally into the integrated development environments (IDEs) of the service developers or can be used as stand-alone applications.

**Runtime support:** Any service provider must expect changes to external dependencies or circumstances to occur. The design-time definitions are used to monitor and evaluate the trustworthiness and security violations of service components, also considering contextual information, such as any change in operation conditions and users’ behaviour. The platform allows a proactive increase in trustworthiness by asking for more credentials and tries to control the damages in case of attack by selecting the appropriate security level on which the service can run. A runtime threat alert-and-adapt mechanism is able to receive emerging threat notifications from the community. All
these are possible triggers to dynamic adaptation or recomposition of the service. The runtime support modules are installed locally into the service execution platform of the service providers.

**Community support:** Service developers can find the Aniketos reference architecture and design patterns as part of the community support, which also includes example services, demonstration material, tutorials, development patterns and guidelines. Threat analysis and notification is provided to both service developers and service providers in order to guide design-time composition or trigger runtime adaptation/recomposition based security goals or service components included in the composite service. The service end user will only want to relate to one entity that he/she can place his/her trust to and keep responsible in case something goes wrong, though a composite service has many underlying service providers. The community support enables a certification programme that allows single-point-of-trust, enables responsibility handling and assures the end user in an easy and understandable way how he/she should relate to this service. The Aniketos marketplace offers a way of requesting/offering service components with defined security and trustworthiness properties, facilitating post-project continuation and development through revenue income. Community support is the only centralised part of the Aniketos platform”.

When describing the existing work related to Aniketos work, many times a technique/method or tool could fit into different areas of the Aniketos platform, depending on which of its objectives or functionalities is being analyzed (e.g. a tool can be used in design time for modelling security properties and then in run time for evaluating them). Therefore, in the description of each of the methods or tools we provide a “Support area” field with information on which area(s) of the platform according to Figure 2 the method or tool could be relevant:

- **Design Time Support**
  - DS - Trustworthiness definition and evaluation
  - DS - Security property definition and evaluation
  - DS - Composite service analysis and preparation

- **Runtime Support**
  - RS - Trustworthiness monitoring and evaluation
  - RS - Runtime validation of secure service behaviour
  - RS - Composite service adaptation and recomposition

- **Community Support**
  - CS - Reference Architecture and Patterns
  - CS - End user trust and assurance
  - CS - Threat analysis and notification
  - CS - Aniketos marketplace
3 Classification of works

This section explains the classification method adopted for describing in the following sections the relevant background inside the consortium and the related relevant existing work.

The descriptions of the available techniques and tools have been grouped by Aniketos platform logical modules or components within the System decomposition model of the Component viewpoint of the Aniketos platform architecture (see D1.2 for more details [2]). This System decomposition model describes the platform components and their relationships, as well as the Environment main components, considering Environment as all the systems or modules that interact with the Aniketos platform but are not part of it, and therefore are not the focus of development in the project.

The components used as reference in this document correspond to the current version of D1.2 due by month 12 and still under development. Therefore, this classification might not fit exactly with the final set of components of the platform, due to some refinements that might still be needed on the architecture. Table 1 and Table 2 provide a short description of the components, internal and external to Aniketos platform respectively.

<p>| Table 1 Descriptions of components in the Aniketos platform |
|-------------------------------|--------------------------------------------------|
| <strong>Name</strong>                      | <strong>Description</strong>                                  |
| Socio-technical security      | This is a graphical modelling tool integrated into the Aniketos platform. It supports a socio-technical security modelling language, which allows the analysis and specification of the services composed with Aniketos. |
| modelling tool                |                                                   |
| Model transformation module  | This module transforms the socio-technical security specification models created using the modelling tool into runtime representations, and creates bindings between services and relevant threats. |
| Trustworthiness prediction    | This module predicts the trust level by calculating the trustworthiness of a composite service based on the trust model, especially considering user-centric mechanisms. |
| module                        |                                                   |
| Monitor trustworthiness       | This module enables runtime monitoring of trustworthiness based on the mechanisms and metrics defined according to the trust model. The changes of trust level are notified using the notification module. |
| module                        |                                                   |
| Contract negotiation module   | This module enables the negotiation and creation of contracts for trust relationship and security properties fulfilment based on the security-by-contract paradigm. |
| Security property determination module | This module is used both at design time and at runtime. It allows the determination of the security properties of individual services as well as the overall properties for the federated composed services. |
| Security verification module  | This module verifies the service compliance to security contracts both at design time and at runtime, including monitoring of contract fulfilment. It uses inputs from the trustworthiness prediction module as well as the security property determination module. |
| Security policy monitoring    | This module enables the monitoring of security policies and properties, as well as the monitoring for security contract fulfilment. |
| module                        |                                                   |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat response recommendation module</td>
<td>This module provides recommendations for threat response at runtime, such as a general plan for the intelligence that will compute the appropriate response. The recommendations will be uploaded to the threat repository module to be used by interested services. At design time, services created with Aniketos platform will consult the recommendations to include the threat response capability in order to make services be threat awareness enabled.</td>
</tr>
<tr>
<td>Service threat monitoring module</td>
<td>This module detects and observes changes or threats at runtime and notifies corresponding components when there is a change of threat level. It also updates the threat information in the threat repository at runtime.</td>
</tr>
<tr>
<td>Notification module</td>
<td>This module provides a notification mechanism for services created with the Aniketos platform with respect to changes in the environment and threats. After environment change or threat detection, an adaptation is computed with regard to the agreed trust level and the security properties of a service. Such adaptation is then notified to the responsible entities.</td>
</tr>
<tr>
<td>Community support module</td>
<td>This module gives support to service developers, service composers and end users, including aspects such as: - patterns and guidelines for establishing trust among end users - a set of platform services and the marketplace service.</td>
</tr>
<tr>
<td>Threat repository module</td>
<td>This module is part of the community support and contains a repository of threats, dynamically updated, with information about the threat type and recommended response. At design time, the developer creating the socio-technical security model will lookup the threat repository, to subscribe to relevant threat types and incorporate corresponding threat response capabilities according to the threat response recommendations.</td>
</tr>
<tr>
<td>Marketplace</td>
<td>This module includes a set of services supporting Aniketos marketplace. The marketplace complements existing service registry technology, such as UDDI [3], with specific information on trust and security characteristics. It acts as a service broker for a service consumer giving specific requirements on trustworthiness and security properties. Service providers must be able to upload their offered specifications so that their services are made available for discovery.</td>
</tr>
<tr>
<td>Training material module</td>
<td>This module contains training and individual learning materials that enable the uptake of Aniketos results and the development and delivery of secure and trustworthy services.</td>
</tr>
</tbody>
</table>
### Table 2 Descriptions of environment components

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service composition framework</td>
<td>Framework for enabling service composition, with functionalities like specification, validation, discovery and assembly of services.</td>
</tr>
<tr>
<td>Service specification/planning mechanism</td>
<td>Specification of a service component or a composite service, or the planning or template creation for service composition. The Aniketos socio-technical security models are defined during service specification or planning.</td>
</tr>
<tr>
<td>Service validation mechanism</td>
<td>Validation of a service (both the components in a composition and/or the composition itself)</td>
</tr>
<tr>
<td>Service discovery mechanism</td>
<td>Used for service discovery, both for service components with and without Aniketos support.</td>
</tr>
<tr>
<td>Service registry</td>
<td>A registry for services, for example, UDDI or similar registries. Aniketos complements existing lookup services providing information about trustworthiness and security properties.</td>
</tr>
<tr>
<td>Aniketos compliant service</td>
<td>Services that can be used in a composition and offering their specifications in a way that is compliant and usable by Aniketos.</td>
</tr>
<tr>
<td>Service runtime environment</td>
<td>Runtime environment for the execution of services.</td>
</tr>
<tr>
<td>Service execution mechanism</td>
<td>Mechanism for loading and executing a service (part of the runtime environment)</td>
</tr>
<tr>
<td>Service recomposition mechanism</td>
<td>Mechanism to recompose services at runtime (part of the runtime environment)</td>
</tr>
<tr>
<td>Service adaptation mechanism</td>
<td>Mechanism to adapt a service at runtime (part of the runtime environment), such as changing security properties like the length of an encryption key.</td>
</tr>
<tr>
<td>Service monitoring module</td>
<td>Functionality or service typically found in a runtime environment that can be used to detect attacks, failure or other things that can trigger Aniketos activities (notifications, recompositions, etc)</td>
</tr>
<tr>
<td>Identity management service</td>
<td>A specific type of service to identify entities to which trust is allocated.</td>
</tr>
<tr>
<td>Context sensor</td>
<td>Functionality or service that at runtime can be used to detect context changes that can trigger Aniketos activities; e.g. change of user location.</td>
</tr>
</tbody>
</table>

Figure 3 depicts the current version of the component diagram of the Aniketos platform and its Environment, and shows where each piece of work presented in this document could be allocated.
The description of the existing work (techniques and tools) in the next sections follows the following schema that includes:

- a table summarizing the purpose of the method or tool and other key information
- a short description or abstract
- the first ideas on how the method or tool could be made part of Aniketos platform
- the proposed development roadmap for work in the corresponding work packages of the project.
Note that for some pieces of work they are completely described later in the document than their first instance, where we only provide a reference to a later section. This is because some methods or tools can be allocated within several components, but we have provided the full information in the particular component the method or tool fits best. For example, there is reference in Section 4.5.1 for the MUSIC component to Section 5.3.1.1, which means that MUSIC better fits to Section 5.3.1 than 4.5.

Finally, for each of the components we provide an other related work section in which we summarize other relevant works external to the project consortium. In the case where the identified work mainly consists of a tool, we have described it using the same table as for internal background.

4 Aniketos platform modules

4.1 Socio-technical security modelling tool

4.1.1 SeaMonster – Security Modelling Software (SINTEF)

<table>
<thead>
<tr>
<th>Name</th>
<th>SeaMonster – Security Modelling Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>A security modelling tool for creating and sharing threat models.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>RS - Trustworthiness monitoring and evaluation</td>
</tr>
<tr>
<td></td>
<td>RS - Runtime validation of secure service behaviour</td>
</tr>
<tr>
<td></td>
<td>CS - Threat analysis and notification</td>
</tr>
<tr>
<td>Developed by</td>
<td>SINTEF [Aniketos] and NTNU [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Version 5 is working satisfactory, but there are many possible improvements.</td>
</tr>
<tr>
<td>References</td>
<td>Download site: <a href="http://sourceforge.net/projects/seamonster/">http://sourceforge.net/projects/seamonster/</a></td>
</tr>
<tr>
<td></td>
<td>Wiki: <a href="http://sourceforge.net/apps/mediawiki/seamonster/">http://sourceforge.net/apps/mediawiki/seamonster/</a></td>
</tr>
<tr>
<td></td>
<td>Meland, Spampinato, et al., 2008 [4]</td>
</tr>
<tr>
<td></td>
<td>Ardi, Byers et al., 2007 [5]</td>
</tr>
<tr>
<td></td>
<td>Meland, Ardi et al., 2009 [6]</td>
</tr>
<tr>
<td>System requirements</td>
<td>JRE 1.6 or later. Can be run as Eclipse plugins or standalone distributions for Windows, OSX and Linux.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Open source, released under the LGPLv3 license, <a href="http://www.gnu.org/licenses/lgpl.html">http://www.gnu.org/licenses/lgpl.html</a></td>
</tr>
</tbody>
</table>

1. Short description

SeaMonster is a security modelling tool continuously being developed by an open source community led by SINTEF. The main advantage of SeaMonster is that it supports notations and modelling techniques that security experts and analyzers are already familiar with, namely attack trees (Schneier, 2001 [7]) and misuse cases (Sindre & Opdahl, 2005 [8]) and links common model elements together through a centralized repository (the SHIELDS SVRS, described in Section 4.13.1). In this repository it is possible to upload, download and search for SeaMonster models, thereby supporting collaboration and reuse of threat models between projects, people and organizations.

SeaMonster is based on Eclipse, which is basically an application platform where a very large set of plugins can be added to suit the needs of the users. The three main Eclipse plugin frameworks
SeaMonster benefits from are the Graphical Modeling Framework (GMF), the Eclipse Modeling Framework (EMF) and the Graphical Editing Framework (GEF). GMF is a framework for developing Eclipse graphical editors, and functions as a bridge between EMF and GEF.

2. Making SeaMonster part of the Aniketos platform

To be concise, Aniketos needs a tool to define security critical properties for a given composite service, so that the Threat analysis and notification module can alert this service if e.g. a sub-service is compromised by an ongoing attack or newly discovered vulnerability. SeaMonster can act as a foundation or input to such a tool.

For composite services, threat modelling is needed to define “triggers” for recomposition and adaptation. E.g. if a sub-service is compromised in a certain way, the composite service must act in order to maintain its trust and security level. This is illustrated in Figure 1 in Section 1.1., where relevant threat awareness is an activity performed during design-time composition, and a threat notification is issued to the service when there is a change in the threat picture, which again triggers a runtime adaptation/recomposition.

Defining critical threats and vulnerability classes that would degrade a service/service component is a necessary task in the Security property definition and evaluation module, which again would trigger actions from the Threat analysis and notification module if concrete threats or vulnerability instances are discovered for e.g. a sub-service of a composite service. The Composite service adaptation and recomposition module will then react in order to sustain the security and trust level of the composite service. A threat model created with SeaMonster can be used in the definition phase at design-time and for threat analysis at runtime in the Community support section of Figure 2 in Section 2.

3. Development roadmap

Creating a socio-technical security modelling language and tool is part of T1.2 (D1.3 and D1.4), where SINTEF will participate. Whether we decide to create something new or extend existing work will influence the time schedule. A tool such as SeaMonster is developed through model driven development, and almost all the code is automatically generated. This requires a good definition of the language to begin with, precise modelling, and less time to actual implementation. MDD and code generation enables fast prototyping, with deployment to many platforms simultaneously. Work in WP’s 2-3-4 depends on the work in WP1, and integration in WP5 is necessary, both for the tool itself, and the mechanisms that handle the community functionality (threat notification).
4.1.2 TECNALIA Security Framework (TECNALIA)

<table>
<thead>
<tr>
<th>Name</th>
<th>TECNALIA Security Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Create and deploy security policies based on WS-SecurityPolicy standard.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>RS - Runtime validation of secure service behaviour</td>
</tr>
<tr>
<td>Developed by</td>
<td>TECNALIA [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype, Work in progress</td>
</tr>
<tr>
<td>References</td>
<td>None</td>
</tr>
<tr>
<td>System requirements</td>
<td>JRE 1.5, Apache Tomcat 6.0.x, Apache Axis2 1.5.1</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Proprietary</td>
</tr>
</tbody>
</table>

1. Short description

TECNALIA’s Security Framework is an Eclipse [9] based tool for the creation and automatic deployment of Security Policies. In fact, the tool is an Eclipse Rich Platform (RCP) application, built upon the RCP framework of Eclipse. The objective of this tool is two folded: (i) to provide a graphical environment for the modelling of WS-SecurityPolicy compliant policies and (ii) to provide a runtime environment to test the designed policies.

This way, after the policy has been modelled through the editor, it can be easily attached to a Web Service, and deployed into an Apache Axis2 application server.

2. Making Tecnalia’s Security Framework part of the Aniketos platform

TECNALIA’s Security Framework is a prototype that fits within the Aniketos design time support techniques and tools, as it targets the modelling stage of security non functional properties, in form of security policies. Although the tool enables the deployment of secured services, it is thought to be used only for testing the security functionality.

TECNALIA’s Security Framework can be improved with Aniketos’ socio technical security modelling language, extending its functionality beyond the compatibility with the WS-SecurityPolicy Standard (its main target right now) and making it able to be integrated into other design tools.

3. Development roadmap

The development roadmap for this tool would be focused into two main goals: (i) improvements with the feedback from the rest of partners and (ii) development of new features with the socio technological language and tool. This work should be done at:

T1.3: Identification of the requirements needed to integrate the socio-technical language. The actual security framework is very technical and work should be done allowing the usage by non-technicians.

T5.1: This task will elaborate the design that will guide the implementation phase. The design decisions taken in this task will serve as baseline for the security framework. This way, the framework will be adapted to interact with Aniketos platform.
4.1.3 Modelling of Evolutionary Requirements (UNITN)

Name: Modelling of Evolutionary Requirements  
Type: Technique  
Purpose: Modelling of security requirements and their lifelong changes.  
Support area: DS - Security property definition and evaluation  
Developed by: DISI – University of Trento (UNITN) [Aniketos]  
Maturity: Final Deliverable (16 months work) of the SecureChange project.  
References: Deliverable D3.2 SecureChange Project in http://securechange.eu/content/deliverables  
System requirements: N/A  
Miscellaneous: N/A

1. Short description
A methodology for evolutionary requirements aims to provide modelling and reasoning mechanisms to cope with eventual changes in security requirements. Changes are inevitable and ensuring a secure change is the main addressed research problem. The methodology allows for accommodating changes in security requirements and guaranteeing the correctness and the fitness to purpose of such changes. To this end, the methodology proposes a model-driven approach to represent security requirements and eventual changes on them. It also provides an automated reasoning to manage eventual changes and the argumentation that may arise amongst different stakeholders/analysts about them. The methodology consists of: (i) A metamodel that incorporates requirements, security requirements, and evolution of security requirements, (ii) Argumentation model and automated support to systematically and iteratively manage argumentation between different stakeholders, (iii) Specifying rules (patterns) for evolutionary changes based on model transformation.

2. Making “Modelling of Evolutionary Requirements” part of the Aniketos platform
Modeling of Evolutionary security requirements framework is the integration into a unified model of two well known mainstream approaches to requirements and security requirements modelling techniques: Tropos Goal Modeling and Problem Frames, besides their security specialized versions: Secure Tropos and Abuse Frames. The integration is not a simple merging; rather it elicits carefully core concepts, unifies the similar ones, and rules out redundant and irrelevant ones. The Evolutionary Requirements metamodel supports the definition of the Socio Technical Security Modeling Tool as it focuses on the analysis of requirements at the problem domain which fits to modelling the socio-technical system, which is the goal of WP1 (Task1.2). The building blocks of this model are actors, goals, roles, context, situation, resources, assets, activity action, and so on. That is to say, the model captures the problem domain of a system and positions the software requirements within a socio-technical system. Aniketos emphasizes the socio-technical system in which services exist and interact with.

Moreover, the model provides constructs to model the evolution of security requirements in terms of changes in situation. This is also essential for Aniketos as the change starts with the socio-technical system and leads the service composition and adaptation.

3. Development roadmap
This work is directly relevant to WP1, namely T1.2 that is to develop a novel requirements engineering language that incorporates organizational and social notions. In fact, Modelling Evolutionary Requirements work also emphasizes the social and organizational dimension in
modelling requirements and adopts designated notions such as role, goal, task, trust, and situation changes, and therefore could be a rich input to exploit while accomplishing T1.2.

Modelling of situation changes and security response, which is developed in Modelling Evolutionary Requirements, can be a conceptual input for WP4 that is targeted to handle responses to changes and threats in service environment.

4.1.4 MASTER Design Toolkit (UNITN)

Master Design Toolkit (see section 4.10.2) allows system analysts modelling and analyzing business processes to identify requirements, which are further refined to obtain a design model. The Design Model describes the interactions between these processes. Especially, the (graphical) workbench (part of the toolkit) facilitates the analysis and modelling process. It provides a design environment that allows users to build a design the model for the business process that is being analyzed. Therefore, MASTER Design Toolkit could be used as a baseline for WP1, Task 1.2. The idea behind this tool could be adapted to satisfy the need of T1.2 for expressing and analysing security requirements for the services.
4.1.5 Goal Risk Model (UNITN)

Name: Goal Risk Model
Type: Technique
Purpose: Perform risk analysis
Support area: DS - Trustworthiness definition and evaluation
DS - Security property definition and evaluation
Developed by: DISI – University of Trento (UNITN) [Aniketos]
Maturity: Applied to EU projects, disseminated through publications
References: MASTER project [10]
Asnar et al., 2007 [13]
Asnar and Giorgini, 2006 [14]
Yu, 1996 [15]
Zannone, 2007 [16]

1. Short description
Goal Risk Model (GRM) is a modelling framework which adapts some basic concepts from i* [15] and SI* [16], extending them with risk related concepts. The framework is intended to capture requirements of a socio-technical system, relevant risks, and necessary treatments to mitigate them. Essentially, it extends the Tropos methodology with three conceptual layers of analysis representing values, events, and treatments. The value layer captures the concepts related to the actors’ needs and the means to achieve them. We assume the system generates the value when it satisfies the stakeholders’ needs. Uncertainties are modelled in the event layer. To reduce the risk of uncertainties, countermeasures are introduced into the organization and modelled in the treatment layer.

2. Making “Goal Risk model” part of the Aniketos platform
Goal-risk modelling framework (GRM) enriches Tropos goal model with risk related constructs. The goal model and the added constructs are meant to represent the requirements and the risks in achieving them when certain events happen and the treatment the system can adopt to mitigate/handle risks. All of these constructs are in line with Aniketos Socio Technical Security Modeling Tool as they are devised for an early stage of modelling which is the socio-technical system modelling that the development process should start with. Modeling risks in achieving users’ requirements is essential for Aniketos socio-technical security requirements.

3. Development roadmap
The Goal Risk model can serve as a baseline for WP1, Task 1.2. Since the framework is intended to capture requirements of a socio-technical system, it is appropriate to support the analysis and the identification of socio-technical security requirements of the services created in the Aniketos platform.
4.1.6 Secure Tropos (UNITN)

<table>
<thead>
<tr>
<th>Name</th>
<th>Secure Tropos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Technique (Methodology)</td>
</tr>
<tr>
<td>Purpose</td>
<td>A Requirements Engineering methodology founded on business and security concepts such as actors, goals, tasks, resources, delegation, and trust.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td>Developed by</td>
<td>DISI – University of Trento (UNITN) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Well known and referenced in the scientific community. Applied to case studies in EU projects (e.g. Serenity)</td>
</tr>
<tr>
<td>References</td>
<td>Zannone, 2008 [17]</td>
</tr>
<tr>
<td></td>
<td>Giorgini, Massacci and Mylopoulos, 2003 [18]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1. Short description

The Secure Tropos methodology, based on the SI* (secure i*, see [15]) modelling language [16], is an agent-oriented security requirements engineering methodology. Secure Tropos allows for the design of secure information systems starting since the initial development stages (early requirements). It has been conceived from the observation that a system cannot be secure if security is not taken into account since the beginning of the development process. The language it is based upon, SI*, takes a subset of the concepts of i* (actor, role, agent, goal, resource, task, softgoal) and complements it with a number of constructs to deal with security issues (objectives, entailments, capabilities, trust of execution and of permission, delegation of execution and of permission). SI* is formally represented and Secure Tropos comes with different automated reasoning techniques. The Secure Tropos methodology consists of two main phases: early requirements (before the system-to-be is introduced) and late requirements (after introduction).

2. Making Secure Tropos part of the Aniketos platform

Aniketos deals with socio-technical systems, which are an interplay of organizations, humans, and technical systems. Secure Tropos allows for the modelling of socio-technical systems, through the notion of actor. The notion of delegation is tightly related to that of service which is at the basis of Aniketos: whenever a service provider commits to provide a service, there is a delegation from the service requester to the provider for that service. Secure Tropos is therefore related to Aniketos for what concerns the modelling of service-oriented settings. The capability to reason about sophisticated security properties (including transitive ones) makes Secure Tropos a good candidate to reason about composed services.

The Secure Tropos methodology provides a framework for understanding security (in particular trust) issues in organizations. Secure Tropos enriches Tropos goal oriented requirements engineering methodology with a specialized construct to model security requirements (trust, ownership, permission delegation, etc). Secure Tropos supports the Aniketos Socio Technical Security Modeling Tool by capturing the security related properties at the socio-technical level that leads to identify what mechanism has to be supported in the implemented system in order to keep the system secure. However, several enhancements can be made to fit it better:
• Consider the socio-technical dimension of the settings Aniketos tackles. This means that different types of actors should be considered (humans, organizations, systems) and such notion might be specialized.

• Take into account settings where there is limited knowledge about participants. Secure Tropos presumes that the designer knows the goals and tasks of participants. The more general case is that such assumption cannot be made. The notion of social commitment is a possible way to better represent such situation.

• Create a more refined version of the framework that can be used for the development of real service-based applications. Many of the concepts of the SI* language can be refined/specialized (e.g. goal, task, resource).

These enhancements mainly relate to WP1, specifically to the definition of a modelling language to express and analyse security and trust requirements.

3. Development roadmap

Secure Tropos can be used as baseline for WP1, specifically for Task 1.2 that deals with the definition of the initial version of the socio-technical security modelling language. In particular, we envisage that the basic concepts and reasoning mechanisms can be exploited and refined.

4.1.7 Si* tool (UNITN)

<table>
<thead>
<tr>
<th>Name</th>
<th>Si* tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Design Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>A computer-aided design tool that allows for modelling organizational settings using the Si* modelling language, the Goal-Risk framework, organizational planning settings, and security patterns; it allows also for automated reasoning on these models in order to identify problems.</td>
</tr>
</tbody>
</table>
| Support area | DS - Trustworthiness definition and evaluation  
DS - Security property definition and evaluation |
| Developed by | DISI – University of Trento (UNITN) [Aniketos] |
| Maturity     | The tool is a prototype but has been used to model several case studies. It is publicly available online. |
| References   | Serenity project [19]  
Website: [http://sesa.dit.unitn.it/sistar_tool/home.php](http://sesa.dit.unitn.it/sistar_tool/home.php)  
The website allows to download the Si* tool. Moreover, it contains detailed information about the tool rationale, the underlying modelling languages, an installation guide, and a support forum.  
The main scientific references are the following:  
Massacci et al., 2007, 14(3) [20]  
Massacci et al., 2007, 17(2) [21] |
| System requirements | Java Runtime Environment. The Si* tool can be obtained as a plugin (platform independent) or as a rich-client platform (Windows only).  
If the tool is downloaded as a plugin, requirements are: Eclipse SDK 3.2.2, GEF-SDK 3.2.2, EMF-SDK 2.2.2, Transaction-SDK 1.0.3, OCL-SKD 1.0.2, Query-SDK 1.0.2, Validation-SDK 1.0.2, GMF-SDK 1.0.3, WTP 1.5.3, JEM-SDK 1.2.3 |
| Miscellaneous | N/A |
1. Short description

Si* Tool is a graphical tool where it is possible to draw Secure Tropos models and to perform the effective formal analysis of Secure Tropos specifications. The tool provides as an Eclipse's plugin and use XML as its document format. Formal analysis is based on logic programming. Si* Tool allows to different systems based on Datalog to analyse Secure Tropos specification.

2. Making Si* tool part of the Aniketos platform

The Si* tool is a powerful tool to model security requirements and reason about them at design-time. Si* tool allows for representing security requirements (graphically and formally) and analysing the model for detecting several properties such as the compliance between functional requirements and security requirements.

The Si* tool is a candidate become part of the Aniketos design-time platform, while its components can be used also for the runtime framework (e.g. the ASP analysis). Si* is based on modelling and reasoning techniques that are compatible with the vision of Aniketos, for they are suitable for socio-technical systems and express trust-related properties. In particular, we envisage the following usages of Si* tool:

- The Si* tool can be extended to support the requirements modelling language that will be developed for Aniketos. On the basis of the new language, new reasoning facilities can be integrated to automatically check the models.
- The existing reasoning techniques (such as ASP analysis or the Goal Reasoning about risk) can be reused for runtime reasoning in the Aniketos platform. These enhancements mainly relate to WP1, but also to WP2, specifically for modelling and reasoning about trust.

3. Development roadmap

The Si* will be part of the baseline for WP1, Task 1.2. The modelling tool is mature enough to be extended, since it has been used in several projects. The UNITN team has developed it, and has the expertise to use it as a baseline for the Aniketos requirements modelling language. Indeed, the current tool does not support all the features needed for the scenarios Aniketos tackles, such as service (re)composition and trustworthiness.

4.1.8 Organizational S&D Patterns (UNITN)

<table>
<thead>
<tr>
<th>Name</th>
<th>Organizational security and dependability (S&amp;D) Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Technique (Modelling)</td>
</tr>
<tr>
<td>Purpose</td>
<td>A modelling technique that allows expressing security and dependability patterns representing recurring S&amp;D issues. Patterns can be applied in a certain context and consist of a solution that tackles the issues.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td>Developed by</td>
<td>DISI – University of Trento (UNITN) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>The technique is based on the well-known Si* language. It is publicly available through the Si* tool and published in scientific venues.</td>
</tr>
<tr>
<td>References</td>
<td>Serenity project [19]</td>
</tr>
<tr>
<td></td>
<td>Compagna et al., 2009 [22]</td>
</tr>
<tr>
<td></td>
<td>Krausová et al., 2009 [23]</td>
</tr>
<tr>
<td>System requirements</td>
<td>Tool support is provided by the Si* tool</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>N/A</td>
</tr>
</tbody>
</table>
1. Short description
Organizational Security and Dependability (S&D) patterns are a specialization of the concept of S&D pattern introduced by the Serenity project. A pattern is a solution to a common problem; design patterns are probably the most popular example of pattern. S&D patterns are a kind of patterns that deals with security and dependability. Organizational S&D patterns talk of S&D at the organizational level, and describe how an organization should change in order to cope with a specific S&D issue. A pattern is composed of a context where the pattern applies, a requirement that describes the provided security property, a solution that describes the set of changes the pattern applies, and a set of consequences.

2. Making Organizational S&D patterns part of the Aniketos platform
Organizational S&D patterns identify well-practiced solutions for a common security and dependability problems. The work supports the Aniketos Socio Technical Security Modeling Tool by identifying patterns in organizational model, represented via Secure Tropos modelling language, where security and dependability are weak and giving a solution that incorporate changes in the socio-technical system.

Organizational S&D patterns can express, among others, trust properties. The trustworthy service composition proposed by Aniketos is a setting where these patterns perfectly fit, for they can serve as a basis to drive re-composition: if the pattern context holds, a different service should be chosen.

Organizational S&D patterns can be used with a twofold purpose:

- At design-time, they might be used to check whether an organization suffers of S&D issues and, if so, the pattern can be applied to effect changes to the model. Such changes will result in a set of actions to effect such changes to the real organization.

- At runtime, they can be used to provide adaptive behaviour, provided that traceability links exist between the S&D patterns and the real organization. A good starting point is the Serenity Runtime Framework, which already encompasses the notion of S&D pattern, though it doesn't provide links to the organization. However, an initial attempt to map this link was done for the final demonstration by UNITN.

3. Development roadmap
Organizational S&D patterns can be the baseline for WP1, Task 1.2. The patterns can be used at requirements time to detect S&D issues. The pattern language should be extended so that the specific features of Aniketos are considered, such as service composition and trustworthiness.

The revised version of the patterns can become an input for WP3, Task T3.3 (Service composition adaptation and reconfiguration techniques for security maintenance). For such purpose, it is fundamental to make sure that all fields in an S&D pattern are formalized in a way the Aniketos middleware can understand at runtime.
4.1.9 Contextual Requirements Modelling and Analysis (UNITN)

<table>
<thead>
<tr>
<th>Name</th>
<th>Contextual Requirements Modelling and Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>Weaving together variability of requirements and context at an early analysis stage, the goal analysis, as a basis for system operating in varying context. Reasoning about contextual requirements for discovering modelling errors, reducing costs, providing runtime adaptation rationale, and tailoring the model to specific deployment environment.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Composite service analysis and preparation</td>
</tr>
<tr>
<td>Developed by</td>
<td>University of Trento (UNITN) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature work: published in international conferences and journals.</td>
</tr>
<tr>
<td>References</td>
<td>Ali et al., 2010 [24]</td>
</tr>
<tr>
<td></td>
<td>Ali et al., 2009 [25]</td>
</tr>
<tr>
<td></td>
<td>Ali et al., 2008 [26]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1. Short description
Most of requirements engineering (RE) research ignores, or presumes a uniform nature of, the context in which the system operates. This assumption is no longer valid in emerging computing paradigms, such as Ambient, Pervasive and Ubiquitous Computing, where it is essential to monitor and adapt to an inherently varying context. There is a strong relationship between requirements and context. Context might be considered to determine the set of requirements relevant to a system, to derive the alternatives the system can adopt to reach these requirements, and to assess the quality of each alternative. Our work presents a RE framework that explicitly captures and analyzes the relationship between requirements and context. We develop a conceptual modelling language, the contextual goal model, which captures the relationship between context and requirements at the goal level. We provide constructs to analyze context and define ways through which an actor can judge its validity.

2. Making “Contextual Requirements Modelling and Analysis” part of the Aniketos platform
Adaptability to context is a core feature for modern socio-technical systems that weave computing with human’s living environment so the interaction between users and computers is minimized. Aniketos targets socio-technical systems engineering and thus the Contextual Requirements Modelling can directly contribute to the modelling language Aniketos will develop.
Contextual Requirements Modelling allows modelling the variability of requirements according to the variability of context and analyzes the resulted contextual requirements models. This framework support Aniketos Socio Technical Security Modeling Tool by studying the changes in the requirements at the organizational level, represented via Tropos goal model, with respect to change of contexts that is represented via constructs devised expressively to allow a systematic identification of relevant contextual information.
Adaptability to context is a main driver to derive system functionality to execute. This can be seen in terms of services as a driver for identifying relevant services to compose. For example, if a person is in a city where he does not know the language and he receives some promotion via his mobile phone in that language then invoking the service of translator could be activated.
3. Development roadmap
Contextual adaptability of requirements is an essential source of service composition adaptability. At the first place, users’ requirements change according to context changes. The changes of user requirements lead to change in the services to invoke and execute. Thus we presume that Contextual Requirements Modelling technique makes core part of the modelling language proposed in WP1 (T1.2).

4.1.10 Modelling Interaction between Agents via Commitments (UNITN)

<table>
<thead>
<tr>
<th>Name</th>
<th>Modelling Interaction between Agents via Commitments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>Modelling of social relationships between different agents.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS - Composite service analysis and preparation</td>
</tr>
<tr>
<td></td>
<td>RS - Trustworthiness monitoring and evaluation</td>
</tr>
<tr>
<td></td>
<td>RS - Composite service adaptation and recomposition</td>
</tr>
<tr>
<td>Developed by</td>
<td>DISI – University of Trento (UNITN) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>This work has been published in top international conferences and journals. Prototypes exist to demonstrate various features.</td>
</tr>
<tr>
<td>References</td>
<td>Desai et al., 2009 [27]</td>
</tr>
<tr>
<td></td>
<td>Chopra et al., 2010 [28]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1. Short description
Social commitments are a conceptual abstraction that represents social relationships between different agents. Commitments are made in a socio-legal context, which defines the rules of engagement for agents in this context, such as penalties and compensations. A commitment is a promise (a contract stipulated in the socio-legal context) between a debtor and a creditor to bring about a certain state of the world. A conditional commitment is a conditional promise: the debtor will do something if some state of the world is brought about before (typically by the creditor). Commitments are an adequate abstraction to represent service-oriented settings, for they represent—at the business level—the interface that service providers (debtors) offer to service consumers (creditors).

2. Making “Modelling Interaction between Agents via Commitments” part of the Aniketos platform
Socio-technical systems are usually designed as a set of interacting agents. Modeling Interaction between Agents via Commitments provides Aniketos Socio-technical security modelling technique with a basic and essential form of interaction that is based on commitments. This allows the modelling framework to support the open settings the socio-technical systems are characterized with. This is a core characteristic for modern systems such as service based systems.

Commitments are a candidate technique to become part of the Aniketos platform both ad design-time and at run-time;

At design-time, commitments can be used to define a service-oriented application in terms of roles and the social relations (at the business level) that can be offered by agents playing those roles. Also, commitments can be part of the design of a single service provider to model and verify whether its goals are achievable in a certain service-oriented application. In order to exploit commitments, the underlying conceptual model should be augmented to deal with (a) the quality of service that
characterizes most services; (b) compensation actions to define robust contracts between agents; (c) security-related properties, such as the possibility of delegating or assigning the entire commitment or parts of it

At run-time, commitments can become the abstraction through which an agent reasons to choose and recompose composite services. The notion of composition is naturally supported by commitments. Indeed, composition means to use together several commitments (sometimes via chaining) to assemble a service that cannot be provided by a single provider. However, this requires us to extend commitments with (a) primitives to treat different commitments as a composite service that is perceived as a unique entity by users; (b) trust and reputation mechanisms that allow for choosing the most reliable commitment.

3. Development roadmap

Commitments can be used as a modelling abstraction for the socio-technical language (D1.3) in T1.2. They capture interaction between agents at a high level of abstraction.

Recent work on adaptation via commitments can be extended to provide threat detection and response. This could be part of the algorithms in D4.1 and D4.3.

4.1.11 Thales Risk Assessment Domain Specific Modelling Language (THALES)

<table>
<thead>
<tr>
<th>Name</th>
<th>Thales Risk Assessment Domain Specific Modelling Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>The Thales RA DSML is a system security engineering tool that enables to perform a security risk analysis when designing a system</td>
</tr>
<tr>
<td>Support area</td>
<td>CS - Threat analysis and notification, DS - Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td>Developed by</td>
<td>Thales [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype V0, Work in progress</td>
</tr>
<tr>
<td>References</td>
<td>Normand and Félix, 2009 [29]</td>
</tr>
<tr>
<td>System requirements</td>
<td>Eclipse</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Thales proprietary framework</td>
</tr>
</tbody>
</table>

1. Short description

System security engineering classically involves: 1) the analysis and assessment of security risks encountered by the system, 2) the specification of requirements for security measures to address those risks, and 3) the design, development, integration and validation of the security architecture, functions and mechanisms that address those requirements.

Our present work is focusing on security engineering activities 1 and 2 above. Our objective is to provide adequate and efficient tooling to security engineers for an effective integration of security engineering in the process of critical system design; this will enable a better targeting of security specifications.

2. Making Thales Risk Assessment DSML part of the Aniketos platform

The Risk analysis concepts and modelling language of the Thales Risk Assessment DSML could be adapted and re-used in Aniketos socio-technical DSML. But this tool has not been identified as relevant for its socio-technical modelling capabilities, but as reference to the next modelling step in the system security requirement elicitation process. We must both avoid redundancies and ensure the transformation of the early security requirement elicitation process with the next phases, in particular risk assessment and risk mitigation. The risk meta-model could be re-used in Aniketos threat analysis and notification module.
The tool enables security engineering at design time and could provide a risk analysis model for design time.

3. Development roadmap

The current version of the Thales Risk Assessment Domain Specific Modelling Language prototype is a standalone tool with an interface to DOORS T-REK. Aniketos being service-oriented, the Thales Risk Assessment Domain Specific Modelling Language will be first ported in a service-oriented framework called Service Modelling Suite (SMS). Adaptations will be performed where required to cope with the service orientation.

In a second step, the tool will be used to perform the risk assessment of the SWIM use-case (WP6b). If and when required by the use-case, the method, language and/or tool will be evolved so as to offer the required functionality.

Finally, the method, language and tool’s performances will be compared to other risk assessment methods, languages and tools available in Aniketos, e.g. the UNITN’s Goal Risk, SINTEF’s SeaMonster Model. Pros and cons will be discussed.

4.1.12 Thales SOA Modelling Suite (THALES)

<table>
<thead>
<tr>
<th>Name</th>
<th>Thales SOA Modelling Suite (SMS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Service modelling</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td>Developed by</td>
<td>Thales [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>In-house, Prototype, Work in progress</td>
</tr>
<tr>
<td>References</td>
<td>None</td>
</tr>
<tr>
<td>System requirements</td>
<td>Eclipse</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Requires Obeo Designer licence.</td>
</tr>
</tbody>
</table>

1. Short description

The purpose of the Thales SOA Modelling Suite is to capture the different concerns related to SOA architectures specifications and implementations. Ultimately, it shall provide its users with domain specific languages (DSLs), as well as their corresponding graphical representations, that allow specifying efficiently SOA concerns. Such DSLs are designed by capturing the concepts associated with SOA standards, technologies and Thales engineers’ specific requirements.

2. Making Thales SMS part of the Aniketos platform

The Thales SOA Modelling Suite already comprises some code-level WS-security policy generation capabilities. The Thales Risk Assessment Domain Specific Modelling Language (see Section 4.1.11) should shortly be integrated within SMS. The integrated result should be used to model the ATM/SWIM use-case in WP6.

By analysing the capabilities to integrate a socio-technical approach within SMS, we are trying to assess the interest of a comprehensive, end-to-end tool for security engineering, with a scope extending from business process modelling to code generation. Some of the modelling concepts proposed within Aniketos for socio-technical modelling have a strong overlap with actually used industrial system modelling languages. We need to avoid redundancies as much as possible, and when we cannot, we need to ensure the consistency of the concepts (static & behavioural), in particular to allow for parallel modelling using both languages by system engineers and security engineers. Consistency of the concrete syntax is secondary, but may also be envisaged.
3. Development roadmap

SMS is an in-house prototype on which work is still on-going. Internal delivery dates should be consistent with WP6 requirements related to the ATM/SWIM use-case.

SMS is a stand-alone design tool; therefore, global integration with other tools in WP5 is not being considered.

4.1.13 Property and Policy Tools (LJMU)

<table>
<thead>
<tr>
<th>Name</th>
<th>Property and Policy Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tools</td>
</tr>
<tr>
<td>Purpose</td>
<td>Creating property and policy definition files</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td>Developed by</td>
<td>Liverpool John Moores University (LJMU) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>In-house prototype</td>
</tr>
<tr>
<td>References</td>
<td>Zhou et al., 2010 [30]</td>
</tr>
<tr>
<td>System requirements</td>
<td>.Net runtime</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Proprietary, currently no licence defined</td>
</tr>
</tbody>
</table>

1. Short description

In order to understand and analyse the nature of composed services, properties of the individual services that make up the composition must be either determined or predefined. In addition, a method is also needed for the definition of policies that are to be applied, either to individual services or to the composition as a whole.

The tools described here allow such descriptions to be constructed using a collection of straightforward user-interfaces. The resulting properties are output in a simple XML file format that can be read in by other tools.

Two tools relate specifically to property definition. These are the PropertyInterface and the CompositionClient property selection interface. In future development we also expect a further tool, the PolicyInterface, to complement these. They each serve a different purpose as follows.

The PropertyInterface allows property sets to be defined. These specify a range of possible properties that a service or device might support. The tool outputs Property Set files.

The CompositionClient property selection interface accepts Property Set files and allows specific properties to be selected from those that are available. It outputs Property files.

The PolicyInterface tool will allow the creation of policies that amount to logical and arithmetic expressions with variables and constants taken from the properties available. The tool will also allow policies to be resolved against specific sets of properties to determine policy satisfaction. The tool will accept Property Set files (for policy creation) and will generate Policy files as output. It is expected to accept Policy files and Property files for policy testing.

2. Making the Property tools part of the Aniketos platform

In the first instance the tools could be used as an interim means of generating property information for services, while other methods are developed to generate this information automatically.

We are also in the process of developing a PolicyInterface for the creation of simple policy rules, as described below in the development roadmap. We expect this could be used as a constituent part of the overall policy definition process, for example allowing the generation of logical expressions for input into other tools.
3. Development roadmap

We anticipate that these tools might be usefully applied or developed as part of WP 1 (task 2.3), WP 3 (tasks 3.1, 3.2, 3.4) and WP 4 (task 4.5).

We expect to add functionality to these tools in a number of areas as follows:

Policy Interface

An additional element of the toolset that we are developing as part of the project is the Policy Interface. This will allow arithmetical and logical expressions to be defined over the set of properties that services can take. Expression of this form could be used to represent a policy statement (or entire policy) for individual services to be tested against.

For example, a policy might specify that a certain property must hold for a given service, or that a risk level must be lower than a certain level. The tool will allow such constructs to be defined and entered easily using a simple user interface.

One of the most important design goals of the interface will be to allow multiple nested expressions to be built up easily. Hence it is intended that there will be no limit to the complexity of the logical expressions that can be created using the tool.

Figure 5 provides an example of an output policy file that might be generated by such a tool. Such files could be tested against Property files to determine whether a particular service satisfies the requirements specified by the policy.

Note that these policies are intentionally rather simplistic in nature and are only intended to apply to individual services. For interactions between services, more complex policy definition such as that provided by MATTS (described in Section 4.7.2) is required. The logical expressions described here can be usefully applied as elements within more complex compositional policies.

![PolicyExpression]

<PolicyExpression>

((Volume > 1000) OR ((Depth > 10) AND (Area > 100)) OR ((Height > 10) AND (Width > 10) AND (Depth > 10)))

</PolicyExpression>

Figure 4. Example Policy file.

4.1.14 Other related work to Socio-technical security modelling tool

There are several approaches related to socio-technical modelling at the level of requirements. The MAP approach [59] describes organizations in terms of intentions and strategies: a map is a directed graph where nodes are intentions and directed arrows represent strategies. A strategy explains how to achieve one intention starting from another intention. Maps have been recently used to define the concept of Intentional Services Oriented Architecture (ISOA) [60], where the authors conceive services in terms of intentional abstractions such as goals. The i* framework [61] starts from the identification of the stakeholders in the analyzed organizational setting and model these stakeholders—actors—in terms of their own goals and the dependencies between them. i* has been recently used to describe services [62] this approach violates agents heterogeneity by making assumptions about other participants’ internals. KAOS [63] exploits a system-oriented perspective to specify requirements. Stakeholders are essential to gather system goals, but they are not explicitly represented in KAOS models. Leaf level goals are assigned to agents on the basis of a responsibility principle.

Some approaches focus on the specification of security properties. Liu et al. [64] uses i* for dealing with security and privacy requirements by introducing soft-goals, as “Security” or “Privacy”, to model these notions, and use dependencies analysis to check if the system is secure. This approach also offers facilities for the analysis threats, vulnerabilities, and countermeasures. Another early RE example is
McDermott and Fox [66] have adapted a UML [67] modelling technique, use cases, to capture and analyze security requirements, and they call these abuse cases. An abuse case is a specification of an interaction between the system-to-be and one or more actors, where the result of such an interaction is harmful to the system or to one of the actors in the system. Sindre and Opdahl [8] propose to capture security threats to the system by defining misuse cases, the converse of UML use cases, which describe uses that the system should not allow. This new construct makes it possible to represent actions that the system should prevent together with those actions which it should support. Together with misuse cases, the authors introduced the notion of misactor to represent attackers external to the system.

Jürjens [68] proposes to integrate security requirements analysis with a standard system development process, and modelling of security related features such as confidentiality, access control, etc. through UMLsec. UMLsec is an extension of UML that includes mechanisms for specifying security requirements and visualizing them on UML diagrams. Basin et al. [69] present a UML-based modelling language, called SecureUML. Their approach is focused on modelling access control policies and integrating them into a model-driven software development process. SecureUML is a modelling language designed to integrate information relevant to access control into application models defined with UML.

An extension of the KAOS methodology is presented in [70] where the notion of obstacle is introduced. KAOS uses the notion of goal as a set of desired behaviours. Likewise, an obstacle defines a set of undesirable behaviours. Therefore, the negation of such obstacles is used to determine preconditions for the goal to be achieved. Obstacles analysis consists in taking a pessimistic view of goals. It aims at identifying as many ways of breaking goals as possible in order to resolve each such situation.

Lin et al. [71] propose to incorporate anti-requirements into problem frames and call this extension Abuse Frames. The purpose of abuse frames is to represent security threats and to facilitate the analysis of the conditions in the system in which a security violation occurs.

Other modelling approaches consider the design of high-variability systems, i.e. systems that can adapt at runtime by switching from one variant to another. Customizing requirements to fit to user skills and preferences was studied in [72] and [73] where goal model was the investigated requirements model. The selection between goal model variants is based on one dimension of context, i.e. user skills, related to the atomic goals (executable tasks) of the goal hierarchy, and on user preferences expressed over soft-goals.

Lapouchnian et al. [74] propose techniques to design autonomic software based on an extended goal modelling framework, but the relation with context is not focused on. Salifu et al. [75] apply Problem Frames approach to analyze different specifications that can satisfy the core requirements, under different contexts. The relationship between contexts, requirements, and the specification (machine) are represented by a problem description. Alternative problem descriptions corresponding to different contexts are elicited to identify variant problems. Variant problems are variations of the original problem adapted for a particular context. Hartmann et al. [76] suggest studying the relation between context and features to support the engineering of software supply chains. Their approach allows for more systematic derivation of a product that fits to the environment in which it operates.

Lund, Solhaug and Stølen [77] propose an approach, named CORAS, for conducting security risk analysis. CORAS is a model-driven method for defensive risk analysis featuring a tool-supported modelling language specially designed to model risks and threats. The purpose is to identify the assets, what are the threats on them and how the assets can be protected.
4.2 Model transformation module

4.2.1 Related work to Model transformation module

There exist various types of technology for doing model transformation, typically related to Model-Driven Development (MDD), where they play a fundamental role. As the model transformations are directly related to the techniques and tools used for developing the socio-technical security modelling module, the analysis of the candidate transformation technologies is scheduled for later stages of the project, once the requirements to the necessary security models (transformation input) and security contracts (transformation output) are defined.

Despite the large amount of research efforts, there is currently no standard for specifying model transformations. In the context of Aniketos, model transformation is required to translate security requirements models, or parts of them, into security contracts and into mappings between threats and services.

As outlined by Czarnecki and Helsen in [77], model transformations can be performed for different purposes: generating lower-level models, mapping and synchronizing models, creating query-based views of a system, model evolution tasks, reverse engineering. Among these purposes, the following two are the most relevant:

- Generation of lower-level models, or code, from higher-level models [79]. For instance, requirements models can be transformed to architectural models. This kind of model transformation is very relevant for Aniketos, where security requirements are transformed into lower-level artefacts;

- Mapping and synchronizing among models at the same or different levels of abstraction [80]. This type of transformation ensures a bidirectional relation between models: changes in either model are propagated to the other model. This type of transformation ensures the consistency between different models. In Aniketos, this is important to check if changes in security contracts or new threats have any impact on requirements.

While some model transformation approaches are proposed in research literature (e.g. VIATRA Visual Automated model TRAnsfomrations [81], ATL Atlas Transformation Language [82], and MOLA MOdel transformation LAnguage [83]), some others are implemented in open-source (e.g. AndroMDA [84], JET Java Emitter Templates [85]) or commercial tools (e.g. MetaEdit+ [86]).

There are two main categories of transformation approaches: model-to-model and model-to-text. We do not focus here on the latter category; despite of its importance for documentation purposes, this is not the primary concern in Aniketos. We review the main approaches to model-to-model transformation, as suggested by Czarnecki and Helsen in [77]:

- **Direct manipulation**: they provide an internal model representation and some APIs to manipulate models. Users have to implement transformation rules, scheduling, tracing, etc. This is the case, for instance, of JMI [87];

- **Structure-driven**: these approaches let the user focus on transformation rules only. First, the hierarchical structure of the target model should be defined. Second, the attributes and references in the target should be set. This approach performs a sort of copy between the source and the target model. Examples of this category are OptimalJ [88] and the QVT submission by Interactive Objects and Project Technology [89];

- **Operational**: they extend the metamodeling formalism with facilities for expressing computations. These approaches refine direct manipulation approaches. For example, QVT operational mappings and Kermeta [90] belong to this category.

- **Template-based**: Model templates are a special kind of model that includes metacode in charge of computing the variable parts of the resulting template instances. Model templates are expressed in
the concrete syntax of the target language, so that the user can predict the result of template usage. An example is that by Czarnecki and Antkiewicz [91]

- **Relational**: these approaches are declarative and are founded on mathematical relations between source and target elements. They can be seen as a sort of constraint solving. Logic programming is a candidate technology to support the execution of these approaches. An example is that by Gerber et al. [92]: they explore the usage of the Mercury logic programming and of F-logic to implement transformations.

- **Graph-transformation-based**: these approaches rely on theories about graph transformation. This requires source and target models to be expressed as graphs. Many successful transformation approaches belong to this family, such as VIATRA [81], MOLA [83], and Fujaba [93].

- **Hybrid and others**: many approaches combine features from different categories. For example, transformation rules in ATL can be fully declarative, hybrid, or fully imperative. An interesting example of model transformation is provided by the Extensible Stylesheet Language Transformation (XSLT) [94]. This language enables the transformation between XML-based models via serialization and metadata interchange. However, implementations of this approach demonstrated to have scalability limitations.

### 4.3 Trustworthiness prediction module

#### 4.3.1 Role-based Trust Management with quantitative notion of trust (CNR)

<table>
<thead>
<tr>
<th>Name</th>
<th>Role-based Trust Management with quantitative notion of trust (RTML with weights)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool/Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>The proposed tool/technique is a framework including a language for weighted credentials able to embody role and attribute based access control, enhanced with delegation capabilities with quantitative notions of trust. The trust metrics are expressed through semi rings.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td>Support area</td>
<td>RS - Trustworthiness monitoring and evaluation</td>
</tr>
<tr>
<td>Developed by</td>
<td>CNR [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Work in progress</td>
</tr>
<tr>
<td>References</td>
<td>D. Fais et al., 2009 [95]</td>
</tr>
<tr>
<td>References</td>
<td>M. Colombo et al., 2009 [96]</td>
</tr>
<tr>
<td>References</td>
<td>S. Bistarelli et al., 2008 [97]</td>
</tr>
<tr>
<td>References</td>
<td>S. Bistarelli et al. 2, 2008 [98]</td>
</tr>
<tr>
<td>References</td>
<td>M. Colombo et al., 2007 [99]</td>
</tr>
<tr>
<td>References</td>
<td>F. Martinelli et al., 2007 [100]</td>
</tr>
<tr>
<td>References</td>
<td>F. Martinelli, 2005 [101]</td>
</tr>
<tr>
<td>System requirements</td>
<td>JVM (also runs on Symbian/Nokia for mobile devices)</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Usable for Aniketos</td>
</tr>
</tbody>
</table>
1. Short description
The proposed tool/technique (RTML with weights) is a framework including a language for weighted credentials able to embody role and attribute based access control, enhanced with delegation capabilities with quantitative notions of trust. The trust metrics are expressed through semi rings.
This approach brings together in a hybrid framework two usual notions of trust:
- Trust based on credentials, policies,
- Trust based on recommendation/reputation.
The framework is extremely flexible being parameterized by the appropriate semiring.

2. Making RTML with weights engine part of the Aniketos platform
As said, RTML with weights is a framework which considers two notions of trust: 1) trust based on credentials and 2) trust based on reputation. These two notions of trust are planned to be implemented as a part of WP2 (Tasks 2.1 and 2.2). Therefore, the ideas from RTML with weights are very useful for Aniketos.
The engine can be used both to represent and propagate trust information. It is also an engine for trust calculation and it can be both used for access control during run time service operation as well as during design time for expressing trust measures.

3. Development roadmap
In terms of big steps, we can consider the following phases:
- Completely develop the framework, with all the usual mechanisms as access control engine, trust metrics and trust negotiation engines.
- Embed it in the security by contract.
- Embed it in the Aniketos run time framework (for access control and for reliably transmitting trust information, stored in signed credentials).

4.3.2 Trust Manager (TSSG)
The goal of the Trust Manager (see section 4.4.3) is to collect the information about provided QoS from a service provider and to compute a trustworthiness rating. These ideas can be used as a part of trustworthiness prediction (T2.1 and T2.2) for determining trustworthiness of composable services and mechanisms for QoS monitoring.

4.3.3 STACS: Scalability Testbed for Access Control performance of Services (TSSG)
STACS (see section 4.6.5) may be used to predict access control performance. Since this is part of the overall service performance, it affects usability of the service and hence such predictions could form part of a usability-based Trust score for a given service.

4.3.4 Other related work to Trustworthiness prediction module
Identification of relevant technology for this module is scheduled later in the project once the scope of service trustworthiness within Aniketos platform is completely defined within WP2 according to the requirements in WP1.
### 4.4 Monitor trustworthiness module

#### 4.4.1 Role-based Trust Management with quantitative notion of trust (CNR)

RTML with weights (see section 4.3.1) is a framework which has a language for weighted credentials able to embody role and attribute based access control, and which is also able to formalize trust predicates. This language can be used in WP2 for creation of the monitor trustworthiness module.

#### 4.4.2 xESB (UNITN)

<table>
<thead>
<tr>
<th>Name</th>
<th>xESB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Perform control enforcement at the message level.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS – Trustworthiness monitoring and evaluation</td>
</tr>
<tr>
<td>Developed by</td>
<td>University of Trento (UNITN) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>In-house extension of publicly available ESB, applied to EU projects, disseminated through publications</td>
</tr>
</tbody>
</table>
| References | MASTER project [10]  
|            | G. Gheorghe et al., 2010 [102] |
| System requirements | Services using a communication platform implemented as a SOA using an ESB JBI (Java Business Integration) is the standard used to deploy and manage services deployed on an ESB. |
| Miscellaneous | N/A |

1. **Short description**

xESB is an instrumented JBI ESB (Enterprise Message Bus) for the enforcement of security policies that are organization wide. xESB is able to enforce both access and usage control policies. The rich enforcement semantics of xESB allows not only to reject ESB messages that violate a policy but also to compensate that violation. xESB also introduces and supports indicators, which aim at helping security administrators analyze and derive useful information about policy violations and their impact to the overall security of the organization.

2. **Making “xESB” part of the Aniketos platform**

xESB is used to monitor and enforce preventive and reactive policies, both for access control and usage control policies, and both inside one domain and between domains. It considers the set of policies that should be applied and performs checks on any message that is intercepted, to check compliance with the set of policies. xESB helps not only to reject ESB messages that violate a policy, but also to compensate that violation. The usage of indicators that help the security administrator analyze and derive useful information about policy violations and their impact to the overall security of the organization is also very important.

Therefore, when performing trustworthy service composition, xESB might operate as part of Aniketos to check policies and prevent compositions that violate such policies.

3. **Development roadmap**

xESB can serve as a baseline for WP2, Task 2.1 where trust based security policies are monitored to ensure the trustworthiness of the composite services.

The tool can also be exploited in performing policy based control of secure service composition, WP3, Task 3.1 and 3.4.
### 4.4.3 Trust Manager (TSSG)

<table>
<thead>
<tr>
<th>Name</th>
<th>Trust Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool/technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>Management of ratings and trust levels based on events and metrics received.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS - Trustworthiness monitoring and evaluation</td>
</tr>
<tr>
<td></td>
<td>RS - Composite service adaptation and recomposition</td>
</tr>
<tr>
<td>Developed by</td>
<td>TSSG [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype</td>
</tr>
<tr>
<td>References</td>
<td>Part of Information Management and Metrics Monitoring middleware in CoMiFin project [103]</td>
</tr>
<tr>
<td></td>
<td>Lodi et al., 2010 [104]</td>
</tr>
<tr>
<td>System requirements</td>
<td>JRE 1.6, JBoss AS, MySQL</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Proprietary</td>
</tr>
</tbody>
</table>

#### 1. Short description
The Trust Manager is a server application. It continuously receives alerts and metrics from QoS monitoring and Event Processing middleware components regarding agents and stores ratings locally in a database. It computes trust levels based on those ratings and forwards the trust levels to subscribing parties in a dynamic environment.

#### 2. Making the Trust Manager part of the Aniketos platform
The Trust Manager may be modified and progressed based on the requirements of the platform to be used as part of the toolset for monitoring the trustworthiness of services and triggering service recomposition and adaptation. The focus of this tool is on trustworthiness evaluation based on service reputation and QoS rather than service security provisions.

#### 3. Development roadmap
The tool does not specifically support trust management of run-time service compositions currently. The aim is to modify the tool to allow monitoring and evaluation of composed services’ trust and reputation and fair sharing of feedback on the trust and reputation between component services e.g. not reducing trust of a component service that was not the cause of the bad ratings.

#### 4.4.4 Other related work to Monitor trustworthiness module
Identification of relevant technology for this module is scheduled later in the project once the scope of service trustworthiness within Aniketos platform is completely defined within WP2 according to the requirements in WP1.
### 4.5 Contract negotiation module

#### 4.5.1 MUSIC studio and middleware (SINTEF)

MUSIC platform (see section 5.3.1.1) provides runtime support for SLA negotiation, creation and monitoring.

#### 4.5.2 Security by Contract with Trust for mobile systems (CNR)

<table>
<thead>
<tr>
<th>Name</th>
<th>Security by Contract with Trust for mobile systems (SxCxT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>SXc paradigm addresses the enforcement of secure interaction between a mobile application and its platform (mobile device) via (semi-)automated contracting. The contract is an agreement on the permissible security-relevant operations the application may perform. The SxCxT paradigm extends the SxC with (i) a run-time contract monitoring for allowing to manage applications trust level and (ii) a trust management mechanism able to guarantee that a mobile application can be safely executed on mobile devices, according to trust recommendations.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS - Composite service analysis and preparation</td>
</tr>
<tr>
<td></td>
<td>RS - Trustworthiness monitoring and evaluation</td>
</tr>
<tr>
<td></td>
<td>RS - Runtime validation of secure service behaviour</td>
</tr>
<tr>
<td>Developed by</td>
<td>CNR [Aniketos] and University of Trento (UNITN) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Security by Contract is a mature work published in international journals and conferences. It is also exploited by the SecureChange Project. Several publications about Security by Contract with Trust are already published in international journals and conferences. It is exploited in the EU CONNECT project for distributed systems. The integration with different trust models is still a work in progress. Work in progress, Prototype</td>
</tr>
<tr>
<td>References</td>
<td>SecureChange project [105]</td>
</tr>
<tr>
<td></td>
<td>N. Dragoni et al., 2010 [106]</td>
</tr>
<tr>
<td></td>
<td>N. Dragoni et al., 2009 [107]</td>
</tr>
<tr>
<td></td>
<td>G. Costa et al., 2010 [108]</td>
</tr>
<tr>
<td></td>
<td>N. Bielova et al., 2009 [109]</td>
</tr>
<tr>
<td></td>
<td>N. Dragoni et al., 2008 [110]</td>
</tr>
<tr>
<td></td>
<td>G. Costa and I. Matteucci, 2010 [111]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>N/A</td>
</tr>
</tbody>
</table>
1. Short description

The notion of Security-by-Contract (SxC) denotes an agreement between a mobile application and a platform, a mobile device. This agreement specifies the permissible interaction between the application and its host device. To accomplish the contract, the application has to disclose its security-relevant features to any platform where it is deployed. The platform has to check these features against its security policy and decide upon whether to sign the contract and accept the installation of the application. It is also possible that the platform accepts some features and neglects others. This selection of features leads to customizing the mobile application to be consistent with its platform choice.

The Security-by-Contract-with-Trust (SxCxT) extends the SxC with a run-time contract monitoring for allowing managing the application trust level. In particular, the contract monitoring checks if the actual behaviour of the application is compliant with its contract and the trust management rewards and penalises the application depending on the answer of that check. It this way, we have a unified framework for dealing with both security and trust on (i) mobile devices and (ii) distributed systems.

Indeed, the SxCxT denotes a paradigm with a trust management mechanism able to guarantee that a mobile application can be safely executed on mobile devices, according to applications trust recommendations.


SxCxT has the primary goal to enforce the contract between a host platform and an application. In principle, the technique can be adapted to composition of services. A composite service will incorporate several service components that are not necessarily owned and/or known by the service developers. Moreover, the lifelong re-composition of services will make it inevitable to import new services to adapt to changes and optimize performance and other quality measures. This means that a contract has to be signed between the composite (host) service and the sub (component) service.

Security by Contract could be applied when Aniketos composes services. A service developer should provide a description of the security-related operations that the service executes. The description has to specify the physical and information resources that the service will use. It has also to specify the interaction with other services that are probably external to the party who is using the service. By signing the contract between a composite service X and a service Y, it means that the invoked service Y code complies with the stated claims on its security-relevant behaviour.

At runtime, Aniketos has to monitor the behaviour of a service and ensure that it does not violate the contract and take the proper action when such occurs. If the service is given access to some resources as specified in the contract then Aniketos has to monitor if the service did not misuse them.

The level of trusting a service could be evaluated based on the compliance between its operation and the signed contract. Aniketos may benefit from Security by Contract with Trust to compute trust on the basis of service adherence of a contract. Indeed, the Security-by-Contract-with-Trust framework well fits with the Aniketos goal of ensuring the trustworthiness of composite services created with the Aniketos platform.

3. Development roadmap

This is directly relevant to WP2 and specifically to:

T2.2: the SxCxT could be used as a starting point for defining patterns and guidelines for establishing and managing trust among end-users and composite services.

T2.3: the SxCxT could be extended and adapted for dealing with the composition of services in a dynamic and distributed system. The services will expose their contract and can be composed by using them.

T2.4. the SxCxT already incorporates a mechanism for the contract monitoring according to which the level of trust of the application is measured and managed. At the present stage, it is developed for mobile applications. It has to be adapted for dealing for a dynamic service composition. This means that the SxCxT will be extended with monitoring mechanisms to ensure the compliance between the
36

D1.1: Consolidation of existing work

service execution and the contract. According to the compliance with the contract, the trust level of a service will be adjusted;

T2.5: this activity is to develop formal verification of services compliance with the contract. Therefore, it will implement and refine the results obtained in T2.3 and T2.4 that concerns accomplishing security by contract between services and monitoring it and adjusting the trust level accordingly. The SxCxT framework has a mechanism for the contract-policy matching that could be consider a starting point for achieving the goal required in this Task also for the composition of services.

4.5.3  SLA Management framework (ATOS)

<table>
<thead>
<tr>
<th>Name</th>
<th>SLA Management framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>SLA Negotiation, Monitoring and Accounting tools</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS - Composite service analysis and preparation</td>
</tr>
<tr>
<td></td>
<td>RS - Trustworthiness monitoring and evaluation</td>
</tr>
<tr>
<td></td>
<td>RS - Runtime validation of secure service behaviour</td>
</tr>
<tr>
<td></td>
<td>RS - Composite service adaptation and recomposition</td>
</tr>
<tr>
<td>Developed by</td>
<td>Atos Origin and FP6-ICT-2006 BEinGRID project partners</td>
</tr>
<tr>
<td>Maturity</td>
<td>No commercial installation of this asset exists. The maturity level is comparable to TRL 6 of the NASA TRL scale (“prototype demonstration in a relevant environment”).</td>
</tr>
<tr>
<td>References</td>
<td>BEinGRID EU FP6 project [112]</td>
</tr>
<tr>
<td>System requirements</td>
<td>JRE, GLOBUS</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Atos Origin owns IPR to the SLA Monitoring and Evaluation, and shares the IPR of the SLA accounting with HLRS. Atos Origin alone has the rights for exploitation and use of the asset as a whole (negotiation-evaluation-accounting).</td>
</tr>
</tbody>
</table>

1. Short description

Service Level Agreements (SLAs) are probably the most important documents in every business-aware framework – at least they should be. Every distributed piece of software faces the problem to assure a decent Quality of Service (QoS). An elegant solution is to describe the QoS within an SLA, linked to a legally binding contract. The BEinGRID solution offers an implementation supporting the negotiation, evaluation and accounting of SLAs. SLAs are of vital importance for modern business models, such as Utility Computing, Software as a service (SaaS) or Infrastructure as a service (IaaS), due to the requirement for observing a certain QoS when providing a service.

2. Making SLA Management Framework part of the Aniketos platform

Electronic services often need to be delivered at a guaranteed service level. Service Level Agreements can be used to address this by defining Quality of Service (QoS). The delivery of run-time composed dynamic services, calls for a more agile system based on dynamic SLAs. The dynamic SLAs proposed by BEinGRID were deemed quite specific and novel based on the requirements elicited from several Business Experiments.
3. Development roadmap

Aniketos project aims to provide developers with a framework that supports the design-time creation and run-time composition of secure dynamic services, where both the services and the threats are evolving. In this context, the management, execution and maintenance of SLAs in the upcoming service-oriented IT landscape, such as the one suggested by Aniketos, need new levels of flexibility and automation not available with the current technology.

The SLA Framework resulting of the BEinGRID project was developed on the top of the GLOBUS toolkit. Consequently, the first, and mandatory, development needed in this area is to adapt the framework to be able to work with WS-* standard stack.

Additionally, the SLA Framework developed in BEinGRID provides sufficient functionality to handle a simple SLA situation (negotiation, monitoring and accounting). However, in the context of Aniketos a more complex functionality is required. One example of this missing functionality that should be addressed inside Aniketos is the capability of renegotiation. Renegotiation is currently a research topic that was an unclear during the lifecycle of BEinGRID project; therefore it should be addressed in more detail taking into account the Aniketos architecture and scenarios.

4.5.4 STACS: Scalability Testbed for Access Control performance of Services (TSSG)

Candidate services might be required to participate in access control performance prediction experiments. STACS (see section 4.6.5) would facilitate such experiments as part of the acceptance process for services. The resulting performance predictions might form part of the contract between services and Aniketos platform.

4.5.5 Other related work to Contract negotiation module

4.5.5.1 Security-Oriented Service Composition and Evolution

<table>
<thead>
<tr>
<th>Name</th>
<th>Security-Oriented Service Composition and Evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>A framework for service-oriented software service composition and evolution. Also, a semantic model for specifying the security objectives and properties at the service and system level.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>RS - Runtime validation of secure service behaviour</td>
</tr>
<tr>
<td>Developed by</td>
<td>Swinburne University of Technology and Qatar University [externals]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype</td>
</tr>
<tr>
<td>References</td>
<td>Han and Khan, 2006 [113]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Most of the text has been taken from the paper (reference above)</td>
</tr>
</tbody>
</table>

1. Short description

This methodology is used as a security-oriented negotiation approach to service composition and evolution. The key building blocks of the framework are specifying the security objectives and properties at the service and system levels, the negotiation and re-negotiation technique for service composition and evolution, and checking the security compatibility between services and the satisfaction of system-level security goals. It focuses on developing methods that allow system developers to design required security into service compositions with predictability and to maintain or adapt service compositions in changed security context.
2. Making Security-Oriented Service Composition and Evolution framework part of the Aniketos platform

The framework could be helpful for Aniketos developers to design required security into service composition and to maintain or adapt service compositions in changed security context. Furthermore, it could be used to specify security objectives and properties at both system and service levels.

3. Development roadmap

We believe that Definition of security objectives and properties phase of the framework could potentially be incorporated into the socio-technical security modelling language and tool (D1.3 and D1.4). Security contract negotiation for service composition phase could be helpful in D2.2 and D2.3. Finally, Security contract re-negotiation for system evolution might be usefully applied or developed as part of WP3 (task 3.3, 3.4).

4.6 Security property determination module

4.6.1 Modular plans for secure service composition (CNR)

Name: Modular plans for secure service composition
Type: Technique
Purpose: Security-driven generation of composition plans
Support area:
- DS - Security property definition and evaluation
- DS - Composite service analysis and preparation
- RS - Runtime validation of secure service behaviour
Developed by: CNR [Aniketos]
Maturity: Work in progress
References:
- G. Costa et al., 2009 [116]
- G. Costa et al. 2, 2010 [117]
- G. Costa et al. 3, 2010 [118]

System requirements: N/A
Miscellaneous: N/A

1. Short description

Web services behave differently according to the way they compose at runtime. Whenever a request is performed, the actual server must be chosen among a list of concurrent providers. A fundamental criterion for deciding which interactions are allowed is the respect of the security policies defined by the services.

Security policies are defined and included in the structure of a service at development time. Clearly, each service has its own security requirements and is usually unaware about the others’ policies.

Our technique aims at verifying which compositions are valid for a service. In this way services can compose dynamically only using safe interactions. This result is obtained by model checking the formal descriptions of the services, i.e. their contracts, against the involved security policies.

Moreover, through a partial evaluation step, we produce special requirements that the services expose to their clients in order to obtain a faster verification process. In this way we also improve the standard security-by-contract (SxC) technique with a policy negotiation phase that extends the security requirements of a service over its clients.
2. Making Modular plans for secure service composition part of the Aniketos platform

The technique presented above widely addresses many of the WP3 objectives. Indeed, our security process involves both design-time and runtime aspects. In particular, the analysis of the security policies can be used to drive and optimise the computation of compositional plans for the service networks (Task 3.1). Moreover we can automatically translate security automata into runtime monitors to be deployed over the network in order to analyse and watch the correct behaviour of the running services (Task 3.2).

3. Development roadmap

This technique can be integrated in the design-time support of the Aniketos platform. This way behavioural abstractions, i.e. contracts, can be automatically generated within the development process. At this stage the platform should also provide a support for designing the necessary security policies and for applying them to the service structure. This process is relevant to the Task 3.1.

A further development would be implementing the instantiation procedure for generating runtime monitors starting from the security policies. These monitors should be organised so that they can be deployed in the distributed environment where services run.

Another step is the investigation of the advantages, in terms of security and performances, of the partial evaluation approach to the synthesis of services’ differential controllers. Both the last two points are specific to the Task 3.2.

Finally, these efforts must lead to the design and implementation of a working prototype according to the objectives of Task 3.5.


<table>
<thead>
<tr>
<th>Name</th>
<th>A Framework For Quantitative Security Analysis of Complex Business Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool/Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>The proposed technique aggregates security indicators to determine the impact of components on the overall business process using the structure of the enterprise (including the business process itself).</td>
</tr>
<tr>
<td>Support area</td>
<td>Composite service analysis and preparation</td>
</tr>
<tr>
<td>Developed by</td>
<td>University of Trento (UNITN) [Aniketos]-CNR [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Work in progress</td>
</tr>
<tr>
<td>References</td>
<td>F. Massacci and A. Yautsiukhin, 2007 [120]</td>
</tr>
<tr>
<td></td>
<td>F. Massacci and A. Yautsiukhin 2, 2007 [121]</td>
</tr>
<tr>
<td>System requirements</td>
<td>Business Process Visual Architect (BPVA) 2.4 is required. MySQL database is required.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Free, for non commercial usage.</td>
</tr>
</tbody>
</table>

1. Short description

The framework considers a three-layered model of an enterprise. Indicators for each atomic element on each layer of the model are combined according to the structure of the enterprise forming an overall indicator for a specific business process. Such aggregation provides an objective way of aggregating indicators. The proposed analysis allows efficient comparison of alternative designs and selects the design for the service which has the best protection, i.e., has the best assurance indicator.

The framework takes into consideration that the flow of business process affects aggregation of security indicators and includes an explicit way of analysing security of the business process structure.
The framework also supports outsourcing and allows correcting values according to the belief of client in genuine fulfilment of the agreed requirements by the contractor.

2. Making this Framework For Quantitative Security Analysis of Complex Business Systems part of the Aniketos platform

Currently, Business Process Modeling Notation (BPMN) by OMG [122] is used as a required model for transformation to a hypergraph. If another modelling technique is used the transformation algorithm has to be adapted. The same is valid for storage of information about the model (currently MySQL is used for these purposes).

First of all, the metrics (indicators) useful for description of quality of protection level have to be determined. These indicators will be the premise values the technique requires. Moreover, the nature of the indicators determines how the indicators have to be aggregated and compared. The aggregation and comparison parts of the algorithms have to be reconsidered correspondingly.

Current support of determination of trust is weak. A new way of correcting expected values is required according to the way of defining trust in the project.

Complex threats which relate to a group of activities could be considered. Currently, we consider only threats which affect operation of one activity, and, thus, compromise the whole business process. More complex threats may affect operation of a sub-process, but not of an activity (e.g., violation of separation of duty principle).

Currently, the supporting tool does not support on-line algorithm which allows to re-evaluate the model when a small changes occurred (e.g., an activity become unavailable and another one should be added).

If the supporting tool is used then its maturity (e.g., failure tolerance) should be improved.

3. Development roadmap

The first step is to adapt the proposed technique to security analysis during a design time (Task 3.1). The first step also includes definition of metrics which can be taken from contracts (or QoP proposals) and monitored.

The next step should be devoted to the adaptation of basic algorithms (transformation of a business process model to a hypergraph and defining functions for aggregation and comparison). This step should be performed in close cooperation with Task 2.1 (to define trust metrics) and Task 1.4.

Then, the technique could be made useful for run-time security analysis (Task 3.2). A new (but already existing) on-line algorithm for re-computation of the best security alternative should be implemented. Also this task should be in compliance with Tasks 3.5 and 4.4 which provides information about changes in provided service levels. The results of the task should be provided to Task 3.3 for reconfiguration of the service according to the selected alternative.

4.6.3 Property and Policy Tools (LJMU)

The PropertyInterface tool (see section 4.1.13) allows sets of properties to be defined based on various criteria. The purpose is not to define specific properties that would apply to a given service, but rather a range of property classes that might apply. The actual properties that apply to a given service would then be selected from within this range (where some of these properties might also be dynamic and change over time). Thus, the Property and Policy Tools could provide a baseline capability for determination of the security properties of individual services as well as the overall properties of the federated composed services.
### 4.6.4 Platform for Run-time reconfigurability of security (ATOS)

<table>
<thead>
<tr>
<th>Name</th>
<th>Platform for Run-time reconfigurability of security (PRRS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>A platform that provides applications and systems with security solutions during run time, upon request</td>
</tr>
</tbody>
</table>
| Support area | DS - Security property definition and evaluation  
RS - Trustworthiness monitoring and evaluation  
RS - Runtime validation of secure service behaviour |
| Developed by | Atos Origin [Aniketos] |
| Maturity | No commercial installation of this asset exists. The maturity level is comparable to TRL 6 of the NASA TRL scale (“prototype demonstration in a relevant environment”). |
| References | Serenity project [19]  
Serenity public deliverables related to PRRS [123] |
| System requirements | This asset is developed in Java, making use of client-server architecture, implementing the SERENITY concepts of Security and Dependability (S&D) patterns and mechanisms. |
| Miscellaneous | This asset is owned by Atos Origin. Complementary information and systems that may be required to operate this asset are co-owned by Atos Origin and other SERENITY project partners. |

1. **Short description**

The Platform for Run-time reconfigurability of security (PRRS) provides run-time management of Security and Dependability (S&D) solutions and monitoring of the system context. These functionalities are provided by means of Executable Components (a service is listening to the applications’ requests) that respond to applications security requirements.

The core of PRRS was developed within SERENITY EU FP7 project and is available from the SERENITY website [123].

2. **Making Platform for Run-time reconfigurability of security (PRRS) part of the Aniketos platform**

PRRS tool could be used in Aniketos platform both for Run-time (RS) and for Design Time (DS) support. It provides trustworthiness monitoring and evaluation at RS, as well as the validation of secure service behaviour. With regard to the second issue, any application or service could access to PRRS tool in order to request a security solution at Design Time.

3. **Development roadmap**

PRRS chooses security patterns based on service and application requirements, while WP3 chooses suitable services based on their security properties. Although they sound like the other way round to each other, PRRS could provide WP3 with well-defined security patterns and the fundamental mechanism of system context monitoring. WP3, on the other hand, could extend the PRRS’s ability to provide security solutions for more complicated application, i.e. composed services. The platform is developed in Java and that makes it easy to combine with other parts of the Aniketos system.

PRRS tool doesn’t support SOAP currently, and therefore one interesting proposal for Aniketos project could be to add the SOAP interface, in order to make the requests from the applications/services to PRRS tool, and send back the responses.
4.6.5 STACS: Scalability Testbed for Access Control performance of Services (TSSG)

<table>
<thead>
<tr>
<th>Name</th>
<th>STACS: Scalability Testbed for Access Control performance of Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Performance Testbed</td>
</tr>
<tr>
<td>Purpose</td>
<td>A testbed facilitating performance and scalability experiments on Access Control systems</td>
</tr>
<tr>
<td>Support area</td>
<td>DS – Security property definition and evaluation</td>
</tr>
<tr>
<td>Developed by</td>
<td>TSSG [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Work in progress</td>
</tr>
<tr>
<td>References</td>
<td>Butler et al., 2010 [119]</td>
</tr>
<tr>
<td>System requirements</td>
<td>JRE 1.5, R 2.10.1</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Open source (licence TBD)</td>
</tr>
</tbody>
</table>

1. Short description
Policy-based access control is a widely used methodology. Research to date has focused on access model correctness, i.e., can each Subject perform approved Actions upon permitted Resource(s) (Services), with no more and no less rights than were intended by the policy author. More recently, concerns have been raised about the scalability of policy based access control, particularly as fine-grained access control is applied to more use cases such as entitlement management and social network interactions. In the special case of XACML-encoded policies, several improvements have been proposed and are typically justified by comparison with Sun’s reference PDP implementation. However it is difficult to compare different improvement proposals because experimental conditions are not easily standardised. The goal of STACS is to provide such a standard experimental testbed to enable effective comparisons to be made.

2. Making STACS part of the Aniketos platform
At service composition design time, the Aniketos platform needs to be able to predict the system impacts (load, costs, etc.) of different service compositions, with a view to optimising a set of criteria. STACS could play a part in enabling the Aniketos platform to include performance and scalability criteria in this mix. In principle, services could submit themselves for access control scalability evaluation. This data could be stored (and updated as necessary) and used by the Aniketos design time service composition broker to make more informed decisions. The core security principle we wish to uphold is availability of composed services.

3. Development roadmap
To improve its utility for Aniketos, we propose extending STACS in the following fashion:
- Make STACS truly client-server rather than reading control files as at present. This would be more representative of composed web services.
- Make it easier to add new services
- Extend the testbed to support measurements on composed services, not just single services.
These are some preliminary ideas and the list above is not exhaustive.

4.6.6 Universal Service Description Language (SAP)
USDL (see Section 5.2.1.2) is a language for specifying services and service interfaces on many different abstraction levels ranging from abstract business requirements to technical system
requirements. The Security Property Determination Module benefits particularly from the means for specifying security requirements that USDL provides.

4.6.7 Avantssar platform for validating trust and security properties (SAP)
The AVANTSSAR Platform (see Section 4.7.4) provides means for analyzing the security properties of composed services. As such, AVANTSSAR contributes first means for determining the security properties of combined services to the Security Property Determination Module.

4.6.8 Other related work to Security property determination module

4.6.8.1 Security-Oriented Service Composition and Evolution
One key aspect of this framework (see section 4.5.5.1) is to specify the security objectives and properties at the service and system level. This could be useful for security property determination module.

4.6.8.2 Composing Security-Aware Software

<table>
<thead>
<tr>
<th>Name</th>
<th>Composing Security-Aware Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>A framework for characterizing the security properties of software components, and how to characterize the entire system’s security properties. Also, how to make these characterized security properties available at runtime.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td>Developed by</td>
<td>University of Western Sydney [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype</td>
</tr>
<tr>
<td>References</td>
<td>Khan and Han, 2002 [124]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Most of the text has been taken from the paper (reference above)</td>
</tr>
</tbody>
</table>

1. Short description
The Composing Security-Aware Software framework addresses how to characterize the security properties of components, how to analyze at runtime the internal security properties of a system comprising several atomic components, how to characterize the entire system’s security properties, and how to make these characterized properties available at runtime.

2. Making Composing Security-Aware Software framework part of the Aniketos platform
The tool could be useful for developers to identify the various security properties of individual components as well as capturing the security properties of the overall composite system.

3. Development roadmap
This methodology might be usefully applied or developed as part of WP3 secure service composition (task 3.1, 3.2, 3.3).

4.6.8.3 Enforcing Secure Service Composition

<table>
<thead>
<tr>
<th>Name</th>
<th>Enforcing Secure Service Composition (ESSC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Technique</td>
</tr>
</tbody>
</table>
### Purpose
Composite services that respect given security requirements.

### Support area
DS - Security property definition and evaluation

### Developed by
University of Pisa, Italy [external]

### Maturity
Prototype

### References
- M. Bartoletti et al., 2005 [114]
- M. Bartoletti et al., 2006 [115]

### System requirements
N/A

### Miscellaneous
The work is quite theoretical; it will be difficult to implement it without support from the original authors.

## 1. Short description
Enforcing Secure Service Composition (ESSC) is proposed by M. Bartoletti et al., 2005 [114] to study secure composition of software. As described in M. Bartoletti et al., 2006 [115], ESSC extends the $\lambda$-calculus with primitives for invoking services that respect given security requirements. Security-critical code is enclosed in policy framings which enforce safety and liveness properties of execution histories. The actual histories that can occur at runtime are over-approximated by a type and effect system. These approximations are model-checked to verify policy framings within their scopes. This allows for removing any runtime execution monitor, and for selecting those services that match the security requirements.

## 2. Making ESSC part of the Aniketos platform
ESSC could be considered in Aniketos design time as an existing interesting material. It provides a technique to choose the services that must satisfy certain security constraints. However, it may not be included eventually, as there are other modelling tools in Aniketos partners’ background, described in this document, which are more concrete and easier to get and use.

## 3. Development roadmap
The work is at a relatively high level and very theoretical. Aniketos could make some concrete work on it to integrate and implement the theory in a prototype environment. However, because it is designed for static analysis, to make it fit into runtime environment like Aniketos is not a straightforward task and the help of the original authors will probably be needed.
4.7 Security verification module

4.7.1 Usage Control Authorization System (CNR)

Name: Usage Control Authorization System  
Type: Tool/Technique  
Purpose: The proposed framework is a Policy Decision Point (PDP) that evaluates security policies expressed with POLPA language. POLPA is an operational policy language that allows expressing usage control policies. In particular, POLPA allows representing the continuous control and the access revocation.  
Support area: RS - Runtime validation of secure service behaviour  
Developed by: CNR [Aniketos]  
Maturity: Work in progress  
References:  
Colombo et al., 2009 [127]  
Sandhu and Park, 2004 [128]  
Martinelli and Mori, 2010 [129]  
Martini et al., 2010 [130]  
Colombo et al., 2007 [131]  
System requirements: None  
Miscellaneous: Usable for Aniketos

1. Short description

The Usage Control Authorization System is a framework that implements the usage control model (Sandhu and Park, 2004 [128]) and consists of a language (POLPA) for expressing usage control policies, and a Policy Decision Point (PDP) that evaluates the security policy for deciding whether an access request is allowed or not.

The main features of the framework is that it implements continuous usage control, i.e. it is able to express and enforce security policies where the controls are executed not only before authorizing the access, such as in traditional access control systems, but also when the access is in progress (ongoing controls). If the ongoing controls detect a policy violation, the framework revokes the access while it is still in progress.

The Usage Control Authorization system has been adopted to enhance security in several scenarios. As an example, in Martinelli and Mori, 2010 [129] the Usage Control Authorization system has been used for regulating the usage of Grid computational services, while in Martini et al., 2010 [130] has been exploited to control the usage of network services provided by Service Oriented Optical Networks.

Moreover, as shown in Colombo et al., 2007 [131], the Usage Control Authorization system can be integrated with a Trust Management framework, such as the RTML one, to exploit Trust information in the decision process.

2. Making Usage Control Authorization System part of the Aniketos platform

The UCON system can be mainly exploited in WP3 and WP4 as a mechanism for enforcing security policies, in particular usage control authorization policies, during the composition of services.
In particular it can be useful for assuring service runtime adaptability. Indeed, at runtime it can be used for monitoring the usage of services and, then, for assuring that any possible adaptation of the service composition is still compliant with the security policies required.

3. Development roadmap

We consider the following steps:

- Development of the Policy Enforcement Points (PEPs) related to the security relevant actions that require to be monitored. These PEPs must be able to communicate with the Usage Control PDP;
- Development of Usage Control policies that express the interesting security requirements;
- Integration of the Usage Control Authorization System in the Aniketos run time framework.

4.7.2 MATTS (Mobile Agent Topology Test System) (LJMU)

<table>
<thead>
<tr>
<th>Name</th>
<th>MATTS (Mobile Agent Topology Test System)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool + Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>Analysis and modelling of secure component composition</td>
</tr>
<tr>
<td>Support area</td>
<td>RS - Runtime validation of secure service behaviour</td>
</tr>
<tr>
<td>Developed by</td>
<td>LJMU [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype</td>
</tr>
<tr>
<td>References</td>
<td>Zhou et al., 2010 [30]</td>
</tr>
<tr>
<td></td>
<td>Zhou et al., 2008 [132]</td>
</tr>
<tr>
<td>System requirements</td>
<td>Windows (MATTS), .Net (CompositionClient)</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Currently internal/proprietary code, but published techniques.</td>
</tr>
</tbody>
</table>

1. Short description

The nature of trustworthy service composition means that it inevitably involves complex interactions between services in diverse environments. Allowing for complex scenarios of interactions and related analysis techniques to be built up and tested quickly and easily is therefore important for better understanding the concepts involved. The MATTS set of tools can provide a platform for the process of exploring, developing and analysing service compositions. The tools have been designed specifically for this task, with an emphasis on Systems-of-Systems security analysis and secure component composition techniques.

The platform is split into a number of software components, most notably the MATTS modelling software and the CompositionClient for interacting with it using real devices. MATTS has been designed primarily to allow modelling of systems-of-systems scenarios and testing of secure component composition analysis techniques. The program integrates together a number of tools:

- A secure, sandboxed, virtual machine.
- A mobile code framework allowing communication between agents.
- A composition analysis engine.
- A formal analysis tool.

MATTS allows scenarios to be created interactively and comprised of a mixture of both real and virtual nodes or services. Separate software (called the CompositionClient) has been developed for use on mobile devices that, when deployed, allows them to appear in the modelled visualisation. The CompositionClient utilises .Net Mobile, allowing it to run across a variety of device types including phones, PDAs, laptops and PCs.
As the scenario evolves and changes, the properties and connections between services are dynamically analysed to determine the security properties of the overall composition of services. The analysis process is based on a scripting technique.

2. Making MATTS part of the Aniketos platform

MATTS platform provides support to verify the service compliance to security contracts at both design time and at runtime. Currently the MATTS tool itself is somewhat monolithic. It would benefit from being deconstructed into its constituent parts (scenario management, visualisation, analysis). Various elements of the tools could then be applied in a relatively technology-neutral way within the Aniketos platform.

The analysis process could be used as part of the security testing methods used by Aniketos. In order to do this, certain improvements are needed in terms of generalising the analysis processes and providing a richer scripting language.

Aniketos would benefit by having a scriptable analysis system that could be used to manage other parts of the security analysis process, and the tools described here could provide this baseline functionality.

3. Development roadmap

We expect to add functionality to these tools in a number of areas as follows:

Decomposition into Services

Currently MATTS provides a variety of different functionalities, including modelling capabilities, visualisation and analysis. For the purposes of Aniketos, the system should be decomposed into a number of services, providing each of these capabilities individually.

Moreover, in order to properly analyse the properties of a service composition, the analysis engine should be restructured to allow security properties to be determined in a distributed manner. For example, each domain could have a single instance of an analysis service to measure security for a defined set of other services. However, these security domains are unlikely to be wholly independent of one another, so that the security in one domain may be dependent on security in neighbouring domains, or services further afield. Given this, multiple analysis services would likely need to interact with one other across domains in order to better determine the security of a given system.

To accommodate this, we intend to devise methods for the analysis service to act in a distributed peer-to-peer manner, using the results from other areas as an input to the analysis process.

The development work would likely fit best into WP 3, for example in tasks 3.2 and 3.6.

Risk Analysis Extensions

Risk assessment within networked services is an important means of establishing whether changes should be made in order to improve security. We intend to extend the tools described here to consider risk within the network, taking individual measurements for risk at nodes (e.g. the likelihood of attack) and aggregating this information across the network in order to calculate overall risk. Overall risk is based not just on the individually assigned values, but also on the network structure, for example by applying higher risk to hubs within the network.

As well as measuring risk, we plan to include capabilities able to provide recommendations for changes to the network intended to minimise risk, for example by reducing the prevalence of hubs and preventing the creation of high-risk links that would segregate the network if attacked.

We anticipate that these techniques might be used as part of WP 2 (tasks 2.1 and 2.3), WP3 (tasks 3.2, 3.3, 3.5 and 3.6) and WP4 (task 4.2, 4.3 and 4.6).

Aspect-Oriented Correction

At present MATTS can be used for the detection of security vulnerabilities in composed service-oriented systems. Once detected however, questions arise as to how the system should react in order to try to address the problems discovered.
The correction problem deals with methods to allow discovered vulnerabilities to be automatically fixed through changes made to the system. We can find a number of theoretical examples of how correction could be applied (e.g. correction in relation to data flow, boundary check, CVP, etc.). Generalising these, we can see that adaptation in order to correct a detected vulnerability in a composition can be achieved in one of five ways:

- Topology rearrangement.
- The introduction of new or removal of existing components from the system.
- Using component wrappers.
- Reconfiguration of existing components.
- Adaptation of existing components.

We intend to look at the fifth of these – adaptation techniques – as an extension to the MATTS tools in the future. We will look at technologies relating to Aspect-Oriented Programming (AOP), and in particular structural reflection. This allows code to be inspected and altered at run-time in order to affect the functionality of an executable. In particular, additional code can be woven into existing executables (or removed from them) at development or run-time in order to augment the security of a service. In the context of service composition, this introduces interesting challenges, since weaving must occur across services, rather than just applied to an individual executable. Our intention would therefore be to incorporate aspect weaving and structural reflection with the tools through the scripting mechanism that MATTS already uses.

We expect that this work might fall most clearly into WP 3 (task 3.3) and WP 4 (4.4 and 4.6).

4.7.3 Property and Policy Tools (LJMU)

The PropertyInterface tool (see section 4.1.13) helps in determination of security properties and save it in a simple XML format. The policy interface tool on other hand allows arithmetical and logical expression to be defined over the set of properties. Expression of this form could be used to represent a policy statement (or entire policy) for individual services to be tested against. This XML file can be transferred from security property determination module to the security verification module for verification. Security verification module could use the policy interface tool to generate expressions or policies in order to verify the service compliance to security contracts both at design and at runtime.
4.7.4 MASTER Real-time Compliance Manager for SOA (ATOS)

<table>
<thead>
<tr>
<th>Name</th>
<th>Policy Deployment Component (PDC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>A component for deployment of policies into a distributed run-time infrastructure in a synchronized way.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS - Composite service adaptation and recomposition</td>
</tr>
<tr>
<td>Developed by</td>
<td>Atos Origin (ATOS)</td>
</tr>
<tr>
<td>Maturity</td>
<td>No commercial installation of this asset exists. The maturity level is comparable to TRL 6 of the NASA TRL scale (“prototype demonstration in a relevant environment”).</td>
</tr>
<tr>
<td>References</td>
<td>MASTER project [10]</td>
</tr>
<tr>
<td></td>
<td>MASTER Technical architecture and APIs for multiple trust domains [132]</td>
</tr>
<tr>
<td>System requirements</td>
<td>This asset is developed in Java, and web services, with a web based GUI for administration management. JAVA: jre1.6.0_03 APACHE TOMCAT: v.5.5 or v.6.0 (tested in both, specially in version on testbed: v6.0.18) AXIS 2: v.1.5.1 POSTGRESQL v.8.3</td>
</tr>
</tbody>
</table>

1. Short description

The MASTER Compliance Manager (described in a public deliverable available in project MASTER website [132]) consists in different artefacts involved in the process of monitoring, assessing and enforcing compliance. The Policy Deployment component allows the deployment of compliance policies for signalling in the target SOA environment, monitoring of control activities, enforcement of control activities, and assessment of the levels of conformance to policies and regulations under study, based on signalling and monitoring. The Policy Deployment component, together with the four specific services that rely on its policies, proposes a methodology for adoption of the Compliance management paradigm of MASTER.

2. Making MASTER Real-time Compliance Manager for SOA part of the Aniketos platform

The MASTER Compliance Manager would be an asset for the tasks related to Policy based control of the secure service composition (Tasks3.4) and for the WP4 threat detection mechanisms. The runtime interfaces for policy deployment for the four services (Reaction and control, Monitoring, Signalling and Diagnostics) could be implemented in Aniketos specific services, thus making their behaviour policy based (through the Policy Deployment component). This would result in a unified management of these Aniketos services, and would extend the measurement of the “degree of compliance” proposed by the MASTER methodology.

Currently, The MASTER Real-time Compliance Manager covers chains of composed services. Extending it to a Service Composition Compliance Manager in the more elaborated secure composition environment that Aniketos proposes would be an interesting direction to take.

3. Development roadmap

A necessary step to make MASTER part of Aniketos would be to tie the Policy Deployment to the modelling tools that are being proposed. The Policy Deployment component could thus be aware of all
the relevant configurations and security requirement that these tools can express, collect them and readjust the monitoring/signalling services accordingly.

4.7.5 Avantssar platform for validating trust and security properties (SAP)

Name: Avantssar platform for validating trust and security properties
Type: Tool and Technique/Method for specifying and analysing security properties of communication protocols and Web service compositions.
Purpose: Modelling language and tool support for the static analysis of security protocols and Web services compositions.
Support area: DS - Security property definition and evaluation, DS - Composite service analysis and preparation
Developed by: AVANTSSAR project consortium [external] (including SAP [Aniketos])
Maturity: Mature, result of EU FP7 project.
References: AVANTSSAR project [134], Armando et al., 2008 [135], Armando and Compagna, 2008 [136]
System requirements: Unix/Linux, Windows, OS X
Miscellaneous: The core tools (model-checking tools for security properties) of the Avantssar platform are released under a BSD-style license. SAP developed a proprietary prototype using the AVANTSSAR technology for analysing security properties of business process models.

1. Short description
The Avantssar platform provides tool support for the specification and automated validation of security protocols in particular and trust and security in service-oriented systems in general. In more detail, the Avantssar platform allows for modelling formally (using the ASLan specification language) and reasoning automatically about services, their composition, their required security properties and associated policies. Thus, the Avantssar platform provides tool support for ensuring the security of service compositions at design-time.

Based on the Avantssar platform, SAP Research a proprietary prototype for specifying and analysing security properties of service orchestrations.

2. Making ASLan and the AVANTSSAR Validation Platform part of the Aniketos platform
In general, the ASLan language and the AVANTSSAR Validation Platform might serve as a basis for the static analysis of Web-service compositions within the Design-time support of the Aniketos Platform (i.e., WP1, WP2, and WP3). Moreover, ASLan might serve a specification language for the use scenarios (developed in WP1) and the Aniketos use studies (WP6).

3. Development roadmap
The Avantssar platform is a mature technology that could either in parts (e.g., only the backend tools) or as a whole be integrated into the design and modelling parts of the Aniketos framework (developed in WP2 and WP3). Both the AVANTSSAR Platform and the Rodin Platform (see next Section 4.7.6) provide frameworks for the formal verification of service properties. As such, both platforms will inspire the architecture of the Security Verification Module and, moreover, will lay out the basis for the verification techniques used in this module.
4.7.6 Rodin Platform (SAP)

<table>
<thead>
<tr>
<th>Name</th>
<th>Deploy: Rodin platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool/Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>Formal analysis of safety properties of service orchestration and BPMN models.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Composite service analysis and preparation</td>
</tr>
<tr>
<td>Developed by</td>
<td>Deploy project consortium [external] (including SAP[Aniketos])</td>
</tr>
<tr>
<td>Maturity</td>
<td>Commercial, In-house, Prototype, Work in progress, etc</td>
</tr>
</tbody>
</table>
| References | DEPLOY project [137]  
Bryans et al., 2009 [138]  
Wieczorek et al., 2009 [139] |
| System requirements | Eclipse |
| Miscellaneous | Rodin platform available as Open Source. SAP developed proprietary extension of the Rodin Platform providing a modelling environment supporting the formal analysis of BPMN models. |

1. Short description
In general, the Rodin platform provides an integrated specification and formal analysis environment based on the B method. In particular, it provides both a refinement-based approach for analysing safety properties (e.g. freedom of deadlocks) of distributed systems.

Based on the Rodin platform, SAP developed a tool for modelling and analyzing service orchestration. On the one hand, the orchestration models can be used (both interactively using a theorem proof and automatically using a model-checker) for checking safety properties. On the other hand, the orchestration models can be used in model-based testing scenarios.

2. Making Deploy part of the Aniketos platform
The orchestration modelling and analysis part of the Rodin platform may serve as a conceptual starting point for the design-time modelling tools developed in WP3.

3. Development roadmap
The Rodin platform is a mature technology that could either in parts (e.g., only the backend tools) or as a whole be integrated into the design and modelling parts of the Aniketos framework. In particular, the Rodin tools may be used for analyzing safety properties of service compositions (WP3). Note that this might also be achieved by re-using tools from the Avantssar platform.

4.7.7 STACS: Scalability Testbed for Access Control performance of Services (TSSG)
If the performance components of a service contract appear not to be fulfilled, STACS (see section 4.6.5) can rerun the access control performance experiments and compare the results with equivalent performance predictions obtained at contract negotiation time.

4.7.8 Other related work to security verification module
There exist various extensions of general purpose theorem proofs such as Isabelle [147], KIV [151] or Coq [152] for supporting high-level modelling languages like Z [154], ASM [151], or UML/OCL [155]. Compared with the Rodin Platform, especially approaches based on a conservative logical embedding [153], provide a strong logical foundation that ensures the consistency and correctness of the reasoning environment. While technically possible, these approaches often lack the deep integration into a widely used modelling environment such as Eclipse that increases the willingness of non formal methods exports to use such tools [147].
While most of the previously mentioned approaches focus on safety properties, formal methods are also widely used for ensuring security properties. For example, several automated tools for security protocol verification have been developed, e.g. [140], [141], [142], [143] and [144], among them the back ends of the Avantssar Platform (i.e. [145], [136], [146]). They allow engineers to find problems in their security protocols before deployment. Indeed, several attacks to security protocols have been detected using automated tools. While, in principle, all of these tools serve a similar purpose, the Avantssar Platform provides already an input language, called ASLan [148] that covers some aspects of composable services. Alternatively, there are also semi-automated or interactive approaches for verifying security protocols. Most prominently the inductive approach inspired by Paulson [149] using Isabelle [147].

We have also identified the following works as relevant for security verification module.

### 4.7.8.1 Security-Oriented Service Composition and Evolution

This approach (see section 4.5.5.1) provides some ideas about the analysis techniques for checking the security compatibility between services. These ideas may be useful for formal verification of services.

### 4.7.8.2 Static Verification Framework Tool

<table>
<thead>
<tr>
<th>Name</th>
<th>Static Verification Framework Tool (SVF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Support the design and analysis of abstract behavioural system specification with respect to key security attributes</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td>Developed by</td>
<td>CITY University [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype</td>
</tr>
<tr>
<td>References</td>
<td>PEPERS project [55]</td>
</tr>
<tr>
<td></td>
<td>PEPERS platform overview [156]</td>
</tr>
<tr>
<td></td>
<td>PEPERS SVF component [157]</td>
</tr>
<tr>
<td></td>
<td>Siveroni et al., 2010 [56]</td>
</tr>
<tr>
<td></td>
<td>Siveroni et al., 2008 [57]</td>
</tr>
<tr>
<td></td>
<td>Zisman, 2007 [58]</td>
</tr>
<tr>
<td>System requirements</td>
<td>Eclipse plug-in</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Open source, released under the LGPL license</td>
</tr>
</tbody>
</table>

1. **Short description**

The main scope of the Static Verification Framework Tool (SVF) is to support the design and analysis of abstract behavioural system specification with respect to key security attributes.

The overall aim is broken down into the following objectives:

- To provide support for the construction of design models representing abstract behavioural system specification
- To provide support for the construction of abuse cases
- To provide support for the specification of security and general application properties to be verified against the abstract behavioural system specification
- To statically verify system specifications against security and general application properties
- To visualize the results of the static verification process
Following from normal software development process, the view is that the Static Verification Framework interacts with the Design and Architecture Framework of the Pepers Framework [156], since the former will use the requirements specifications, architecture models, and high-level design models generated by the Design and Architecture Framework, in order to support the construction of the abstract behavioural specifications and properties to be verified against these specifications.

2. Making SVF part of the Aniketos platform

The functionality of this tool relates to the Security Verification module of the Aniketos platform. The tool is open source and is provided as an Eclipse plug-in, thus it can be extended to comply with the requirements identified in Aniketos for supporting the design-time security of the implemented services and workflows.

3. Development roadmap

SVF can be extended to meet the functional requirements of the Aniketos platform. The tool can interface with the Security Policy Determination module and provide design-time decisions over the selected security policies.
4.8 Security policy monitoring module

4.8.1 Risk-aware Usage Decision Making in Highly Dynamic Systems (CNR)

Name: Risk-aware Usage Decision Making in Highly Dynamic Systems
Type: Technique
Purpose: The proposed technique helps to make a rational decision when the required up-to-date values are unavailable and prediction is required.
Support area: Runtime validation of secure service behaviour
Developed by: CNR [Aniketos]
Maturity: Work in progress
References: L. Krautsevich et al., 2010 [125]
L. Krautsevich et al. 2, 2010 [126]
System requirements: N/A
Miscellaneous: N/A

1. Short description
The main goal of any access control is to guarantee that a resource is used as it is stated in established policies and to prevent unauthorized users from accessing or corrupting the resource. An authorization context received by a policy decision point can be imprecise because of two types of unintentional or intentional causes.

We propose a basic risk-based approach which helps in making decision for usage control model when a number of uncertainties are present. We provide a probabilistic approach based on Markov chains to model mutability of authorization context. The approach can be used to solve different problems caused by presence of uncertainties. We employ risk analysis in order to make the most rational decision and be as flexible as possible.

2. Making Risk-aware Usage Decision Making in Highly Dynamic Systems part of the Aniketos platform
First of all the monitored attributes should be considered carefully to find those to which our approach is applicable (Markovian property must be satisfied).

The technique should be modified to provide valuable information for Aniketos. This could be either the probability of violation of a policy or decision about the further access. In other words, the Aniketos framework may use the result as an advisory or as a decision making process.

The proposed approach should be made more effective by using the values (probabilities) found during the previous access check rather than re-computing the values from the beginning. Such on-line approach should significantly reduce computational cost and make the analysis more suitable for run-time analysis.

Finally, the prototype should be created and possible overheads should be estimated.

3. Development roadmap
The proposed technique should help in monitoring different policies when up-to-date information is available (Tasks 3.4 and 3.5).

The first step is to determine what kind of monitored attributes can be evaluated with the proposed technique. Those attributes which do not satisfy Markovian property should be reconsidered and, if possible, the technique should be adapted to be applied to a wider range of attributes.
The next step is to decide how the results of the analysis can be used by Aniketos and how required information (transition probabilities and costs) can be acquired and updated.

The main activity, which is devoted to improving applicability of the approach, should follow. Currently, the main issue is to reduce amount of computations required for finding probability of policy failure, but other issue may arise.

The final step is to implement the approach, test it and integrate with Aniketos.

4.8.2 MATTS (Mobile Agent Topology Test System) (LJMU)

The MATTS (see Section 4.7.2) can provide a platform for analyzing service compositions and testing of secure component composition analysis techniques. It has the capability of analyzing a service composition at run-time based on patterns of potentially policy-violating interactions and could be useful for identifying security problems in a service composition. It could be used for monitoring of security policies and properties, as well as monitoring of security contract fulfilment.

4.8.3 Platform for Run-time reconfigurability of security (ATOS)

PRRS (see section 4.6.4) provides run-time management of S&D Solutions and monitoring of the system context. PRRS supplies the necessary and proper security solution to applications during runtime, upon request. The platform includes a Monitoring Service that analyses the events and informs when a rule violation occurs. Security experts are in charge of creating those monitoring rules.

4.8.4 Model Driven Security/OpenPMF

<table>
<thead>
<tr>
<th>Name</th>
<th>Model Driven Security/OpenPMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool and Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>High level security policies applied automatically in SOAs</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>RS - Trustworthiness monitoring and evaluation</td>
</tr>
<tr>
<td></td>
<td>CS - Threat analysis and notification</td>
</tr>
<tr>
<td>Developed by</td>
<td>ObjectSecurity [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Commercial</td>
</tr>
<tr>
<td>References</td>
<td>Object Security project, [156]</td>
</tr>
<tr>
<td></td>
<td>Ritter et al., 2006 [159]</td>
</tr>
<tr>
<td>System requirements</td>
<td>Eclipse, Intalio BPMS, CORBA, various others</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Integrates with Intalio (BMPS Integrated Process Development Environment) which is partially open source, however the OpenPMF™ plugin appears to be proprietary.</td>
</tr>
</tbody>
</table>

1. Short description

Model Driven Security (MDS) allows security policy to be defined at a high level of abstraction before being converted into lower level technology-specific policies. This is useful in distributed, agile environments such as service oriented architectures, since the high level policy can be enforced across the entire SOA environment.

OpenPMF™ is a commercial product that allows the configuration, enforcement and monitoring of the business-driven security policies through the exploitation of the model-driven security. In addition to allowing the high- to low-level policy translation, OpenPMF™ also provides a means to directly apply the policy by adding security components into an SOA deployment. The system makes use of the
CORBA Component Model (CCM) to allow introspection and navigation of component features in order to achieve this.

2. Making MDS/OpenPMF™ part of the Aniketos platform

MDS and OpenPMF™ could provide some of the functionality intended to be part of Aniketos. For example, it could provide some of the policy/property translation techniques intended for use as part of Task 3.4 (“Policy based control of secure service composition”).

It also appears to align closely with some of the other elements of Aniketos. For example, the security modelling elements may be relevant for Task 1.2 (“Socio-technical security modelling language and tool”). The notification and response elements may be appropriate for Task 4.2 (“New methods for threat notification and response capability”) and Task 4.3 (“Monitoring mechanisms for changes”).

Finally, the application of the tool as part of the AD4 project, where it was used to apply security to a service-oriented air traffic simulation tool, could provide potentially useful background for Case Study B (“The Emerging European Air Traffic Management systems”) of WP6.

3. Development roadmap

The MDS approach is described in the literature; however the main elements of OpenPMF™ appear to be available only commercially. OpenPMF™ does apparently integrate with the Intalio BPMS development environment, which itself is partially open source.

Given this, maybe the work would be more appropriate for providing inspiration or techniques that can be applied in the work packages, rather than as a system that can be incorporated into it.

The work appears to be potentially relevant for WP1, WP3 and WP4.

4.8.5 Other related work to Security policy monitoring module

We are currently not aware of any suitable relevant work apart from SVF (see below) that could be used to realize this module, therefore technological surveillance should continue within WP4.

4.8.5.1 Static Verification Framework Tool

SVF tool (see section 4.7.8.2.) is allocated in the security policy monitoring module to interface with the relevant monitoring components that will assess the design-time verification of security policies applied in the provided services and workflows.

4.9 Threat response recommendation module

4.9.1 MUSIC studio and middleware (SINTEF)

MUSIC (see section 5.3.1.1) allows service developer to include threat responses mechanisms in the service modelling. For example, the threat level can be defined as context information, and adaptive behaviour can be modelled by specifying different response actions (e.g. reconfiguration of components) with regard to the changes of threat level. The MUSIC middleware at runtime will trigger adaptation process and perform reconfigurations recommended by the MUSIC Adaptation Reasoners when changes in the threat level are detected by context sensors.

4.9.2 SeaMonster – Security Modelling Software (SINTEF)

SeaMonster (see section 4.1.1) is not a runtime module, but has the technology of looking up information and adding new entries into the SVRS (which is a candidate for the Threat repository module). This can be exploited when developing the threat response recommendation module. Threat models created with SeaMonster might also be used as recommendations for human users, as they tend to include mitigations and security activities (linked to Security Activity Graphs and other artefacts).
4.9.3 Risk-aware Usage Decision Making in Highly Dynamic Systems (CNR)

The Risk-aware Usage Decision Making technique (see section 0) provides support for making decisions for resource usage control when a number of uncertainties are present and in this sense could be relevant for the threat response recommendation module.

4.9.4 Goal Risk Model (UNITN)

Goal Risk Model (see section 4.1.5) weaves between events which lead to risks in reaching goal and allow for defining treatment to reduce/handle risks. The reasoning part of the GRM framework supports Threats Response Recommendation Module as it analyzes the Goal-Risk model and discovers what the risks are and how to avoid risky alternatives at the organizational level. This guides what alternatives the implemented system has to adopt at the technical level and that is of minimum risks when certain events happens. GRM helps in the design of a service composition based on risks and treatments, therefore assuring a robust model that satisfies user preferences. At runtime, treatments can be services.

4.9.5 Architecture for Self-Adaptive Socio-Technical Systems (UNITN)

<table>
<thead>
<tr>
<th>Name</th>
<th>Architecture for Self-Adaptive Socio-Technical Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>This architecture enables the development of adaptive socio-technical systems. In these systems, multiple actors (both social and technical) interact in order to achieve their own goals. The mission of the architecture is to monitor the supported actors, diagnose failures and under-performance, and enact compensation actions to cope with identified threats.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS - Composite service adaptation and re-composition</td>
</tr>
<tr>
<td>Developed by</td>
<td>DISI – University of Trento (UNITN) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Work published in international conferences</td>
</tr>
<tr>
<td></td>
<td>The architecture is implemented and can be applied to different domains</td>
</tr>
<tr>
<td>References</td>
<td>Dalpiaz et al., 2009 [160]</td>
</tr>
<tr>
<td></td>
<td>Dalpiaz et al., 2010 [161]</td>
</tr>
<tr>
<td>System requirements</td>
<td>JVM 1.6+</td>
</tr>
<tr>
<td></td>
<td>Eclipse 3.3 with EMF (for design support)</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1. Short description

A Socio-Technical System (STS) consists of an interplay of humans, organizations and technical systems. STSs are heterogeneous, dynamic, unpredictable and weakly controllable. Their operational environment changes unexpectedly, actors join and leave the system at will, actors fail to meet their objectives and underperform, and dependencies on other actors are violated.

To deal with such situations, we propose an architecture for STSs that makes an STS self-reconfigurable, i.e. capable of switching autonomously from one configuration to a better one. This architecture performs a monitor-diagnose-reconcile-compensate cycle: it monitors actor behaviours and context changes, diagnoses failures and under-performance by checking whether monitored behaviour is compliant with actors’ goals, finds a possible way to address the problem, and enacts compensation actions to reconcile actual and desired behaviour.
This architecture is requirements-driven: correct behaviour is defined in terms of requirements models and evaluated according to the Belief-Desire-Intention paradigm. An extended version of the architecture specifies actor interaction via the notion of social commitment. Actors interact—via commitments—to achieve their own goals. Consequently, commitments are monitored and, if threatened or violated, adaptation is required to avoid goal failure.


The Architecture for Self-Adaptive Socio-Technical systems treats adaptation at its origin i.e. at the socio-technical system design. The architecture enables the development of self-adaptive socio-technical systems that exploit high-level abstractions (requirements) to adapt. Thus, the architecture supports Aniketos Threat response recommendation module by providing mechanisms to monitor changes in the system, e.g. context changes, and recommending/enacting a proper plan as response. The architecture is supported by a tool to monitor such changes, diagnose the requirements model and act accordingly.

Two characteristics of Aniketos make this architecture suitable:

Service-orientation: Aniketos explores service-oriented settings, where service providers and consumers (not necessarily mutually exclusive) interact via services to achieve their own objectives. The architecture supports multiple interacting actors that interact via commitments. Notice that commitments exist whenever we talk of a service: the service provider commits to the service consumer for the provision of a service.

Socio-technical perspective: Aniketos deals not only with technical aspects, but also takes into account the social aspects. This architecture is in line with this vision since it conceives the system at the business level. Indeed, each subsystem is characterized in terms of its business goals, and consequently adaptations are enacted to make sure that the social entity (e.g. the organization) behind technical systems achieves their goals.

The architecture can be used in Aniketos in different ways: (i) as a baseline implementation of a middleware for adaptive software; or (ii) to guide the development of a novel framework that relies on the principles and algorithms defined for this architecture.

Of particular importance will be the algorithms for diagnosis and reconfiguration that this architecture supports. The diagnosis algorithms enable to identify a wide range of failures (goal failures, plan failures, domain assumption violations, and dependency and commitments violations). The reconfiguration algorithm supports variant generation and selection (on the basis of different criteria, such as non-functional requirements and cost).

3. Development roadmap

The architecture for self-adaptive socio-technical systems can contribute to different areas in the Aniketos platform:

WP1, specifically D1.2, the Aniketos architecture: The Architecture for Self-adaptive systems can be taken as a reference and provide principles and guidelines to structure the Aniketos architecture, specifically for what concerns adaptation.

WP4: the threat identification and adaptation algorithms of Architecture for Self-adaptive systems can serve as a baseline for Aniketos. The focus of Aniketos will require us to customize these algorithms to the specific characteristics (trust, security, service recomposition, composed services).

4.9.6 Other related work to Threat response recommendation

We are not aware of any relevant technology that could be specifically used to realize this module, but technological surveillance should continue within WP4.
4.10 Service threat monitoring module

4.10.1 Platform for Run-time reconfigurability of security (ATOS)

PRRS (see section 4.6.4) is relevant to be considered within this module due to it performs the run-time monitoring of events described inside the pattern’s monitoring rules. The mechanism includes a Monitoring Service that analyses the events and informs when a rule violation occurs. Security experts are in charge of creating those monitoring rules.

4.10.2 MASTER Design Toolkit (UNITN)

<table>
<thead>
<tr>
<th>Name</th>
<th>MASTER Design Toolkit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>A set of tools that are useful for analysts in deploying the MASTER system in a specific organization, comprising three main parts: workbench, repository, and verification &amp; validation (V&amp;V) tool.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Composite service analysis and preparation</td>
</tr>
<tr>
<td></td>
<td>RS - Runtime validation of secure service behaviour</td>
</tr>
<tr>
<td>Developed by</td>
<td>DISI – University of Trento (UNITN) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Specified in deliverable 3.1.1 of the EU project MASTER. Toolkit under development, alpha version</td>
</tr>
<tr>
<td>References</td>
<td>MASTER project [10]</td>
</tr>
<tr>
<td></td>
<td>MASTER Design Workbench [12]</td>
</tr>
<tr>
<td>System requirements</td>
<td>Assumes that the target IT systems are developed using SOA</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1. Short description

MASTER Design Toolkit is a set of tools that comprises three main parts: workbench, repository, and verification & validation (V&V) tool. The (graphical) workbench facilitates the modelling process. It provides a means for the analysts to model, analyze, and design the MASTER system. Its graphical design environment allows users to build a design model for the business process that is being analyzed. The repository provides support to the MASTER analysts to store, checkout, and manage versioning of the development artefacts (e.g., models, policies). It is also useful for a knowledgebase system that helps the analysts in analyzing and designing the MASTER system. Finally, the set of V&V tools is used to verify and validate particular properties of the model before implementing the design in terms of executable control processes.

2. Making “MASTER Design Toolkit” part of the Aniketos platform

MASTER Design Toolkit enables organizations to assess the level of control of business processes at any given moment. It provides a set of methodologies and tools that facilitate the implementation of quantifiable indicators aimed at assessing the level of control of IT systems underlying these business processes. The level of control encompasses compliance with company objectives, regulatory requirements and service level agreements. MASTER is notably useful within the context of outsourced services.

3. Development roadmap

The Master Design Toolkit could be used as a baseline for WP1, Task 1.2. The idea behind this tool could be adapted to satisfy the need of T1.2 for expressing and analysing security requirements for the
services. The graphical workbench provides a suitable graphical environment to support modelling and analysis.

The tool is also worthy to be considered as baseline for WP2, since it ensures that the final model satisfies the control objectives (security requirements could be considered as such).

4.10.3 Other related work to threat response recommendation

We are not aware of any relevant technology apart from the background described above that could be specifically used to realize this module, and therefore technological surveillance should continue within WP4.

4.11 Notification module

4.11.1 MATTS (Mobile Agent Topology Test System) (LJMU)

MATTS (see Section 4.7.2) has a background analysis module which monitors the changes of composition and other security-related parameters. Its graphical user interface can give clear notification (by audio and visual means) to the users about the composition change or any violations against security policies that have been analysed.

4.11.2 Platform for Run-time reconfigurability of security (ATOS)

The PRRS (see section 4.6.4) includes a Console that is relevant to be considered as baseline for the notification module. The console is the main GUI which allows users to be informed on the configuration and status of the framework and its logs.

4.11.3 Other related work to Notification module

We are not aware of any relevant technology apart from the background described above that could be specifically used to realize this module, and therefore technological surveillance should continue within WP4.
4.12 Community support module

4.12.1 Related work to Community support module

With respect to external works relevant for this module, we can only mention OAuth (see below). Further identification work shall be conducted within WP5.

4.12.1.1 OAuth

<table>
<thead>
<tr>
<th>Name</th>
<th>OAuth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Technology</td>
</tr>
<tr>
<td>Purpose</td>
<td>Decentralised mechanism for delegating API access; also enables secure authentication between RESTful web services</td>
</tr>
<tr>
<td>Support area</td>
<td>CS – End user trust and assurance</td>
</tr>
<tr>
<td>Developed by</td>
<td>OAuth community [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>v1.0 used since 2006; adopted by Google [164], Facebook [165], Twitter[166], Yahoo [167], and others</td>
</tr>
<tr>
<td>References</td>
<td>OAuth protocol [168]</td>
</tr>
<tr>
<td></td>
<td>OAuth v1.0 specification [169]</td>
</tr>
<tr>
<td>System requirements</td>
<td>Implementations are available in C#, VB.NET, ColdFusion, LISP, Java, JavaScript, Perl, PHP, Python, Ruby.</td>
</tr>
</tbody>
</table>

1. Short description

OAuth is an IETF authorization standard (OAuth v1.0 Protocol is the RFC 5849 [169]) that allows a service (Consumer) to access protected resources accessible through another service (Provider) when delegated by a user. Unlike OpenID [163] or similar technologies, the identity of the user is never revealed to any of the services; the delegation is accomplished by the use of a secure token.

OAuth was originally created to allow authentication delegation for Twitter applications [166]; as of September 2010, all 3rd party Twitter apps are required to support it. Its design allows for delegating access to all kinds of services: SOAP and REST-based web services as well as web applications.

2. Making OAuth part of the Aniketos platform

Each Component could be issued a unique identifier by Aniketos when they register in the Marketplace. If using RSA-SHA1 public key cryptography, Aniketos could store the RSA public key certificates of each Component instead of requiring them to share the certificates with each other manually. If using Hash-based Message Authentication Code [171] with SHA1, i.e. HMAC-SHA1, each Component pair (Components that could conceivably replace each other during a recomposition event) needs to create shared secrets.

3. Development roadmap

OAuth v2.0 is currently under development. However, signatures and cryptography have been dropped in v2.0 in favour of bearer tokens (cookies); this makes the protocol less useful for secure discovery. Further analysis on the possibilities to profit in Aniketos from OAuth should be done within WP5 Platform Construction.
4.13 Threat repository module

4.13.1 Security Vulnerability Repository Service (SEARCH)

Name: Security Vulnerability Repository Service
Type: Storage service
Purpose: Manages and stores security-related resources (vulnerabilities, threats, security activities, security models and artefacts), and interconnections between these resources.
Support area:
- DS – Trustworthiness definition and evaluation
- DS – Security property definition and evaluation
- RS – Runtime validation of secure service behaviour
- CS – Threat analysis and notification
Developed by: SEARCH-LAB (SEARCH) [Aniketos] and other SHIELDS project partners: TXT e-solutions s.p.a [external], Montimage EURL [external]
Maturity: V1.0
References:
- SHIELDS project [172]
- SHIELDS SVRS [173]
System requirements:
- GlassFish application server [174], PostgreSQL database [175], Java EE 5 platform [176], JSP 2.1, Jersey 1.0
Miscellaneous:
- The SVRS was created by three different companies. Montimage owns the rights for the subscription, notification and statistics modules of the application logic source code; SEARCH-LAB owns the rights for the database, machine interface, and the rest of the application logic source code. Content already in the SVRS – contributed by the SHIELDS consortium – is freely available.
- TXT created the user interface and released it under the GNU General Public Licence 3.0.
- The SVRS is operated by SEARCH-LAB.

1. Short description

The Security Vulnerability Repository Service is a repository of security-related resources (i.e. vulnerabilities, threats, security activities, security models) and the links between them.

Content consumers (software developers) download resources from the SVRS either by special development or testing tools, or via a web interface where they are presented in a human-readable and intelligible format. To access the content stored within the SVRS, consumers can use multiple search interfaces (free text-based or ontology-based).

Besides the aforementioned repository functions, the SVRS also provides community services. Community services include communication functionality related to security in general (e.g. articles, comments, and forum) and the SVRS (feature requests, bug reporting).

2. Making the SVRS part of the Aniketos platform

As said, the SVRS is a semantic repository of security-related information, including security models of possible threats to composite services; this makes it an important component of the threat repository module.
The SVRS contains a constantly expanding semantic repository of threats, vulnerabilities, security activities and other security-related information. Its greatest usefulness in the Aniketos platform is serving as a reliable knowledge base for the various tools in the Aniketos tool chain both at design time and runtime. Thus new security formalisms and tool-specific content introduced by Aniketos would need to be supported by the SVRS as well.

New requirements or requests for improvement (either posed by the tools or the Aniketos platform itself) would also need to be implemented in the SVRS. An example of such a requirement would be an improvement to the way the SVRS handles connections in order for tools to more easily locate indirect connections between security models and core elements.

Similarly, the SVRS allows tools to provide feedback – both regarding the occurrences of threats and vulnerabilities, and the relative usefulness of security activities in dealing with those threats and vulnerabilities. This functionality could be extended, and feedback can be used to generate various statistics both for human and tool use: both to find the most widespread vulnerability types, and the most effective ways to combat those vulnerabilities.

Since the current design of the SVRS is somewhat monolithic (as opposed to the distributed vision of the Aniketos platform), it could be re-engineered to be more distributed using either of the following approaches:

- Multiple SVRS instances all over the world with local databases; these instances would be fully independent with the option to link them together to perform periodic ad-hoc synchronisation.
- Multiple SVRS instances all over the world with local databases, and a central synchronisation service (envisioned as a single server or network of servers); all instances would use the synchronisation service to keep their own localised databases up-to-date as well as submit any new or changed data so that other instances can make use of them.

Another design-level change to consider is moving from the rigid “Quality Responsible” process to a more community-driven rating system to endorse content and prioritise search results. Such a system would be more flexible and reliable in a distributed environment.

3. Development roadmap

The SVRS can potentially serve as an information repository for all tools and services that make up the Aniketos platform; however, since its main purpose is to serve as a repository for threats and other security information, the development work fits best into WP 4. We envision the role of the SVRS in WP4-specific tasks as such:

T4.3 – The SVRS can serve as the information source for the Aniketos platform components to help discover changes and threats. It could store and share threat information with other services as well via a proactive notification system added to the SVRS (similar to the already-existing notification system for human users).

T4.4 – The SVRS can assist runtime analysis: it can serve as an input source, and also record statistical data based on the results of an analysis.

T4.5 – Retrieval of up-to-date threats and related information (to be used during design-time and runtime analysis) is the SVRS’s main purpose. This functionality could be extended via a proactive notification system as well.

4.13.2 MASTER Design Toolkit (UNITN)

The repository that is part of the set of tools of MASTER Design Toolkit (See section 4.1.4) provides support to the MASTER analysts to store, checkout, and manage versioning of the development artefacts (e.g., models, policies). Moreover, the repository is also useful for a knowledgebase system that helps the analysts in analyzing and designing the system. Therefore, the repository could be used in Aniketos as well, containing relevant threats and incorporating recommendations for the response.
4.13.3 Other related work to Threat repository module

Two main approaches may be considered related to the threat repository. On the one hand security focused Information Repositories, on the other hand vulnerability databases. Information Repositories serve as information sources for secure development. These repositories are completely public, and all content is accessible via a web browser. Vulnerability databases keep track of vulnerability instances in software. These databases are aimed at system administrators and users (helping them find information about vulnerabilities in software they have installed, and any possible ways to mitigate them).

**Information repositories:**

Build Sec In [177] provides a comprehensive library of articles related to secure development principles and best practices. Each phase of development from design to testing is covered by the body of articles. Submitted papers undergo peer review and an approval process before addition to the database. Build Sec In therefore depends on the submissions of security experts to expand and maintain its knowledge base. Furthermore the articles themselves are merely meant to be informative – no guarantees on developed code quality possible – since there are no formal models available that support automated verification tools.

Secure Coding Initiative [178] is the most widely used C, C++ and Java secure coding standards on the web. Two source code analyzer tools (Fortify and Compass/ROSE) have partially adopted the standards for their own use. It collects secure programming “best practices” for three programming languages only, and is only applicable to one stage of software development.

Common Weakness Enumeration (CWE) [179] provides a comprehensive list of generic software weaknesses, and ways to avoid them. Weaknesses are typically linked to multiple external taxonomies for easy classification. CWE is chiefly useful for software developers only, mitigations are only available in text form. They are not intended to be processable by software.

**Vulnerability Databases:**

Common Vulnerabilities and Exposures (CVE) [180] provides a central directory for all vulnerabilities across multiple vulnerability databases [181], [182], [183], [184], [185], and [186] aimed at users and system administrators instead of developers. CVE focus on treating the symptoms instead of the root causes of the vulnerabilities. The Open Vulnerability and Assessment Language (OVAL) [187] builds on CVE. OVAL is a machine-understandable language that describes security contents. Tools to create or process models already exist. Focus is on deciding whether a computer system is vulnerable to a certain attack. A drawback with OVAL is that its database is very large and redundant – there is no requirement to reuse similar tests.
4.14 Marketplace

4.14.1 Composition Repository (ATC)

The implementation knowledge and principles behind the Composition Repository (see section 5.4.1) could be exploited to develop the appropriate Storage Layer components for the Aniketos Marketplace. The envisaged enhanced Composition repository can offer the security and trustworthy specifications and requirements to the consumers of the Marketplace services.

4.14.2 Other related work to Marketplace

We do not consider in this section application marketplace technology (e.g. Android marketplace, App store, etc.) as relevant for the project.

4.14.2.1 Yellow pages (UDDI)

Yellow Pages are one of the three main components of Universal Description, Discovery and Integration (UDDI) (described in [3] and [243]) the specification for distributed web-based information registries of web services.

Yellow pages provide a classification of the services or businesses, based on standard taxonomies. As described in the specification [243], UDDI allows users to define multiple taxonomies. In such a way, multiple classification schemes can be overlaid on a single UDDI entity. Therefore taxonomies in a UDDI entity can be standard codes (such as the United Nations Standard Products and Services Code System (UNSPSC) [188]) or a new taxonomy created and distributed by the user.

The other two components of UDDI are White Pages for describing the business (including address, contact, and known identifiers), and Green Pages for describing the technical information about services that are exposed by the business including how to access the service.

4.15 Training material module

4.15.1 Automated Training Material Generator (SEARCH)

<table>
<thead>
<tr>
<th>Name</th>
<th>Automated Training Material Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Design technique/tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Provides a set of techniques and tools for creating customized training material (PowerPoint slides, hand-outs, etc.) for a variety of target audiences from a large training material component pool.</td>
</tr>
<tr>
<td>Support area</td>
<td>CS – Reference Architecture and Patterns</td>
</tr>
<tr>
<td>Developed by</td>
<td>SEARCH-LAB (SEARCH) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>In-house; continuous improvement in progress</td>
</tr>
<tr>
<td>References</td>
<td>SEARCH-LAB internal use for conducting trainings in the past</td>
</tr>
<tr>
<td>System requirements</td>
<td>Microsoft Office 2007 or more recent versions, .NET framework 3.5 or newer</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>The ATMG was created by SEARCH-LAB in order to help in creating and maintaining training slides for security trainings targeted at various audiences. The tools have evolved from the initial macros toward a more sophisticated set of software; the methodology was also improved during the last years based on SEARCH-LAB’s frequently conducted security trainings.</td>
</tr>
</tbody>
</table>
1. Short description

The ATMG is a tool and set of techniques for creating customized training material subsets (PowerPoint slides, hand-outs, etc.) for various target audiences from a large training material components pool. It is able to automatically generate different “flavours” of the training slides from the same base components depending on the needs of specific audience groups.

The system consists of the ATMG software, the configuration descriptions for the individual trainings, and the base training material component pool. We also established a methodology about how the system should be best utilized in everyday practice.

The content creators are contributing to the component pool, while the training manager is in charge of defining the requirements for an individual training until the training material has been generated. Then the presenter receives a fully customized, presentation-ready material package for his particular training session.

2. Making the ATMG part of Aniketos

The ATMG has been designed to best suit the needs of SEARCH-LAB’s training portfolio, but it can be used – with or without modifications – to serve other similar needs, like the internal/external trainings in Aniketos. The ATMG allows the creation of customised training material. It serves as an important collaboration tool within the training material module, enabling content creators to contribute to the pool of training material components in a decentralised way.

Aniketos will produce original content for the trainings coming from several partners. There will be trainings planned that combine input from various partners; there will also be trainings that are similar in scope but are targeted at different audiences. The ATMG is especially good in helping us to manage a number of slightly different presentations coming from various sources and presented by different presenters.

Preparing training materials by understanding the learning needs of the targeted users is very much in line with the UCD (User-centred Design and Interaction Design) methodology described in this document.

The best use of ATMG in Aniketos would be as a collaboration tool that is able to integrate results from all the partners instead of having separate, individual performances.

3. Development roadmap

Since the tool has been designed with SEARCH-LAB’s needs in mind, it may need a few modifications to be more efficient in the Aniketos environment. Our suggested development roadmap is the following:

Since the heart of the tool’s configurability is a well thought-out tag system, categorizing the pieces of the original material for further use, we recommend adapting it to Aniketos’ needs. It can be done after the training plans have been defined – after we know the various target audiences, and planned training topics.

Technical issues related to large numbers of input sources and possible incompatibilities between Office document versions should be addressed during the project.

4.15.2 Other related work to Training material module

With respect to external works relevant for the realisation of this module, we are not aware of any relevant technology that could be specifically used to realize this module. Further identification work shall be conducted within WP5.
5 Environment modules

In the next sections we provide the descriptions of the most relevant works with regards the Aniketos platform Environment. The Environment is fundamental for Aniketos platform to work but is not the core of the research within the project, so we mostly will benefit from available results from other research projects or industry. For this reason, the works presented in this section are all external works, except some few, where we have indicated the Aniketos partner that brought it to the project.

In the cases the work is a tool, we have used the same table to describe it as for the internal background.

5.1 Integrated environments

For the Aniketos Environment we can identify a number of service infrastructures that allow developers building, deploying and managing services. In the next tables we provide information on a selection of them according to their relevance due to they are widely used or because they provide features interesting for Aniketos. But for a more detailed comparison of business integration and business process automation software please refer to [188].

Some architectures are based on an enterprise service bus (ESB). An ESB is a software architecture construct which provides fundamental services for complex architectures via an event-driven and standards-based messaging engine (the bus). Developers typically implement an ESB using technologies found in a category of middleware infrastructure products, usually based on recognized standards.

An ESB takes the complexity out of integration, providing connectivity to a wide range of technologies and creating services that can be reused across an organization. An ESB does not itself implement a service-oriented architecture (SOA) but provides the features with which one may implement such.

Developers can exploit the features of an ESB in order to integrate applications and services without custom code. Developers can shield services, regardless of location, from message formats and transport protocols. Data can be transformed and exchanged across varying formats and protocols.

5.1.1 Oracle SOA Suite

<table>
<thead>
<tr>
<th>Name</th>
<th>Oracle SOA Suite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Oracle SOA Suite is a software suite that lets to build, deploy and manage Service-Oriented Architectures (SOA). The components of the suite are hot-pluggable and benefit from common capabilities including consistent tooling, a single deployment and management model, end-to-end security and unified metadata management. Therefore, Oracle SOA Suite enables system developers to set up and manage services and to orchestrate them into composite applications and business processes. Oracle SOA Suite is composed of hot-pluggable components, such us Oracle JDeveloper, Oracle Business Rules, Oracle BPEL Process Manager, Oracle Service Bus, etc. Figure 5 shows the Oracle SOA Suite components and their relationships.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS-*</td>
</tr>
</tbody>
</table>
### 5.1.2 WSO2 SOA Platform

**Name**  
WSO2 SOA Platform

**Type**  
Tool

**Purpose**  
WSO2 SOA Platform is not a product in itself, but a complete set of components such as: WSO2 Carbon, WSO2 Application Server, WSO2 Enterprise Service Bus, WSP2 Business Rules Server, WSO2 Governance Registry, WSO2 Identity Server, etc. The WSO2 Carbon is the base platform for WSO2’s enterprise-grade middleware stack based on OSGI technology. The WSO2 Carbon core platform hosts a rich set of middleware components encompassing capabilities such as service hosting and management, message routing and transformation, governance and identity management, business process management and business activity monitoring, and more. The point-and-click WSO2 Carbon Component Manager makes it a snap to select the perfect set of components to address your unique requirements.

Figure 6 shows the WSO2 SOA Platform components and their relationships.

**Support area**  
RS-*

**Developed by**  
WSO2 [external]

**Maturity**  
Mature.
5.1.3 SAP Enterprise Services Architecture

Name: SAP SOA Middleware
Type: Tool
Purpose: SAP SOA Middleware is "an open architecture for adaptive business solutions" that promotes 5 key principles: abstraction, modularity, standardised connectivity (allowing flexible service composition), loose coupling and incremental design. Enterprise SOA is not a product in itself, but products like SAP NetWeaver, SAP xApps, and SAP Business Suite adopt the SOA principles.

Support area: RS-*
Developed by: SAP [Aniketos]
Maturity: Mature
References: SAP and SOA [194]
SAP Enterprise Services Architecture [195]
System requirements: N/A.
Miscellaneous: Proprietary
5.2 Service composition framework

5.2.1 Service specification/planning mechanism

5.2.1.1 MUSIC studio and middleware (SINTEF)
MUSIC studio (see section 5.3.1.1) together with MUSIC methodologies provides a generic service composition framework for adaptive and context aware applications, but it does not handle specific security related aspects. MUSIC can benefit from Aniketos services, e.g., security requirements modelling and trustworthiness predictions.

5.2.1.2 Universal Service Description Language (SAP)

<table>
<thead>
<tr>
<th>Name</th>
<th>Universal Service Description Language (USDL) &amp; SDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>The Unified Service Description Language is a new conceptual model for describing business and technical services. The service delivery framework (SDF) is a reference architecture for service-oriented platforms supporting various aspects of service offerings (deployment, composition, trading, etc. Of services).</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>CS - Reference Architecture and Patterns</td>
</tr>
<tr>
<td>Developed by</td>
<td>SAP [Aniketos] (and partners from the Theseus/TEXO project [external])</td>
</tr>
<tr>
<td>Maturity</td>
<td>Work in progress</td>
</tr>
<tr>
<td>References</td>
<td>Internet of services [31]</td>
</tr>
<tr>
<td></td>
<td>Cardoso et al., 2010 [32]</td>
</tr>
<tr>
<td>System requirements</td>
<td>Windows (USDL Modelling Editor available as [33])</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>A binary (using a proprietary licence) release of the USDL Editor is expected within the next weeks. It is planned to release the USDL Editor under an Open Source licence.</td>
</tr>
</tbody>
</table>

1. Short description
The Unified Service Description Language (USDL) originated from the German lighthouse project Theseus/TEXO. USDL builds on models for describing business and technical services, and creates a unified description of related research efforts. It is not meant to replace other specifications in the technical service stacks, but aims to complement them by adding necessary business information. On the other side, it was not designed for targeting automated services only. USDL is generic enough to be used for the description of manual services that have no technical implementation.

The Service Delivery Framework (SDF) provides a dedicated way through which services can be provisioned and delivered beyond “firewalls”, out to global business networks. Beyond the classical access and orchestration considerations of SOA, the SDF will allow services to be discovered, repurposed and operationally optimized on a global scale, be it for: exposure beyond software registries to service marketplaces; new service innovations and channels through third-parties potentially unknown to original providers; re-hosting through low-cost cloud environments for SMEs and untapped consumer segments, outsourcing interoperability and other service delivery support.
functions through specialists like gateways and integrators – among various provisioning considerations.

2. Making USDL part of the Aniketos platform

USDL is a language for specifying services and service interfaces on many different abstraction levels ranging from abstract business requirements to technical system requirements. Still, USDL does not replace technical Web-service standards such as the WS-* family of standards. Thus, USDL contributes to the expressiveness of the service specifications that can be handled in the service specification/planning within the Service composition framework. It is planned to enrich USDL with means for specifying compositions. Still, the Service Composition Framework can benefit from USDL as USDL provides detailed descriptions of services that are relevant for building composed services.

The architecture of the SDF might be a starting point for the Aniketos Reference Architecture (WP5). Moreover, the “Public Sector Case Study” (available as [33]) might give some ideas for the Case Study C of Aniketos (in WP6). The SDF Public Sector Case Study demonstrates a service used as a single point of access for citizens to register a new business. The aim is to show how the Internet of Services could assist in breaking down barriers to cross-border trade, hence making it easier for service providers, especially Small Medium Enterprises (SME), to set up business and offer services in other jurisdictions.

3. Development roadmap

USDL is continuous development and a W3C Incubator Group working its standardization was founded recently. Both the design time support techniques for modelling trust (WP2) and service compositions (WP3) might profit from the already existing modules of USDL. Moreover, the transfer of the Aniketos requirements and the experiences (including the case studies carried out in WP6) in using SDL might influence the future development and standardization of USDL.

5.2.1.3 Other related work to Service specification/planning mechanism

MUSIC and USDL build on the standards for the technical IT description efforts for services such as WSDL. In particular, USDL supports economics, e.g. the specification of business and operational requirements.

There exists a plethora of service description efforts that focus just on single aspects. They can be grouped into six strands:

The first strand of service description efforts is the field of Service-oriented Architectures (SOA). Originally, several standards bodies specified several dozens of different aspects which are collectively known as WS-*. The OMG standards body dedicated its focus to software engineering for SOA, and, subsequently defined the Service-oriented architecture Modeling Language (SoaML), see [35]. Finally, the multitude of description efforts and different definitions of SOA led to a Reference Model for Service Oriented Architecture (SOA-RM) by OASIS (see MacKenzie et al, 2006 [36]) and a Reference Architecture Foundation for Service Oriented Architecture (SOA-RA) also by OASIS [37]. When it comes to web services architectures, the Business Process Execution Language for Web Services (WSBPEL or shortly BPEL) by OASIS [38] is a de-facto standard for modelling web services composition.

A second strand consists mainly of ontologies in the field of Semantic Web Services. As presented in the seminal paper, viz., McIlraith et al., 2001 [39], the main goal of Semantic Web Services approaches is automation of discovery, composition, and invocation of services in a SOA by ontology reasoners and planning algorithms. Prominent efforts are described in Ankolekar et al., 2001 [40] and Roman et al., 2006 [41]. Many surrounding and similar efforts have surfaced in literature, for example Dobson, Sánchez-Macián, 2009 [42], Farell & Lausen, 2006 [43], Gomadam et al., 2010 [44].

The third strand is rooted in the rise of on-demand applications that led to the notion of software-as-a-service (SaaS), covering software applications and business process outsourcing to cloud and platform services, see the W3C recommendation called Service Modeling Language (SML) as described in Pandit et al., 2009 [45].
The fourth strand draws attention mainly to describing Service Networks, i.e., the ecosystem and value chain relationships between services of economic value. So far, this strand did not output any standards and is represented by academic approaches only, for example see Akkermans et al., 2004 [46].

Fifth, there are overarching efforts that concentrate on the bigger picture of service systems or service science also taking into account socio-economic aspects, see Alter, 2008 [47] and Dhanesha et al., 2009 [48].

The final strand is driven by schools of business administration and business informatics and focuses on capturing the purely economic aspects of services regardless of their nature (with less or no focus on IT services and software architectures), for example see O’Sullivan et al., 2006 [49].

In the following sections we provide a number of specific technologies and projects that are relevant to be considered for the service specification mechanism.

5.2.1.3.1 SoaML tools

- **Enterprise Architect**

<table>
<thead>
<tr>
<th>Name</th>
<th>Enterprise Architect (EA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>EA is a modeling, visualization and design platform for business, software and systems. EA is based on the UML v2.3 standard and supports Archimate, BPMN, OMG SysML, SPEM and Business Rules.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS</td>
</tr>
<tr>
<td>Developed by</td>
<td>Sparx Systems [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature</td>
</tr>
<tr>
<td>References</td>
<td>Enterprise Architect tool [50]</td>
</tr>
<tr>
<td>System requirements</td>
<td>NA</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Proprietary</td>
</tr>
</tbody>
</table>

- **Cameo SOA+**

<table>
<thead>
<tr>
<th>Name</th>
<th>Cameo SOA+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Cameo SOA+ is a UML tool with SoaML modeling and provisioning that allows the creation of SOA business and systems architectures. Cameo SOA+ uses ModelPro MDA tooling from ModelDriven.org (see below) as provisioning engine.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS</td>
</tr>
<tr>
<td>Developed by</td>
<td>NoMagic [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature</td>
</tr>
<tr>
<td>References</td>
<td>Cameo SOA+ tool [51]</td>
</tr>
<tr>
<td>System requirements</td>
<td>NA</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Proprietary. Cameo™ SOA+ is packaged as a plugin to the MagicDraw® tool and is available for purchase separately.</td>
</tr>
</tbody>
</table>
- **ModelPro**

<table>
<thead>
<tr>
<th>Name</th>
<th>ModelPro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Open Source MDA provisioning for SoaML. SoaML Cartridge plugin for ModelPro automates the design and development of Service Oriented Architectures, initially on Java Platforms. Based on SoaML, the SoaML modeling profile supports the range of modeling requirements for service-oriented architectures, including the specification of systems of services, the specification of individual service interfaces, and the specification of service implementations.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS</td>
</tr>
<tr>
<td>Developed by</td>
<td>ModelDriven.org [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature</td>
</tr>
<tr>
<td>References</td>
<td>ModelPro tool [52]</td>
</tr>
<tr>
<td>System requirements</td>
<td>NA</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Open source, GNU public license version 2.</td>
</tr>
</tbody>
</table>

- **Objecteering**

<table>
<thead>
<tr>
<th>Name</th>
<th>Objecteering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>SoaML Profile for UML tool.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS</td>
</tr>
<tr>
<td>Developed by</td>
<td>Softeam [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature</td>
</tr>
<tr>
<td>References</td>
<td></td>
</tr>
<tr>
<td>System requirements</td>
<td>NA</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Proprietary</td>
</tr>
</tbody>
</table>

- **Rational Software Architect**

<table>
<thead>
<tr>
<th>Name</th>
<th>Rational Software Architect (RSA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>RSA is an integrated environment for managing complex solution architectures, general modeling and model-driven development, and deployment planning. The tool supports UML with SoaML and code generation. The latest version v8 includes the UML Modeler that provides BPMN 2 (Business Process Modeling Notation), UML 2 modeling, code visualization, and reconciled modeling support for Java™, C#, and VB.NET (Microsoft® Visual Basic® .NET).</td>
</tr>
</tbody>
</table>

[ANIKETOS]
5.2.1.3.2 Process support tools for web services

- **Oracle BPEL Process Manager**

  Name: Oracle BPEL Process Manager  
  Type: Tool  
  Purpose: Oracle BPEL Process Manager, a member of the Oracle SOA Suite family of products, enables enterprises to orchestrate disparate applications and Web services into business processes. The ability to quickly build and deploy these processes in a standards-based manner delivers critical functionality for developing a Service-Oriented Architecture (SOA) [198].

- **Apache ODE**

  Name: Apache ODE  
  Type: Tool  
  Purpose: Apache ODE (Orchestration Director Engine) is a WS-BPEL compliant web services orchestration engine. Apache ODE supports 2 communication layers: one based on Axis2 (Web Services http transport) and another one based on the JBI standard (using ServiceMix). It also supports for the HTTP WSDL binding, allowing invocation of REST-style web services.
### JOpera for Eclipse

<table>
<thead>
<tr>
<th>Name</th>
<th>JOpera for Eclipse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>JOpera for Eclipse is a rapid service composition tool offering a visual language and autonomic execution platform for building distributed applications out of reusable services, which include but are not strictly limited to Web services. You can invoke SOAP and RESTful Web services, Java snippets, Human activities, Grid services, ... JOpera provides integrated process modelling and execution.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS</td>
</tr>
<tr>
<td>Developed by</td>
<td>University of Lugano, Switzerland [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature</td>
</tr>
<tr>
<td>References</td>
<td>JOpera for Eclipse [200]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Proprietary (JOpera Licence - Redistribution and use in source and binary forms, with or without modification, are permitted for strictly non-commercial purposes provided that 3 special conditions are met. See <a href="http://www.jopera.org/licence">http://www.jopera.org/licence</a>)</td>
</tr>
</tbody>
</table>

### ZenFlow

<table>
<thead>
<tr>
<th>Name</th>
<th>ZenFlow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>ZenFlow is a visual composition tool for web services written in BPEL4WS. ZenFlow provides several visual facilities to ease the definition of a business process such as multiple views of a process, syntactic and semantic awareness, filtering, logical zooming capabilities and hierarchical representations. The visual composer and the engine supporting the deployment and execution of BPEL processes are both implemented in Java. The visual composer can be run either as a standalone Java application or as a plugin in the Eclipse platform. The visual composer is enriched with BPEL and WSDL compilers as well as the corresponding code generators to fully support BPEL processes.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS</td>
</tr>
<tr>
<td>Developed by</td>
<td>Universidad Politécnica de Madrid (UPM) [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype</td>
</tr>
<tr>
<td>References</td>
<td>Martínez et al., 2005 [201]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Proprietary</td>
</tr>
</tbody>
</table>
5.2.1.3.3 Projects

- **SOA4All**

SOA4All FP7 project [205] objective is to provide a comprehensive framework that integrates complementary and evolutionary technical advances (i.e., SOA, context management, Web principles, Web 2.0 and semantic technologies) into a coherent and domain-independent service delivery platform. The SOA4All project has developed an integrated tool SOA4All Studio (summarized below) that is relevant for the service composition framework within Aniketos.

<table>
<thead>
<tr>
<th>Name</th>
<th>SOA4All Studio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>The SOA4All Studio [206] developed within SOA4All project [205] is a rich web platform that provides users with a unified view covering the whole lifecycle of services, including design-time, run-time and post-mortem analysis. The tool enables the creation, provisioning, consumption and analysis of the platform services and various 3rd party business services that are published to SOA4All. SOA4All Studio has the SOA4All Distributed Service Bus as the infrastructural backbone around which all the components communicate and collaborate by combining Semantic Spaces and Enterprise Service Bus. The SOA4All Suite architecture can be seen in next Figure 7.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS-*</td>
</tr>
<tr>
<td></td>
<td>RS-*</td>
</tr>
<tr>
<td>Developed by</td>
<td>SOA4All consortium [external] (including ATOS [Aniketos] and SAP [Aniketos])</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype</td>
</tr>
<tr>
<td>References</td>
<td>SOA4All project [205]</td>
</tr>
<tr>
<td></td>
<td>SOA4All Studio [206]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Proprietary</td>
</tr>
</tbody>
</table>

ANIKETOS
The SLA@SOI Reference Architecture for an integrated multi-level Service Level Agreements (SLA) Management framework [210] is a result of the on-going SLA@SOI FP7 project [211]. The SLA@SOI SLA management framework aims at the management of service delivery systems in order to meet the QoS objectives (goals) specified in SLAs. Service Level Agreements are used for managing the non-functional aspects of the complete service lifecycle. Furthermore, SLA translations across different layers allow for consistent interlinking of complete service networks and hierarchies.

- supports SLA management across multiple layers with SLA (de-)composition across functional and organizational domains;
- supports arbitrary service types (business, software, infrastructure) and SLA terms;
- covers the complete SLA and service lifecycle with consistent interlinking of design-time, planning and run-time management aspects; and
- can be applied to a large variety of industrial domains.

The validation of the SLA management framework is still on-going and should be ready for the end of the project in June this year.

The Q-Impress project [212] aims at bringing service orientation to critical application domains, such as industrial production control, telecommunication and critical enterprise applications, where guaranteed end-to-end quality of services is particularly important.

One of the main goals of this project is to predict the impact of design decisions on performance, reliability and maintainability of the software, through quality impact analysis and simulation techniques. This capacity for prediction is based on models and worthy to be studied for Trust prediction of Aniketos Platform.
- **CHOReOS**
  The CHOReOS Large Scale Choreographies for the Future Internet project [213] will implement a framework for scalable choreography development. CHOReOS will deliver formally grounded abstractions and models, dynamic choreography-centric development processes, governance and service-oriented middleware manipulated via an Integrated Development Runtime Environment (IDRE) aimed at overcoming the ULS impact on software system development. This project is mainly executed in parallel of Aniketos, so collaboration with the CHOReOs consortium is suggested in order to enrich Aniketos platform with the progress of CHOReOs results.

- **NEXOF-RA**
  NEXOF-RA [214] developed within NEXOF-RA EU FP7 project [215] provides a reference architecture for service-based systems. As a reference for Aniketos, NEXOF-RA shall be considered when designing and building the modules of the platform aligned with NESSI Open Service Framework (NEXOF). NEXOF concept is composed by the following set of building blocks:
  - An Open Reference Model (ORM), a technology independent and application-domain independent conceptual model that defines the types of entities and relationships relevant for engineering and maintaining service based systems.
  - An Open Reference Architecture (ORA), (NEXOF-RA) the architectural specification of potential implementations of NEXOF, defined in order to formalise the reference model into open specifications facilitating a precise implementation of the service environment in different domains and technologies.
  - An Open Reference Implementation (ORI), the guide for further NEXOF instantiations by different organisations, for different domains and adopting different technological approaches.
  - A Compliance Test Suite, to validate each NEXOF instance and the related provided services, not only to be fully operational, but also to be compliant with the ORA in order to assure maximum interoperability.

The next Figure 8 shows the main components of the NEXOF-RA architecture.
5.2.2 Service validation mechanism
Identification of relevant technologies for service validation is planned for later in the project.

5.2.3 Service discovery mechanism

5.2.3.1 MUSIC studio and middleware (SINTEF)
MUSIC studio (see section 5.3.1.1) supports dynamic QoS-aware service discovery and binding.

5.2.3.2 Other related work to Service discovery mechanism
When looking for a suitable service for performing a task, service discovery mechanisms facilitate the search to consumers. For web services, usually service providers publish the web service interface through the use of a Web Services Description Language (WSDL) document. And optionally providers can register web services in a UDDI registry [3] and [243] and/or publish Web Services Inspection Language (WSIL) documents that facilitate the discovery. Other authors like Shijian et al., 2004 [216] propose agent-based services discovery mechanisms to increase efficiency in scenarios where services information documents are stored in distributed servers, and servers are regarded as independent agents.

5.3 Service runtime environment

5.3.1 Service execution mechanism
We have identified the following existing service runtime environments that could be adopted for Aniketos platform. For available technologies external to the consortium, a detailed collection and comparison of web service frameworks can be found in Wikipedia [217].

5.3.1.1 MUSIC studio and middleware (SINTEF)

<table>
<thead>
<tr>
<th>Name</th>
<th>MUSIC studio and middleware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool and Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>Support for development and execution of self-adaptive, context-aware mobile applications which are dynamically composed from components and services and reconfigurable at runtime</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS - Composite service analysis and preparation</td>
</tr>
<tr>
<td></td>
<td>RS - Composite service adaptation and recomposition</td>
</tr>
<tr>
<td></td>
<td>CS - Reference Architecture and Patterns</td>
</tr>
<tr>
<td>Developed by</td>
<td>MUSIC consortium [external] (including SINTEF [Aniketos])</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature, result of EU FP6 IP project</td>
</tr>
<tr>
<td>References</td>
<td>MUSIC project [216]</td>
</tr>
<tr>
<td></td>
<td>Rouvoy et al., 2009 [219]</td>
</tr>
<tr>
<td></td>
<td>Hallsteinsen et al., 2009 [220]</td>
</tr>
<tr>
<td></td>
<td>Jiang et al., 2010 [221]</td>
</tr>
<tr>
<td>System requirements</td>
<td>Java 1.4 or Java ME CDC, OSGi</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>MUSIC tools and middleware are open source, released under the LGPL license. They are available for download from <a href="http://developer.berlios.de/projects/ist-music/">http://developer.berlios.de/projects/ist-music/</a></td>
</tr>
</tbody>
</table>
1. Short description

MUSIC provides application developers with methodologies and an open platform for the development of self-adaptive context-aware mobile applications, which includes a runtime environment (MUSIC middleware) and development tools (MUSIC studio). MUSIC targets component-based applications that may be dynamically composed and reconfigured at runtime. MUSIC supports both component-oriented and service-oriented adaptation, and facilitates the building of “systems of systems” where self-adapting applications collaborate through dynamic QoS-aware service discovery and binding and adapt in a coordinated way driven by service level negotiations. The adaptation planning is based on the utilities of alternative configurations in response to context changes, where utilities are evaluated at runtime based on the property predictors of the composing components and services.

2. Making MUSIC part of the Aniketos platform

Generally we consider MUSIC platform as a runtime environment for running the case studies integrating with other components of the Aniketos platform. In particular, MUSIC middleware supports dynamic composition and adaptation of services and components. Thus MUSIC middleware can be integrated with Aniketos platform for runtime support. Specifically in relation with the security contract negotiation module, MUSIC platform provides runtime support for SLA negotiation, creation and monitoring.

MUSIC property predictor framework is generic and the property predictors can be used to model different types of properties, including security. The MUSIC visualization tool can simulate the runtime property evaluation. We can integrate Aniketos security modelling language with MUSIC’s property predictor framework and the visualization tool for design time support of security property definition and evaluation.

MUSIC provides dynamic QoS-aware service discovery. Discovery plug-in can be extended to support discovery based on security and trustworthiness properties. This is a core function enabling the composite service adaptation and recomposition. In addition, threat situation can be modelled as context information and monitored at runtime. When threat situation changes, MUSIC will adapt the affected services by e.g. reconfiguring the service or replacing a service with a more suitable one, based on utility functions calculated from property predictors and context information.

There can be SLAs specifying security and trust levels between provided services and consumed services. MUSIC SLA mechanism is generic, and can be extended to support security and trustworthiness in order to maintain secure and trusted composite services based on SLA negotiation, creation and monitoring. MUSIC can be used to monitor the threats and adapt the affected services using the SLA mechanisms at runtime.

MUSIC has ontology support including service ontology, which can be used for community support, e.g. for Aniketos marketplace. MUSIC architecture can also contribute to reference architecture and patterns.

3. Development roadmap

MUSIC needs the work from WP1, WP2, WP3 and WP4 as the basis for specifying security property predictors, in particular, the theory for computing the composed security properties, and the definition of security properties for threat-awareness. In addition, it needs to integrate the security-by-contract work from WP2 with the MUSIC SLA mechanisms. The runtime SLA monitoring mechanism needs also extension for elaborated service level comparison (e.g., consider only a subset of properties in the agreement, and the conditional comparison of offered and agreed service levels) depending on the modelling of security properties.

5.3.1.2 Other related work to service execution mechanism

MUSIC adopts a utility functions-based approach to self-adaptation, where utility functions assign a utility value to each application variant as a function of application properties, context and goals. QuA [222] is a QoS aware adaptation framework also based on utility functions and quality predictors. It
calculates predicted quality using predictors and specifies quality requirements and adaptation policies using utility functions that map quality prediction to a scalar value. Rainbow [223] uses component-based architecture model and adaptation strategies based on situation-action rules are scored using utility preferences specified for the quality dimensions, where the adaptation manager selects the highest scoring strategy.

Menasce and Dubey [224] propose a QoS brokering approach in SOA. Consumers request services from a QoS broker, which selects a service provider that maximizes the consumer’s utility function with regards to its cost constraint. MUSIC approach differs in that the offered properties are considered as alternatives to determine the best application configuration and the client is allowed to adapt to the service landscape.

Adaptive Service Grids (ASG) [225] and VIEDAME [226] are initiatives enabling dynamic compositions and bindings of services for provisioning adaptive services. In particular, ASG proposes a sophisticated and adaptive delivery lifecycle composed of three sub-cycles: planning, binding, and enactment. VIEDAME proposes a monitoring system that observes the efficiency of BPEL processes and performs service replacement automatically upon performance degradation. However, none of the approaches support uniform planning of both components and services as MUSIC’s planning-based framework for ubiquitous applications does.

Similar to the MUSIC approach, CARISMA [227] also addresses self-adaptation supported by generic middleware, but focuses on adaptation of middleware level services.

There are several existing approaches to adaptation based on dynamic software product line techniques as MUSIC does. DiVA [228] uses model-driven engineering and aspect-oriented modelling techniques to support runtime variability. The Genie approach [229] also uses architecture models to support the generation and execution of dynamically adaptive systems based on component-based middleware techniques. The SESAMO approach [230] reuses design variability models during runtime to tackle runtime variability and develops a framework for building context aware and self adaptive systems relying on context sensors to monitor relevant context.

In the following sections we provide a number of specific technologies that are relevant to be considered for the service execution mechanism.

5.3.1.2.1 Apache Axis2

<table>
<thead>
<tr>
<th>Name</th>
<th>Apache Axis2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Apache Axis2™ is a Web Services / SOAP / WSDL engine, the successor to the widely used Apache Axis SOAP stack. Axis2 not only allows adding Web services interfaces to Web applications, but can also function as a standalone server application. Axis 2 also has integrated support for the REST style of Web services. Axis2 architecture is flexible so its functionalities can be extended using modules. Apache Rampart module provides WS-Security standard features to Axis2 and Apache Sandesha2 is a module that implements WS-Reliable Messaging specification.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS</td>
</tr>
<tr>
<td>Developed by</td>
<td>Apache Software Foundation [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature</td>
</tr>
<tr>
<td>References</td>
<td>Apache Axis2 [231]</td>
</tr>
<tr>
<td></td>
<td>Apache Web services [232]</td>
</tr>
</tbody>
</table>
Apache Axis2 modules [233]

System requirements N/A
Miscellaneous Open source under Apache license.

### 5.3.1.2.2 Apache CXF

<table>
<thead>
<tr>
<th>Name</th>
<th>Apache CFX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Apache CXF is an open source web service framework that supports a number of standards such as SOAP, WS-Addressing, WS-Policy, WS-ReliableMessaging, WS-Security, WS-SecurityPolicy, and WS-SecureConversation. CXF is often used with Apache ServiceMix, Apache Camel and Apache ActiveMQ in Service-oriented architecture (SOA) infrastructure projects.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS</td>
</tr>
<tr>
<td>Developed by</td>
<td>Apache Software Foundation [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature</td>
</tr>
<tr>
<td>References</td>
<td>Apache CXF [234] Apache CXF community [235]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Open source under Apache license.</td>
</tr>
</tbody>
</table>

### 5.3.1.2.3 Apache Synapse

<table>
<thead>
<tr>
<th>Name</th>
<th>Apache Synapse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Apache Synapse “is a mediation framework that allows users to get in the middle between service requesters and providers and perform various tasks - including transformation and routing and that helps to promote loose coupling between services” [236]. Therefore, it is not a ESB in itself. Synapse supports for XML, Web Services and REST and content interchange formats, such as plain text, binary, Hessian and JSON.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS</td>
</tr>
<tr>
<td>Developed by</td>
<td>Apache Software Foundation [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature</td>
</tr>
<tr>
<td>References</td>
<td>Apache Synapse [237]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Open source under Apache license.</td>
</tr>
</tbody>
</table>
### 5.3.1.2.4 Mule ESB

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th>Mule ESB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Tool</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Mule ESB offers a simplified development model and lightweight architecture, which allow developers to easily create and integrate application services. Mule ESB takes the complexity out of integration, enabling developers to easily build high-performance, multi-protocol interactions between heterogeneous systems and services. One of the most important integration APIs is the specification of the service messaging types (SOAP/REST).</td>
</tr>
<tr>
<td><strong>Support area</strong></td>
<td>RS - Trustworthiness monitoring and evaluation RS - Runtime validation of secure service behaviour RS - Composite service adaptation and recomposition</td>
</tr>
<tr>
<td><strong>Developed by</strong></td>
<td>Mule ESB community [external]</td>
</tr>
<tr>
<td><strong>Maturity</strong></td>
<td>Mature; over 1.5 million downloads and 2,500 production deployments.</td>
</tr>
<tr>
<td><strong>References</strong></td>
<td>Enterprise Service Bus (ESB) [196] Mule ESB [197]</td>
</tr>
<tr>
<td><strong>System requirements</strong></td>
<td>-</td>
</tr>
<tr>
<td><strong>Miscellaneous</strong></td>
<td>Open source (CPAL: OSI-approved license, bundles other open source libraries that are made available under their respective licenses).</td>
</tr>
</tbody>
</table>
5.3.1.2.5 GlassFish

Name: GlassFish
Type: Tool
Purpose: GlassFish is an open source application server for the Java EE platform. GlassFish has a modular and extensible architecture based on OSGi. The GlassFish implementation includes open-source implementations of all of component technologies in the former Java Web Services Developer Pack (Java WSDP).
Support area: RS
Developed by: Oracle (Sun Microsystems) [external]
Maturity: Mature
References: GlassFish community [174]
GlassFish wiki [238]
System requirements: N/A
Miscellaneous: GlassFish is free software, dual-licensed under two free software licences: the Common Development and Distribution License (CDDL) and the GNU General Public License (GPL) with the classpath exception.
### 5.3.1.2.6 Swordfish

<table>
<thead>
<tr>
<th>Name</th>
<th>Swordfish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Swordfish is an extensible open source SOA runtime framework. The framework is based on the Eclipse Equinox (the Eclipse Foundation’s OSGi implementation.) runtime technology and it is designed to be complemented by additional open source components such as a service registry, a messaging system, a process engine etc. to form a comprehensive open source SOA runtime environment. Swordfish is built on proven open source components such as Apache ServiceMix and Apache CXF, and allows application developers and system integrators alike to build their own ESB that can be tailor-made to their requirements.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS</td>
</tr>
<tr>
<td>Developed by</td>
<td>Swordfish open source community [external]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype. Project in incubation phase</td>
</tr>
<tr>
<td>References</td>
<td>Swordfish SOA Runtime Framework project [193]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Open source</td>
</tr>
</tbody>
</table>

### 5.3.1.2.7 WSO2 Web Service Framework

<table>
<thead>
<tr>
<th>Name</th>
<th>WSO2 Web Service Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>WSO2 Web Services Frameworks are open source frameworks that provide developers the infrastructure to create and consume Web services in their favourite scripting language. It is a comprehensive Web services enabler allowing developers to focus on processing business logic rather than on infrastructural aspects such as WSDLs and Qualities of Service (QoS) functionality. Based on C, the frameworks extend support for other scripting languages such as PHP, C++, Perl, Spring, Ruby, Python and Jython. The three products Web Services Framework for PHP/for C/for C++ deliver comprehensive WS-* based Web Services support for the PHP/C/C++ world. With WSF the applications can acquire enterprise grade web services capabilities and seamlessly integrate with other Java or .Net systems using web services. WSO2 WSF/PHP is based on the WSF/C framework for providing/consuming Web services in PHP (including REST services) and is the only extension that supports the full Web services (WS*)-stack including security and reliable messaging.</td>
</tr>
<tr>
<td>Support area</td>
<td>RS</td>
</tr>
<tr>
<td>Developed by</td>
<td>WSO2 [external]</td>
</tr>
</tbody>
</table>
5.3.1.2.8  .NET Framework

Name        .NET Framework
Type        Tool
Purpose     The Microsoft .NET Framework is a software framework for Microsoft Windows operating systems. It includes a large library and supports several programming languages.
            The .Net Framework consists of:
            o Common Language Runtime – abstraction layer over the operating system
            o Base Class Libraries
            o Development frameworks and technologies

The Microsoft .NET Framework SDK tools are aimed at creating, deploying, and managing applications and components that target the .NET Framework.

“Web Services Enhancements for Microsoft .NET (WSE) is a .NET class library for building Web services using the latest Web services protocols, including WS-Security, WS-SecureConversation, WS-Trust, and WS-Addressing. WSE allows you to add these capabilities at design time using code or at deployment time.
through the use of a policy file.” [240]

Support area RS
Developed by Microsoft [external]
Maturity Mature
References .NET Framework [241]
System requirements Microsoft Windows OS. Microsoft's SDK for Windows 7 or .NET Framework 4 (or newer) or Visual Studio 2010 installed is required to develop applications in .NET Framework.
Miscellaneous Proprietary

5.3.2 Service adaptation mechanism

5.3.2.1 MUSIC studio and middleware (SINTEF)
MUSIC studio (see section 5.3.1.1) supports both component-oriented and service-oriented adaptation, through dynamic QoS-aware service discovery and binding, which allows applications adapting in a coordinated way driven by SLA negotiations.

5.3.2.2 Other related work to Service adaptation mechanism

5.3.2.2.1 Dynamic and Static Aspect Weaving

Name Dynamic and Static Aspect Weaving (DSAW)
Type Tool/Technique
Purpose Runtime adaptation of software services
Support area RS - Composite service adaptation and recomposition
Developed by University of Oviedo [external]
Maturity Mature, open source
References Ortín et al., 2005 [202]
ReadyAOP/DSAW [203]
System requirements .NET
Miscellaneous DSAW can be downloaded from the website [203]. It has an MIT/BSD style licence.

1. Short description
Aspect Oriented Programming (AOP) provides a way to allow the cross-cutting functionalities of a system – referred to as aspects – to be added into a program in an effective manner. Examples of cross-cutting concerns are logging and profiling capabilities, security issues, etc.

The DSAW platform provides a way to inject such aspects. However, unlike many other AOP platforms, DSAW utilises structural reflection to inject these aspects dynamically at run-time. This provides a potentially effective way to dynamically adapt a software service at run-time in order to adjust its security properties.

2. Making DSAW part of the Aniketos platform
DSAW has been designed for use with .NET programs. However, similar tools are also available for Java and other languages (e.g. AspectJ [204]). Incorporating this functionality into the Aniketos platform may therefore depend on the development platform and tools chosen for Aniketos.
Using AOP to adapt services effectively presents a number of challenges. The DSAW platform provides the structural reflection mechanisms required to inject code into existing services. However, the means of defining join points and aspects has been designed for weaving concerns through only a single application. Moreover in general it is assumed that some specific characteristics of the program (e.g. the names given to particular methods) are known. Aniketos would require aspects to be woven across multiple services simultaneously (e.g. in the case of the encryption example above, injecting encryption into just the sending service will cause problems; decryption functionality is needed at the receiving end too). Aniketos would require some kind of orchestration method in order to adjust the security properties of the services. This orchestration would have to be dependent on the security requirements, services, threat characteristics and so on of the system.

Aniketos would benefit by allow security properties of services to be dynamically adjusted at run-time. This could for example be in response to threats, or changes in policy. Moreover, the security properties of existing services could be changed, even when the source code of those services is unavailable.

3. Development roadmap

The work of extending and integrating this functionality into Aniketos could form part of Task 3.3 or Task 4.4.

5.3.3 Service recomposition mechanism

5.3.3.1 MUSIC studio and middleware (SINTEF)

MUSIC studio (see section 5.3.1.1) together with MUSIC methodologies provides a generic service composition framework for adaptive and context aware applications, but it does not handle specific security related aspects. MUSIC can benefit from Aniketos services, e.g., security requirements modelling and trustworthiness predictions.

5.3.3.2 Other related work to Service recomposition mechanism

5.3.3.2.1 Security-Oriented Service Composition and Evolution

This methodology (see section 4.5.5.1) is based on automated re-negotiation for system evolution aiming to re-establish security compatibility and security requirements satisfaction in a changed context.

5.3.4 Service monitoring module

5.3.4.1 Property and Policy Tools (LJMU)

Property & Policy Tools (see section 4.1.13) have the mechanism to monitor or detect changes of the network topology and data flow. For example, once a new connection established in the monitored environment, the tools will send a notification message to the server, where it will be signaled out on the screen; or it will display a warning message to the end users. This mechanism can be applied to monitor new services and trigger some Aniketos activities in response.

5.3.4.2 Platform for Run-time reconfigurability of security (ATOS)

PRRS (see section 4.6.4) allows monitoring of the execution of the services at runtime through the use of Monitors that are external entities that have the responsibility of performing dynamic analysis of proper operation of S&D solutions provided by the framework. The PRRS has an event management mechanism in which the Event Collectors send the events to the Event Manager, which is in charge of resending them to the Monitoring Service to be analyzed. Thus, the PRRS is also relevant for the service monitoring module.
5.3.4.3 Other related work to Service monitoring module
We are currently not aware of any suitable relevant work that could be used for this module, therefore technological surveillance should continue within WP4.

5.3.5 Context sensor

5.3.5.1 MUSIC studio and middleware (SINTEF)
The service adaptation planning in MUSIC studio (see section 5.3.1.1) is based on the utilities of alternative configurations in response to context changes (e.g. changes in threats), where utilities are evaluated at runtime based on the property predictors of the composing components and services. This functionality is relevant for the context sensor module.

5.3.5.2 Platform for Run-time reconfigurability of security (ATOS)
PRRS (see section 4.6.4) is relevant for this module because it is a run-time platform that provides management of security and dependability solutions and monitoring of the system’s context data (S&D framework configuration, active patterns, event history). PRRS supplies the necessary and proper security solution to applications during runtime, upon request and it is context aware. This means the provided solution changes with the execution environment.

5.3.5.3 Other related work to Context sensor

5.3.5.3.1 The Contextual Service Adaptation Framework
The Contextual Service Adaptation Framework [209] developed within SOA4All project [205] is a platform for supporting the adaptation of services to diverse contexts according to a virtually infinite variety of dimensions. The context and entity definitions within this framework:

“Context” is any information that can be used to characterize the situation of an entity. And “an entity” is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.

In order to use this framework the context and entity definitions must be suitable for Aniketos platform.

5.4 Service registry

5.4.1 Composition Repository (ATC)

<table>
<thead>
<tr>
<th>Name</th>
<th>Composition Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool</td>
</tr>
<tr>
<td>Purpose</td>
<td>Repository for storing and managing service compositions</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Composite service analysis and preparation</td>
</tr>
<tr>
<td>Developed by</td>
<td>ATC [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype</td>
</tr>
<tr>
<td>References</td>
<td>SODIUM Project [242]</td>
</tr>
<tr>
<td>System requirements</td>
<td>MySQL 4.1 Community Edition</td>
</tr>
<tr>
<td></td>
<td>MySQL Connector/J – 3.1</td>
</tr>
<tr>
<td></td>
<td>Java SDK 1.5 or greater</td>
</tr>
<tr>
<td></td>
<td>Tomcat 4.1.31 (Windows installer)</td>
</tr>
</tbody>
</table>
1. Short description
The Composition Repository is a storage area of the service compositions created by developers using the SODIUM [242] tools for both documentation and reuse purposes. It is an internal module accessible by developers within an organisation.
Beyond the SODIUM project, the Composition Repository can be used within organisations to support reuse of service compositions and improve the management of the internal implementation of services.
The tool can evolve into a registry publishing directory listings of resulting service compositions.

2. Making the Composition Repository part of the Aniketos platform
This tool can be extended to support the security assertions envisaged in Aniketos, in order to build secure composite services. The tool will be customised to the needs of the Aniketos project and will be exploited during the integration work.

3. Development roadmap
The knowledge gained in the SODIUM project can be extended in Aniketos to provide this tool with advanced capabilities with respect to discovery of service compositions, as well as the Aniketos Marketplace. Thus, this tool is envisaged to be further developed in Aniketos to support the Service Registry environment component by providing a semantic knowledge behind this tool implementation logic thus enabling the appropriate mechanisms for the discovery of security enabled service compositions. On top of that, the idea behind this tool will be exploited to develop the Aniketos Marketplace, as it will provide the implementation principles for offering security and trustworthy specifications and requirements to the consumers of the Marketplace services.

5.4.2 Other related work to Service registry

5.4.2.1 Universal Description, Discovery and Integration
As described in [3], Universal Description, Discovery and Integration (UDDI) [243] is a platform-independent, Extensible Markup Language (XML)-based registry for businesses worldwide to list themselves on the Internet and a mechanism to register and locate web service applications. UDDI is an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS), enabling businesses to publish service listings and discover each other and define how the services or software applications interact over the Internet.
UDDI is designed to be interrogated by SOAP messages and to provide access to Web Services Description Language (WSDL) documents describing the protocol bindings and message formats required to interact with the web services listed in its directory.

5.4.2.2 Oracle Service Registry
Oracle Service Registry within Oracle SOA Suite [190] (therefore, proprietary) is compatible with UDDI v3 (see above) and enables dynamic discovery of services and policies at runtime. As stated in [191], Oracle Service Registry provides a 'DNS'-like reference for SOA runtime infrastructure to dynamically discover and bind to deployed services and end points.

5.5 Aniketos compliant service
Aniketos compliant service refers to any service being Aniketos compliant, i.e. it offers its specifications in a way that is compliant and usable by Aniketos platform. And as such, the Aniketos services are not part of the Aniketos platform; hence they belong to the Environment. The project will define how services can become Aniketos compliant.
5.6 Identity management service

5.6.1 ED DIAMETER AAA (ELSAG)

Name | ED DIAMETER AAA
---|---
Type | Tool
Purpose | Authentication, Authorization & Accounting (AAA)
Support area | RS-*
Developed by | ElsagDatamat S.p.A. (ELSAG) [Aniketos]
Maturity | Prototype
References | EUQOS project [244]
System requirements | JRE, JDK, JBoss Rules
Miscellaneous | Proprietary

1. Short description
ED Diameter AAA is a Java-based implementation of the DIAMETER protocol [246] in which the Authentication, Authorization & Accounting (AAA) rules are defined by a rule language and evaluated by a rule engine (JBoss Rules). The software framework is mainly composed by a AAA Client software module and a AAA Server software module, acting respectively as the authenticator and the Authentication Server. These two peers accomplish their communication through the DIAMETER protocol, which is a recent protocol used for AAA purposes which overcomes RADIUS protocol’s [247] defects.

2. Making ED Diameter AAA part of the Aniketos platform
The ED Diameter AAA tool can be usefully used inside WP5 as runtime support for service providers. It is highly probable that the tool will need to be further modified and refined in order to cope with Aniketos’ needs and to be fully integrated with the other Aniketos platform tools. ED Diameter AAA is an Authentication, Authorization and Accounting Framework, and as such, it can be allocated in Identity Management service, but with some reservations: ED Diameter AAA manages user access to network resources, it grants services to the requesting user and it collects accounting data, but it doesn’t encompass all the tasks required by an Identity Management System, such as the creation of the user identities. Therefore, ED Diameter AAA should be modified and enhanced or integrated with other tools to build the Identity Management service needed by Aniketos platform.

We will have to wait for future WP1 results in order to understand the precise requirements and usage scenarios where ED Diameter AAA may be used, but the tool has proven itself as a flexible framework for heterogeneous communication protocols and contexts. See also Appendix A for a more detailed description of contexts where it has been successfully used.

3. Development roadmap
The first prototype of ED Diameter AAA was developed in the FP6 project EuQoS [244]; its main target was addressing QoS features, by adding to wireless connections a raw but working set of AAA features. The prototype implementation was further enriched in the FP6 project WEIRD [245] by applying it to WiMAX [248] connections and by adding a first rule engine. Then it was also used in the Italian authorities funded project M3-CAST for testing the feasibility of a seamless integration of DVB-H with UMTS [250].

Elsag Datamat aims to use and improve this framework, which has proven itself as a flexible tool, in the context of Aniketos project, by specializing it to Aniketos peculiarities and by enhancing the rule-based features of the tool.
More specifically ED diameter AAA might be usefully exploited in the context of telecommunication case study in WP6, where the evolution towards federated identity management has to be supported. In order to achieve this result it has to be further investigated if ED Diameter AAA can be modified so that it could act as an Identity Provider, upon which to build a federated identity management system. If at M16 it will be established that ED Diameter AAA is well-suited for Aniketos purpose, we will bring the needed changes within M24.

5.6.2 Other related work to Identity management service

Identity management (or ID management, or simply IdM) is a broad administrative area that deals with identifying individuals in a system (such as a country, a network, or an organization) and controlling access to the resources in that system by placing restrictions on the established identities of the individuals.

Federated identity, or the "federation" of identity, describes the technologies, standards and use-cases which serve to enable the portability of identity information across otherwise autonomous security domains. The ultimate goal of identity federation is to enable users of one domain to securely access data or systems of another domain seamlessly, and without the need for completely redundant user administration. Typical use-cases involve things such as cross-domain, web-based single sign-on (SSO), cross-domain user account provisioning, cross-domain entitlement management and cross-domain user attribute exchange.

Federation is enabled through the use of open industry standards and/or openly published specifications, such that multiple parties can achieve interoperability for common use cases.

Including IdM in the system will help to:

- improve User experience in accessing composite services (i.e. single-sign-on)
- contribute to keep the level of trust among Service Providers and Users by leveraging Identity Providers.

In the following a short overview of the mainstream initiatives is provided:

**WS-Federation**, is an Identity Federation specification by OASIS [251], developed by a consortium of companies including BEA Systems, IBM, Microsoft, Novell and VeriSign. Part of the larger Web Services Security framework (WS-*), WS-Federation defines mechanisms for allowing disparate security realms to broker information on identities, identity attributes and authentication. With WS-Federation, companies can federate their users between WS-Federation interoperable products and services.

**Windows CardSpace**, formerly known by its codename InfoCard [252], is a framework developed by Microsoft which securely stores digital identities of a person, and provides a unified interface for choosing the identity for a particular transaction, such as logging into a website. Windows CardSpace is built on top of Web Services Protocol Stack by OASIS [253], an open set of XML-based protocols, including WS-Security, WS-Trust, WS-MetadataExchange and WS-SecurityPolicy. CardSpace stores references to users' digital identities for them, presenting them to users as visual Information Cards. CardSpace provides a consistent UI designed to help people to easily and securely use these identities in applications and web sites where they are accepted.

**Liberty Alliance**, also known as Project Liberty [254], is an industry group formed in 2001 by approximately 30 organizations with the goal of developing open technical, business and privacy standards for federated identity management. Liberty Federation, which consists of the Liberty Identity Federation Framework (ID-FF) [256] 1.1, 1.2 and SAML 2.0 [255] specifications, allows consumers and users of Internet-based services and e-commerce applications to authenticate and sign-on to a network or domain from any device and then visit or take part in services from multiple Web sites. This federated approach does not require the user to re-authenticate and can support privacy controls established by the user.
Project Liberty has also created a standard that can be used to integrate SOA applications as well as other legacy client–server applications to the standards-based authentication and authorization infrastructure. The Liberty ID-WSF (WSF - Web Services Framework) provides the framework to implement the necessary components into the client and server applications for integration. The most recent version of the specification is Liberty ID-WSF 2.0.

As of June 2009, the work of the Liberty Alliance is transitioning to the Kantara Initiative [257].

**Security Assertion Markup Language 2.0 (SAML 2.0)** [255] is a version of the SAML OASIS standard for exchanging authentication and authorization data between security domains. SAML 2.0 is an XML-based protocol that uses security tokens containing assertions to pass information about a principal (usually an end-user) between an identity provider and a web service. SAML 2.0 enables web-based authentication and authorization scenarios including single sign-on (SSO).

SAML 2.0 was ratified as an OASIS Standard in March 2005, replacing SAML 1.1. The critical aspects of SAML 2.0 are covered in detail in the official documents SAMLConform, SAMLCore, SAMLBind, and SAMLProf. Some 30 individuals from more than two dozen companies and organizations were involved with the creation of SAML 2.0. In particular, and of special note, Liberty Alliance donated its Identity Federation Framework (ID-FF) specification to OASIS, which became the basis of the SAML 2.0 specification. Thus SAML 2.0 represents the convergence of SAML 1.1, Liberty ID-FF 1.2, and Shibboleth 1.3.

**OpenID** [163] is an open, decentralized, free framework for building single sign-on systems. OpenID starts with the concept that anyone can identify themselves on the Internet the same way websites do with a Uniform Resource Identifier (URI), also called a URL or web address. Since URIs are at the very core of Web architecture, they provide a solid foundation for user-centric digital identity. On OpenID-enabled sites, Internet users do not need to register and manage a new account for every site before being granted access. Instead, they only need to be previously registered on a website with an OpenID “identity”. Today, OpenID has emerged as the de-facto user-centric identity framework allowing millions of people to interact online.

**Higgins** [258] is an extensible, platform-independent, protocol-independent, software framework to support existing and new applications that give users more convenience, privacy and control over their identity information. It is an open source Eclipse-based project for building Internet identity frameworks designed to integrate identity, profile, and social relationship information across multiple sites, applications, and devices.

Basically, Higgins provides users with PC/mobile clients and personal data services (PDSs). These allow users to log into websites by using a variety of methods (e.g. information cards, OpenID or username/password) and to efficiently manage their digital identities by sharing selected aspects of their information with people and organizations that they trust. In addition, Higgins provides two identity provider services: the first is based on a Security Token Service (STS) supporting WS-Trust; the second supports SAML 2.0. Finally, Higgins also includes the “relying party” libraries necessary to enable websites and systems to request and accept information cards.
6 Usability and acceptance of the platform

In this section we describe other existing work, which is not exactly relevant because it could be integrated into a module or component of the platform, but because it is interesting for the platform to accomplish its objectives in terms of usability and user acceptance.

6.1 User-centred Design and Interaction Design (PLUS, DBL)

<table>
<thead>
<tr>
<th>Name</th>
<th>User Centred Design and Interaction Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Design Approach, Design Techniques</td>
</tr>
<tr>
<td>Purpose</td>
<td>User-centred design (UCD) is an approach to design that grounds the process in information about the people who will use the product. It ensures the consideration of user-related issues (usable security, user trust, user acceptance) in the development of secure and trustworthy systems. Interaction Design refers to the shaping of interactive and composite products and services with a specific focus on their final use.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS -- Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS -- Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS -- Composite service analysis and preparation</td>
</tr>
<tr>
<td></td>
<td>CS -- End user trust and assurance</td>
</tr>
<tr>
<td>Developed by</td>
<td>DBL [Aniketos] customised techniques developed by external partners in last decades and well know in literature, merging them in an integrated approach, tailored for safety-critical domains such as the Air Traffic Management (ATM). PLUS [Aniketos] also continuously applies and further develops methods and techniques for UCD in various domains, for instance the development of UX patterns for audio-visual networked systems.</td>
</tr>
<tr>
<td>Maturity</td>
<td>High maturity, applied both in commercial and R&amp;D projects in various domains</td>
</tr>
<tr>
<td>References</td>
<td>ISO 9241-210:2010 [259]</td>
</tr>
<tr>
<td></td>
<td>Cooper et al., 2007 [260]</td>
</tr>
<tr>
<td>System requirements</td>
<td>N/A</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1. Short description

From a Human-Computer Interaction (HCI) and Usability viewpoint, an understanding of users and their interaction with the interface of a system is central for the creation of usable systems. In contrast to other design approaches, User-centered design (UCD) starts with investigating the needs, wants, and limitations of users, on which all further steps of the design and development process are based. Thus, when deploying a UCD approach in system development, user acceptance and actual usage of the system will be at a higher level. At large, UCD defines a general process for including human-centered activities throughout a development life-cycle by focusing on users through the planning, design and development of a product.
The UCD approach is a complement to other software development approaches rather than a replacement for them. System development following such a design approach is characterized by an investigation of user and task requirements and an active involvement of end users in the system development lifecycle. In this lifecycle, user requirements elicitation, design activities and user requirements evaluation tasks take place in iterations.

2. Making the User-centered Design Approach part of the Aniketos platform

There have to be considered two types of users in the Aniketos project: (a) (Composite) service providers and developers, who use the Aniketos platform to define and maintain the security and trustworthiness of services, and (b) (composite) service end users, who are going to use the composed services. Thus, the UCD approach can be applied in the Aniketos project in two ways: First of all for the development of the Aniketos platform consisting of tools and techniques, and secondly for the realisation of the case studies based on the platform.

If applied in the development of the Aniketos platform, the central objective of this approach is to gain an understanding of the needs, wants, and limitations of future users of the Aniketos platform and to use this understanding of users as guidance throughout the development process. One of the main challenges for effectively applying UCD activities in Aniketos is to align UCD activities with the overall Aniketos development process.

According to the IEI model of Metzker and Offergeld, 2001 [261], the first out of five steps for the integration of UCD (respectively Human-centered design (HCD)) into the Aniketos platform development process is to analyse the software development process, which will be practiced in Aniketos. Secondly, UCD activities that are currently practiced have to be analysed. Aniketos partners (e.g. DBL and PLUS) can contribute various methods and techniques for eliciting user requirements for the Aniketos platform modules, e.g. personas and scenarios, contextual inquiry, ethnographic methods, cultural probing, user questionnaires and interviews, activity and task analysis. In addition, various methods and techniques for analysing the compliance of the user requirements in the platform are available e.g. qualitative and quantitative analysis of system performances, experience sampling methods, field observations, usability evaluations, expert judgment, and Critical Interaction Analysis Methodology (CRIA) [262].

The third step is to select suitable UCD tasks/activities and integrate them into the practiced software development process. The development roadmap section below gives some UCD task examples, which could be integrated into the design and development of the Aniketos platform. The main activities in the UCD approach comprise the following:

1. Understand and specify the context of use of the Aniketos platform
2. Specify the user and organizational requirements for the Aniketos platform
3. Produce design solutions for the Aniketos platform
4. Evaluate the design solutions against the previously specified requirements

Common UCD process models and methods have to be tailored to fit to the specific system development process characteristics of the various Aniketos partners/stakeholders. Thus, characteristics like for instance system domain, development team size, experience and skills of the development team (e.g. knowledge and skills in applying UCD methods, internal or external human factors experts contributing UCD related knowledge and skills to system development), and current system development practices (i.e. applied development process models and methods, e.g. waterfall model, spiral model, fountain model, formal methods, agile methods...) have to be considered.

The fourth step for the integration of UCD consists of supporting the effective performance of the defined UCD activities in Aniketos, and the final step is to collect and disseminate best practices and artefacts concerning UCD tasks.

3. Development roadmap

Some examples are given below, how UCD could be integrated into the design and development of the Aniketos platform and composite services:
<table>
<thead>
<tr>
<th>Aniketos task examples</th>
<th>UCD process activities with UCD application examples</th>
</tr>
</thead>
</table>
| **WP1/T1.2 Socio-technical security modelling language and tool** | (1) Understand and specify the context of use:  
- Identify platform stakeholders (types of service developers) who will use the socio-technical security modelling languages and tool  
- Consider existing (work) practices of requirements modelling language users and their experiences with previous modelling languages  
(2) Specify the user and organizational requirements:  
- Identify and consider specific characteristics and needs of platform stakeholders to ensure their acceptance of the modelling language and tools  
(3) Produce design solutions:  
- Build the socio-technical security modelling language and tool based on the specifications of (1) and (2)  
- Involve future users of the modelling language and tool in the design process |
| **WP1/T1.3 Aniketos platform requirements and scenarios** | (1) Understand and specify the context of use:  
- Identify and consider the (work) practices of platform stakeholders (service developers, service providers…) in order to achieve empirically grounded scenario descriptions  
- Consider routines, skills, experiences, values and norms of platform stakeholders for scenario content production  
(2) Specify the user and organizational requirements:  
- Based on (1), construct user/stakeholder scenarios and requirements. |
| **WP2/T2.2 Define patterns and guidelines for establishing trust** | (1) Understand and specify the context of use:  
- Consider the existing practices of pattern application by platform stakeholders (service developers, service providers…), and existing problems of pattern usage  
- Identify characteristics (values, norms, practices…) of service end users which influence their trust in composite services |
| **WP3/T3.1 Design-time tools and patterns** | (1) Understand and specify the context of use:  
- Identify and consider current usage and preferences of design-time techniques and tools by service developers, which will guide the tools and patterns development process  
(3) Produce design solutions:  
- Invite (future) service developers to participate in the development of design-time tools and patterns  
(4) Evaluate design against requirements:  
- Investigate if the developed/selected design-time techniques and tools fit with the work practices of service developers |
| **WP5/T5.1 Aniketos platform design** | (3) Produce design solutions:  
- Develop platform prototypes (user interfaces), which allow platform users (service developers and providers) to give feedback on the platform to platform developers |
| WP6/T6.1 Description of industrial case studies | (1) Understand and specify the context of use:  
- Identify and consider the (work) practices of industrial case study specific platform stakeholders (service developers, service providers...) in order to achieve empirically grounded scenario descriptions  
- Consider routines, skills, experiences, values and norms of industrial case study specific platform stakeholders for scenario content production  
(2) Specify the user and organizational requirements:  
- Based on (1), industrial case study specific requirements and scenario descriptions are written |
| WP6/T6.3 Aniketos applied to case study A: “Future telecommunication services” | (1) Understand and specify the context of use:  
- Identify (future) telecommunication service end users who will use the composite services  
(2) Specify the user and organizational requirements:  
- Identify and consider specific characteristics and needs of telecommunication service end users to ensure their acceptance of, and trust in the composite services  
(3) Produce design solutions:  
- Build the composite services (execution of the case study) based on the specifications of (1) and (2) |
| WP7/T7.3 Scenario based evaluation | (4) Evaluate design against requirements:  
- Evaluate if the identified scenarios and requirements (of T1.3) fit to the goals and current practices of platform stakeholders |
| WP7/T7.4 Validation and evaluation of the Aniketos platform | (4) Evaluate design against requirements:  
- Evaluate if the requirements modelling language users can successfully apply the modelling language and tools (of T1.2)  
- Evaluate if trust models for service composition (of T2.1) are e.g. comprehensible and useful for service developers  
- Evaluate if patterns and guidelines for establishing trust (of T2.2) are e.g. comprehensible and useful for service developers  
- Identify possible problems and challenges for service developers, when using the design-time tools and patterns (of T3.1) |
| WP7/T7.5 End user evaluation of the industrial case studies | (4) Evaluate design against requirements:  
- Evaluate if composite services (of WP6) are e.g. considered useful, trusted and accepted by service end users |
| WP8/T8.1 Development of distance learning materials | (1) Understand and specify the context of use:  
- Identify (future) types of users of the distance learning materials  
(2) Specify the user and organizational requirements:  
- Identify and consider specific characteristics, interests and needs of tutorial and webinar attendees  
Additionally: Prepare training material related to UCD methodologies, techniques, and tools, and about how to integrate UCD into system development processes |
6.2 Other related work to usability and acceptance

There exists a number of related work which applies a UCD approach e.g. in developing security engineering methods and tools. This research is interested in how to make such methods and tools consistent to the needs, wants and constraints of security engineers.

One of the initial tasks in the UCD process is to understand and specify the context of use. Several workplace studies have been conducted to investigate the work practices and tool usage of security professionals. For instance, Werlinger, Hawkey and Beznosov, 2008 [263] have focused on researching human, organizational and technological challenges for IT security practitioners. The analysis of data from semi-structured interviews with these practitioners led to the development of a framework on challenges which security practitioners have to face within their organizations, which helps to understand the needs of and difficulties for this group and which in turn could be used for identifying user and organisational requirements for future techniques and tools for security practitioners as a next step.

Based on published study results on the usage of IT security management tools, Jaferian et al., 2008 [264] compiled and categorized guidelines and recommendations for improving security tools. These design guidelines reflect the needs of security professionals and system administrators to be provided with usable interfaces, and are intended to inform the design process of IT security tools.

As the final step in the UCD process, the evaluation of the produced designs against the requirements takes place. Several studies about the evaluation of security requirements techniques have been presented so far, e.g. by Opdahl and Sindre, 2009 [265]. They stressed the importance of empirically investigating the usefulness and practical applicability of security threats elicitation techniques for specific usage situations. They conducted comparative controlled experiments for identifying the effectiveness and of two techniques for security threat identification (attack trees and misuse cases) among students of systems analysis and design. Moreover, the perceived usefulness, the perceived ease of use and the intention to use the two techniques were studied via a post-task questionnaire.
7 Background for the case studies

This section describes the consortium background useful for the three case studies that will be developed within Aniketos project. These descriptions are worthy in order to have an early understanding of what type of techniques and tools the Aniketos platform should collaborate with.

7.1 Case study A: “Future telecommunication services”

7.1.1 Italtel VoIP Suite (ITALTEL)

<table>
<thead>
<tr>
<th>Name</th>
<th>Italtel VoIP Suite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Technique</td>
</tr>
<tr>
<td>Purpose</td>
<td>SIP-based communication</td>
</tr>
<tr>
<td>Support area</td>
<td>RS-*</td>
</tr>
<tr>
<td>Developed by</td>
<td>Italtel S.p.A. (ITALTEL) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Prototype</td>
</tr>
<tr>
<td>References</td>
<td>WEIRD project [245]</td>
</tr>
<tr>
<td></td>
<td>COMESI project [266]</td>
</tr>
<tr>
<td>System requirements</td>
<td>Java, Linux Red Hat</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Proprietary</td>
</tr>
</tbody>
</table>

1. Short description

Italtel’s VoIP Suite is a software framework that enables the delivery of voice over IP (VoIP) services and communication on top of SIP/IP core networks.

It is basically composed of a SIP Server, which supports IETF Session Initiation Protocol (SIP) for session management, and a Presence Server for storage, management and distribution of presence information, according to the Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) specifications by OMA [268].

2. Making Italtel VoIP Suite part of the Aniketos platform

The presented framework has major relevance with the telecommunication industrial case study which is going to be developed in WP6.

A potential improvement in Aniketos can be in the realm of identity management, which may be associated with “attributes” like presence (and location). Identity, presence and location are characteristics that lie at the core of some of the most critical emerging technologies in the market today: real-time communications (including VoIP, instant messaging and mobile communications), collaboration and identity-based security.

Authentication and authorization mechanisms generally focus on determining the “who” aspect of identity. But knowing “where” (location) and “what” (presence) enables more sophisticated Authentication/Authorization methods, like consistency checking, selective access, task-based pre-authentication and proximity (de-)authentication.

Tying identity to the concept of presence in Aniketos will enable the development of next generation identity management services. All these topics need further investigation, which will be provided by the project studies on scenarios (WP1) and on platform requirements (WP5).
3. Development roadmap

The first version of the VoIP Suite has been implemented in the EU FP6 project WEIRD [245], for the demonstration of SIP-based VoIP applications over WiMAX infrastructure. The SIP Server has been complemented by a SIP client for multimedia communication: both are written in Java. Several COTS clients are compatible with this framework, like X-lite from Counterpath [267], which in addition provides support for presence.

The suite has been recently completed by the addition of the Presence Server part, based on a Java application server. This is being deployed in another research project, the Eureka/CELTIC COMESI, which deals with converged IP communication services based on OMA standards.

Some developments will be carried on within Aniketos platform design (WP5) and case studies (WP6).

To this purpose, it will be also considered the possibility to design some selected features of the suite, which could be part of Aniketos - either platform or application scenarios - by using an open-source communication engine, OpenSIPS [269], written in C language, as an alternative approach for reducing complexity and cost.

7.1.2 WindChester (WIND)

<table>
<thead>
<tr>
<th>Name</th>
<th>WindChester</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool (Integrated Services)</td>
</tr>
<tr>
<td>Purpose</td>
<td>Network Security enforcement option for Wind connection services (NetRide and WindConnect)</td>
</tr>
<tr>
<td>Support area</td>
<td>DS - Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS - Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>RS - Trustworthiness monitoring and evaluation</td>
</tr>
<tr>
<td></td>
<td>RS - Runtime validation of secure service behaviour</td>
</tr>
<tr>
<td></td>
<td>CS - End user trust and assurance</td>
</tr>
<tr>
<td></td>
<td>CS - Threat analysis and notification</td>
</tr>
<tr>
<td></td>
<td>CS - Aniketos marketplace</td>
</tr>
<tr>
<td>Developed by</td>
<td>WIND [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature.</td>
</tr>
<tr>
<td>References</td>
<td>None</td>
</tr>
<tr>
<td>System requirements</td>
<td>Windows OS, Linux, UNIX</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Commercial. Based on several 3rd party proprietary SW and HW as follows: CheckPoint, Cisco ASA, Fortinet (UTM), StoneSoft, StoneGate, Trend Micro, Fortinet-Fortimail, WebSense, Tipping Point, RSA envision, Fortinet-FortiAnalyzer, EQNetwork, Insightix</td>
</tr>
</tbody>
</table>

1. Short description

*WindChester* is a security service product developed and managed by Wind.

*WindChester* is an option offered as part of the Wind connection services NetRide or WindConnect. This is part of the offer which comprises supply/management of network security HW equipment and dedicated SW.
Wind by including in its Business service portfolio for Managed Security becomes the ideal partner for connectivity and security at the same time.

**WindChester**, the service for Managed Security, is delivered through a security management/control center called Security Operation Center (SOC). The SOC is connected over the Internet to the various security devices and apparatus located at the customer premises.

The Security Operation Center (SOC) is the collection of technical resources, standards, procedures and technological platforms (software and hardware) dedicated to the management and monitoring of the security aspects which are offered as services by Wind.

The SOC manages and controls the whole security flow of the customers by means of the management, control and analysis of the various customer apparatus in order to detect areas or systems that are more vulnerable to security threats.

### 2. Making WindChester part of the Aniketos platform

Being an integrated architecture composed of various software/hardware as supplied by different manufacturers and encompassing a broad range of security aspects it may be incorporated into Aniketos where needed.

### 3. Development roadmap

Future evolution would follow the upgradings of the various technologies and software/hardware that compose the WindChester architecture.

### 7.2 Case study B: “The emerging European Air Traffic Management systems”

#### 7.2.1 Operational Validation (DBL)

<table>
<thead>
<tr>
<th>Name</th>
<th>Operational Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Validation Methodology</td>
</tr>
<tr>
<td>Purpose</td>
<td>Validation is the Process needed to demonstrate how a system or a process can function in real life conditions with the required level of performances, security and operability. This involves checking that technological feasibility and target security level, cost-efficiency, end-users acceptability are all achieved.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS -- Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS – Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS -- Composite service analysis and preparation</td>
</tr>
<tr>
<td></td>
<td>CS -- End user trust and assurance</td>
</tr>
</tbody>
</table>

**Developed by**

Deep Blue (DBL) [Aniketos] applies the European Operational Concept Validation Methodology (E-OCVM), created by EUROCONTROL in last decades. The E-OCVM methodology was updated and further developed by many European projects such as MAEVE, CAATS I and CAATS II to which Deep Blue participated actively.

**Maturity**

High maturity - Deep Blue has applied and tailored validation principles underpinning the E_OCVM in many R&D projects in Air Traffic Management and tailored by Deep Blue for various domains, such as ICT and Healthcare. Most recently, Deep Blue applied its Validation Methodology and Process in two main security-related European ICT projects: SERENITY [19] and
SecureChange [105].

References
EUROCONTROL, European-Operational Concept Validation Methodology [270]
Lanzi et al., 2003 [271]
Pasquini et al., 2007 [272]
Leone et al., 2007 [273]
Felici et al., 2009 [274]
Felici, Meduri, et al., 2009 [275]

System requirements N./A.
Miscellaneous N./A.

1. Short description
Validation is a generic term has wide usage but with a diversity of interpretations. Deep Blue applies the Operational Validation, which encompasses aspects of technical validation with work practice evaluation.

Deep Blue’s validation methodology articulates an iterative and interactive process to answer the question: “Are we building the right system?” It supports the definition of a model enhancing stakeholders’ confidence by providing evidence of the fitness for purpose of the proposed concepts and their integration. The overarching aim is to build a sound body of evidences showing that the product satisfies stakeholders’, and in particular users’, needs.

The specific validation methodology adopted by Deep Blue is based on the European Concept Validation Methodology (E-OCVM) [270]. The E-OCVM has emerged over more than 10 years from experiences gained in many European co-operative ATM (Air Traffic Management) R&D activities, involving R&D institutes, ATM service providers and industry mostly in the framework of EC and EUROCONTROL funded projects. The E-OCVM validation methodology is of particular interest as it defines different validation objectives, activities and techniques, depending on the maturity level of concept/technology.

The main Steps in our Validation Methodology are:
- Set the evaluation strategy
- Identify the end users of the outcome
- Identify (with the users) its usage and its purpose
- Identify the high level objectives of the validation
- Determine the trial needs:
  - Identify (with the users) what criteria are to be used
  - Decompose them iteratively, in order to obtain measurable indicators
- Decide how they will be measured and analysed
- Set out a plan of how the trial will be conducted
- Conduct the trial:
  - Go through different methods (expert evaluation, activity analysis and direct observations, ethnographic studies, intensive testing, prototyping sessions, simulations, WoZ Simulations, user interviews and questionnaires)
- Determine the results:
- Qualitative and quantitative analysis of the measurements taken.
• Indicators will then require different types of evaluations. Different types of methods can be used to support the Validation:
  • Deterministic: e.g., formal proof of compliance to a specification, demonstration of S&D requirements, etc.
  • Probabilistic: e.g., quantitative statistical reasoning to establish a numerical level
  • Qualitative: e.g., compliance with rules that have an indirect link to the desired criteria (e.g., compliance with standards, staff skills and experience, user evaluation).

2. Making Operational Validation part of the Aniketos platform

Deep Blue, by means of its involvement in the different Aniketos tasks, will support the integration of analytical tools and methodologies that have been applied in other relevant ICT as well as domain-specific projects.

Deep Blue applied its Validation Methodology and Process in two main security-related projects: SERENITY [19] and SecureChange [105]. In particular, Deep Blue has gained experience in the validation of operational concepts as well as Security and Dependability features. Deep Blue has developed and tailored various analytical tools such as questionnaires and scenarios, which have been used in various validation phases, from requirements gathering and validation to scenario-driven validation exercises. These analytical tools support also the gathering and integration of domain-specific expertise. Deep Blue has run various validation workshops with ATM experts. Similar analytical tools would be a means to integrate Deep Blue’s validation expertise into project artefacts and deliverables giving rise to the Aniketos platform.

3. Development roadmap

Deep Blue is concerned with developing further experience in deploying its validation expertise, and relevant analytical tools and methodologies, by means of the Aniketos tasks. We foresee specific relevancies between the technical tasks and specific validation aspects, analytical tools and methodologies. In particular, there are various developmental opportunities for the following tasks we are involved in:

Task 1.2 Socio-technical security modelling language and tool. The main task’s outcomes will take into account “organisational and social notions, such as role, goal, task, trust, risk, dependency, policy”, and so on. We have already contributed to the validation of organisational and social notions in other projects. In particular, within the SERENITY project, we have validated the concepts of Organisational Security and Dependability Patterns. Moreover, we have experienced the viability of novel concepts by means of validation and iterative prototyping. The expected activities will extend our experience in gathering requirements for novel concepts and artefacts. On the other hand, our previous experience will contribute directly to the task objectives.

Task 1.3 Aniketos platform requirements and scenarios. We have experienced validation activities during different developmental phases. Of particular relevance it is our experience in gathering requirements, and building scenarios in order to validate them. We have actively participated in shaping and validating the requirements for the concepts underpinning the SERENITY methodology and its service-oriented platform. We see particular useful the involvement of “end-users of the composite service”. We have a strong experience in running workshops with end-users from the ATM domain (e.g., Air Traffic Controllers, IT Managers, Safety Experts, etc.).

Task 6.1 Description of Industrial Case Studies. Our access to domain experts will support the gathering of domain-specific requirements for the ATM case study. This is useful in order to identify those requirements that will make the Aniketos platform acceptable within specific application domains. On the other hand, it will be an opportunity to investigate how novel concepts, methodologies and tools like the Aniketos ones are perceived in industry.

Task 6.2 Establish a work plan for the case study realisations. We have found quite useful, within the SecureChange project, the adoption of validation principles in order to monitor the progress of project outcomes. The case studies provide a means to progressively achieve project results. In particular,
monitoring how case studies interact with the other activities is critical to “analyse how their maturity level will progress with the Aniketos timeframe.” This will allow us to strengthen our experience in using validation principles within ICT projects.

**Task 7.3 Scenario based evaluation.** We have extensive experience in using scenarios in order to validate operational concepts as well as relevant requirements and technologies. In the Aniketos context, we find quite challenging and innovative the use of scenarios in order to validate service compositions and relevant Security and Trust features. These aspects are becoming more and more critical within the ATM domain with the deployment of the infrastructure modernisation programme envisioned by the SESAR project [276].

**Task 7.4 Validation and evaluation of the Aniketos platform.** Our experience of validating operational concepts and newly implemented technologies will contribute to the task objectives. Of particular relevance, it is our experience to run semi-structured interviews and to tailor questionnaires that support qualitative as well as quantitative evaluations. Qualitative evaluations and empirical results drawn from scenario-based evaluations are cross-examined in order to highlight the operational impact (e.g., the effect on work practices) of new concepts and technologies. Most recently, we have also used questionnaires for the investigation of Trust alongside other safety operational aspects (e.g., workload, situational awareness and teamwork). We will further develop our analytical tools and methodologies by the validation and evaluation of the Aniketos platform.

**Task 7.5 End-user evaluations of the case studies.** Within the ATM domain, we have run end-user evaluations throughout the validation process. Our validation approach has been instrumental in order to run end-user evaluations by scenarios and to take into account expert judgement.

### 7.2.2 Air Traffic Management (ATM) domain expertise (DBL)

<table>
<thead>
<tr>
<th>Name</th>
<th>Air Traffic Management Expertise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Domain Knowledge</td>
</tr>
<tr>
<td>Purpose</td>
<td>Providing its operational and technical expertise in the ATM domain for the collection of requirements for the Aniketos solutions, the provisioning of feedback and recommendations during the whole tool design and development life-cycle and the validation Aniketos final results in real operational settings.</td>
</tr>
<tr>
<td>Support area</td>
<td>DS -- Trustworthiness definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS -- Security property definition and evaluation</td>
</tr>
<tr>
<td></td>
<td>DS -- Composite service analysis and preparation</td>
</tr>
<tr>
<td></td>
<td>CS -- End user trust and assurance</td>
</tr>
<tr>
<td>Developed by</td>
<td>Deep Blue (DBL) [Aniketos] gathered its domain expertise in more than 10 years of R&amp;D and consultancy in the ATM domain both for various Air Navigation Service Providers all over Europe (ENAV, DSNA, NAV Portugal, etc.), National Regulators (ENAC) and European Agencies (EUROCONTROL, EASA) and in R&amp;D projects (FP7 and TEN-T) and Initiatives (SESAR Programme [276]).</td>
</tr>
<tr>
<td>Maturity</td>
<td>High maturity. Deep Blue’s expertise is concerned with the entire system life cycle, from concepts generation and validation, requirements gathering, to validation of operational aspects and deployments.</td>
</tr>
<tr>
<td>References</td>
<td>SESAR Programme [276]</td>
</tr>
</tbody>
</table>
1. Short description

Deep Blue is actively involved in the SESAR Programme as ENAV consultant for Human Factors, Safety and Validation aspects. Deep Blue is participating in many Validation Activities funded by EUROCONTROL (e.g., TMA2010+ and Fasti South) for the introduction of innovative tools and procedures in the ATM domain. Moreover, Deep Blue is participating to the SWIM-SUIT Project User Group.

We have developed and used our ATM expertise by different types of R&D projects supported by Air Traffic Service Providers, EUROCONTROL, and the EU Transport and ICT Programmes. Our in-house analytical tools and methodologies are a means to gather, analyse and enhance our ATM Expertise. Of particular relevant for the Aniketos project is how our ATM expertise has support various R&D activities, e.g.:

- **Gathering domain-specific requirements.** We provide and involve various ATM experts with different roles within relevant organizations. For such type of activity, our team of safety and human factor experts adopt, e.g., semi-structure interviews, questionnaires and observational studies.

- **Validating ICTs by scenarios.** The validation exercises benefit of our experience of building validation scenarios drawn from ATM operational experiences and datasets. We have been running different validation activities tailored for specific outcomes, such us, requirements validation, system compliance and key performance indicators (e.g., safety) assessment. Most recently, we have also integrated the concept of trust alongside other key performance indicators in order to enhance our understanding of subtle complex dependencies among different operational objectives.

- **Supporting risk analysis.** We have supported risk analysis by the integration of Expert Judgement in the identification and evaluation of hazards, their frequencies and severities concerning change requirements drawn from the SESAR programme, and in the discussion of mitigation strategies. In particular, we have tailored questionnaires or Safety Culture and Evolutionary Risk Analysis in order to understand how such aspects interact with risk analysis models and methodologies.

2. Making ATM Expertise part of the Aniketos platform

Deep Blue will involve Air Traffic Controllers, Security and Safety Experts, Decision Makers in ATM to evaluate and validate the Aniketos solutions through their whole development process. It will use a set of tailored analytical methodologies and tools in order to support the acquisition ATM domain knowledge and expert judgment for the Aniketos deliverables.

3. Development roadmap

Task 1.3 Aniketos platform requirements and scenarios. The access to ATM domain knowledge and expert judgement is useful in order to gather specific requirements related to the ATM case study and to depict sensible scenarios. On the other hand, we are particularly interested in gathering requirements and building scenarios due to the service-oriented nature of the Aniketos platform and its emphasis on Security and Trust. The SESAR programme is a major shift in the way Air Traffic Management Services are organised and deployed across Europe. This is a unique opportunity to assess the feasibility and acceptability of service-oriented concepts usually associated with common ICTs, rather than application domains characterised by stringent critical requirements (e.g. safety).

Task 6.1 Description of industrial case studies; Task 6.2 Establish a work plan for the case study realisations. Our domain specific expertise and collaborations are relevant for the description of the ATM case study. Specific organizational and cultural aspects distinguish the ATM domain from other ones. Tailoring our case studies and enriching them will be critical in order to assess Aniketos against...
industry acceptability criteria. The case studies will be critical in order to assess the maturity and applicability of the Aniketos platform.

Task 7.3 Scenario based evaluation; Task 7.4 Validation and evaluation of the Aniketos platform; Task 7.5 End user evaluation of the industrial case studies. The validation and end user evaluation of the Aniketos platform will benefit of our ATM expertise and collaborations with end users. We have gained extensive experience in running scenario based validations and evaluation with end users. For us, the opportunity is to tailor our analytical methodologies and tools to service-oriented concepts and tools like the ones envisioned by the Aniketos platform.

7.3 Case study C: “Land-buying and eGovernance”

7.3.1 Poleod (DAEM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Poleod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Back-office Application</td>
</tr>
<tr>
<td>Purpose</td>
<td>An application covering the management needs of the Department of Urban Planning of the Municipality of Athens</td>
</tr>
<tr>
<td>Support area</td>
<td>RS - Runtime validation of secure service behaviour</td>
</tr>
<tr>
<td></td>
<td>RS - Composite service adaptation and recomposition</td>
</tr>
<tr>
<td>Developed by</td>
<td>Daem S.A.(DAEM) [Aniketos]</td>
</tr>
<tr>
<td>Maturity</td>
<td>Mature.</td>
</tr>
<tr>
<td>References</td>
<td>None</td>
</tr>
<tr>
<td>System requirements</td>
<td>Oracle Database, Clients with Windows Operating System</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>In-house</td>
</tr>
</tbody>
</table>

1. Short description

Poleod was developed at first to replace the need of a manually maintained register book for the Department of Urban Planning of the Municipality of Athens. Since then it has evolved into an application capable of tracking the progress of the citizens' requests towards the department as well as providing various information concerning Urban planning such as building permits.

2. Making Poleod part of the Aniketos platform

Poleod is a program intended as a back-office application only. As such it has no interfaces available to the public, either human or machine interfaces. Even so, it contains data from its many years of usage, based on the needs of the Department of Urban Planning. This pool of data is accessible to services developed on the Aniketos Platform. Then, these services can be tested and verified against this dataset.

3. Development roadmap

There are two main issues that need to be addressed in order Poleod to be part of Aniketos. The first one is its client-server architecture with the business logic exclusively on the client part. For any integration with another system to be attempted, some of this logic should be either duplicated or migrated to the server part, so as to be reusable. Then other interfaces must be developed in order to access the information stored in Poleod.

The second issue is that Poleod is not internationalized. It is dependent on a non-Unicode character set and its current interface is only in Greek. This should be addressed for the application to be part of a multi-national effort.
7.3.2 **OPS (Integrated Information System) of Municipality of Athens (DAEM)**

<table>
<thead>
<tr>
<th>Name</th>
<th>OPS (Integrated Information System) of Municipality of Athens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>E-Government Services</td>
</tr>
<tr>
<td>Purpose</td>
<td>Providing e-government services to citizens and business through a Web Interface</td>
</tr>
</tbody>
</table>
| Support area | DS - Composite service analysis and preparation  
 | | RS - Runtime validation of secure service behaviour  
 | | RS - Composite service adaptation and recomposition  
 | | CS - Reference Architecture and Patterns  
 | | CS - End user trust and assurance |
| Developed by | Daem S.A.(DAEM) [Aniketos] |
| Maturity | Mature. |
| References | Municipality of Athens SI projects [277] |
| System requirements | Oracle Database 10g, Oracle Application Server 10.1.3, Oracle Portal, ArcGis Server |
| Miscellaneous | In-house. Licenses and application’s development are proprietary to DAEM S.A. |

1. **Short description**

OPS was created in order to integrate various back-office applications used by the administration of the Municipality of Athens and to create new services not only for the citizens of this municipality but also for business that operate within its limits. These services are provided through a web interface and include an information portal as well as the issuance of various certificates and the on-line payment of debts.

The portal provides the end user with information about both online and offline services provided by the Municipality as well as real-time maps with places and events taking place. The rest of the services are accessible through the same portal but require the use of both server and client side certificates in order to ensure encryption and authenticity.

2. **Making OPS part of the Aniketos platform**

OPS is a ready-to-use Integrated Information System for the Municipality of Athens. As such it can provide its data and services and can also be used as an end-user to the Aniketos platform. The quantity of real and reliable data means that it can be used as a stress test for the performance of the aforementioned platform.

3. **Development roadmap**

New interfaces – either web interfaces or web services – may be needed in order to provide the right environment to carry on specific tests from the perspective of the end user of the Aniketos platform.
8 Recommendations for usability and acceptance of the platform

In this section, guidance in selecting suitable techniques and tools for each module is given from a user-centred design point of view. User-centred design (UCD) is concerned with a close inspection and consideration of users’ needs, wants and limitations through the planning, design and development of a system, which should facilitate a higher user acceptance of the system. From this point of view, a technique or tool is recommended to be integrated into the Aniketos platform if it suits the needs and wants of future Aniketos platform users (i.e. security experts, security engineers, security analyzers, software developers, etc.), who are intended to deploy the offered techniques and tools.

More specifically, several recommendations for selecting and developing techniques and tools for the Aniketos platform users are provided. These recommendations are based on usability factors (e.g. effectiveness, efficiency and satisfaction) and on criteria for the successful acceptance of technology and methodologies, which have been derived from theoretical models (Unified Theory of Acceptance and Use of Technology (UTAUT) by Venkatesh et al., 2003 [279]) and their application in developer acceptance studies (e.g. Riemenschneider et al., 2002 [280]):

- **Usability of the technique or tool for the users:** It is recommended to select or develop techniques and tools, which can effectively, efficiently and satisfactorily be deployed by envisioned users. For instance, the graphical socio-technical security modelling tool should allow security experts to successfully analyse and specify services within a reasonable amount of working time.

- **Perceived ease of use of the technique or tool for the users:** Selecting or developing techniques and tools, which require an adequate amount of resources and effort when using them, will also contribute to an increased acceptance of these techniques or tools. This applies e.g. to tools for the monitoring of the trustworthiness of composite services during runtime. A security expert should find trustworthiness monitoring tools easy to use. Aside from this, applying the tool should be easy to learn for the security expert, who hasn’t known the tool before.

- **Perceived usefulness of the technique or tool by the users:** Only those techniques and tools for creating and maintaining secure and trusted composite services should be selected for the Aniketos platform modules, which will be regarded as useful by future users of the platform. Users (e.g. security experts) need to believe that the tool or technique enhances their job performance and that it is even more useful than previously used techniques and tools; otherwise the acceptance rate of the tool or technique will be low. Also the social (work) context of tool usage will affect the perceived usefulness of the technique or tool by the user, for instance when co-workers or supervisors don’t favour its usage.

- **Compatibility of the technique or tool with existing values, needs and experiences of users:** It is recommended to select or develop techniques or tools for the Aniketos platform, which are consistent with values, needs and experiences of the future users of these techniques or tools. For instance, tools for predicting the trust level of composite services should fit in with the current trust level prediction practices and tools of users, who create and maintain secure and trusted composite services. Those tools, which account for and build on existing skills and knowledge of security experts and which consider the needs of security analysts (e.g. by providing guidance and support for tool usage), are more likely to be accepted by them.

By applying a user-centred design approach, the user requirements (needs, wants and constraints of users) for the Aniketos platform tools and techniques can be identified and taken into account in the development process in order to successfully meet the described acceptance criteria.
9 Conclusion

This document compiles existing work related to the relevant research and development areas of the Aniketos project. We have consolidated the descriptions of the work following the same template and putting the Aniketos consortium internal background in relation to existing commercial products and results from other research projects.

The work presented has been structured in relation to the components in the current version of the system component model of the Aniketos platform. Components belonging to both the Aniketos platform itself and its Environment have been taken into account. Figure 3 provides an overview of the allocation of relevant work to the Aniketos components. This figure visualises the classification of the relevant existing work to Aniketos areas of work and the particular components identified in D1.2.

The document presents initial thoughts on how the techniques and tools could be made part of the platform or its environment, and the foreseen adaption and integration work. Discussions on the selection of the baseline techniques and tools are still on going and each of the technical WPs in the project will carry out detailed analysis and selection, once the platform requirements and architecture have been finally specified in D1.2 (for month 12). The final integration is ensured through the work on platform integration in WP5.

Finally, a set of recommendations to ensure the usability and user acceptance of the Aniketos platform are provided in order to guide the selection and future development of the techniques and tools within the platform.
References


D1.1: Consolidation of existing work


WSBPEL v2.0 specification by OASIS, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html


ModelPro tool, http://portal.modeldriven.org/project/ModelPro


Pepers project, http://www.pepers.org/?Page=StaticVerification


[87] Java Metadata Interface 1.0 (JMI), Sun Microsystems, Inc. (June 2002), http://java.sun.com/products/jmi


[89] Object Management Group, Interactive Objects and Project Technology, MOF Query/Views/Transformations, OMG Document ad/03-08-11, ad/03-08-12, and ad/03-08-13 (revised submission, 2003).


[94] XSL Transformations (XSLT), Version 1.0, James Clark, Editor, W3C Recommendation (November 16, 1999), http://www.w3.org/TR/xslt


[113] J. Han and K. M. Khan, “Security-Oriented Service Composition and Evolution” in the XIII Asia Pacific Software Engineering Conference, December 06-08, 2006, pp.71-78,


[137] DEPLOY project, http://www.deploy-project.eu/


D1.1: Consolidation of existing work


Facebook, http://facebook.com/

Twitter, http://twitter.com/

Yahoo, http://yahoo.com/

OAuth protocol, http://www.oauth.net/


RSA-SHA1 Signature Suite v1.0, http://www.w3.org/PICS/DSig/RSA-SHA1_1_0.html


SHIELDS EU FP6 project, http://www.shields-project.eu/


GlasFish community, http://glassfish.java.net/

PostgreSQL, http://www.postgresql.org/

ORACLE’s Java EE, http://www.oracle.com/technetwork/java/javase/overview/index.html


CERT secure coding standards, http://www.securecoding.cert.org/


Common Vulnerabilities and Exposures (CVE®) http://cve.mitre.org


The Open Source Vulnerability Database OSVDB, http://www.osvdb.org


Purdue University’s CERIAS vulnerability database, https://coopvdb.cerias.purdue.edu/

Open Vulnerability and Assessment Language OVAL, http://oval.mitre.org/


WSO2 Developer portal for SOA, http://wso2.org/

WSO2 company site, http://www.wso2.com


[205] Service Oriented Architectures for All (SOA4All) EU FP7 project, http://www.soa4all.eu/


[211] SLA@SOIS EU FP7 project, http://sla-at-soi.eu/

[212] Q-ImPrESS EU FP7 project, http://www.q-impress.eu/wordpress/


| [232] | Apache Web services – Axis, [link](http://ws.apache.org/axis/) |
| [237] | Apache Synapse, [link](http://synapse.apache.org/) |
| [238] | GlassFish wiki, [link](http://wikis.sun.com/display/GlassFish/GlassFishWiki/) |
| [239] | WSO2 Web Services Framework, [link](http://wso2.com/products/web-services-framework/) |
| [241] | Microsoft’s .NET Framework, [link](http://www.microsoft.com/net/) |
122

D1.1: Consolidation of existing work

[244] EUQOS EU FP6 project, http://www.euqos.eu/
Special Issue: Model-Driven Development for Secure Information Systems, May 2009, pp. 916-932


Appendix A. More detailed descriptions of works

1 Socio-technical security modelling tool

1.1 SeaMonster (SINTEF)

1. Description

SeaMonster is a security modelling tool continuously being developed by an open source community lead by SINTEF. The main advantage of SeaMonster is that it supports notations and modelling techniques that security experts and analyzers are already familiar with, namely attack trees (Schneier, 2001) and misuse cases (Sindre & Opdahl, 2005), and links common model elements together through a centralized repository (the SHIELDS SVRS, located and described on http://www.shields-project.eu/). In this repository it is possible to upload, download and search for SeaMonster models, thereby supporting collaboration and reuse of threat models between projects, people and organizations.

SeaMonster is based on Eclipse, which is basically an application platform where a very large set of plugins can be added to suit the needs of the users. The three main Eclipse plugin frameworks SeaMonster benefits from are the Graphical Modeling Framework (GMF), the Eclipse Modeling Framework (EMF) and the Graphical Editing Framework (GEF). GMF is a framework for developing Eclipse graphical editors, and functions as a bridge between EMF and GEF.

2. Detailed description of basic operation

SeaMonster is a tool for creating threat models. The goal of threat modelling is to create early security awareness during development, and is of great aid during risk analysis and security planning. The threat models visually present aspects that represent threats to a software system or a component of a system. These should be used to prioritise what is to be protected (and thus indicate security goals) and identify relevant attacks that exploit commonly found vulnerabilities for this type of system. Additionally, the models can show alternative measures to mitigate the threats. The modelling formalisms we use for threat modelling are misuse cases and attack trees.

Creating a misuse case diagram for an application, component or service can be performed through an activity\(^1\) where you either make from scratch or reuse an existing model. The latter requires less security expertise, but depends on a previously created model from a similar project. A misuse case model describes functionality as activities performed by actors, vulnerabilities as something that can occur related to functionality, threats as something that can exploit functionality or vulnerability, and finally security use cases as something that mitigates threats or vulnerabilities, see Figure 11.

\(^1\) See the SHIELDS Approach guide, http://www.shields-project.eu/files/docs/D1.4%20Final%20SHIELDS%20Approach%20Guide.pdf
Threat, vulnerability and security activity elements can with SeaMonster be retrieved from the SHIELDS SVRS, where they are interlinked through a common domain model. This enables reuse of the same model elements in many models, and if a new type of threat is identified for a vulnerability class, all models using making use of this can be discovered and subsequently notified. This threat can also be connected to suggested mitigations, which then can be inserted into the model.

Attack trees, as shown in Figure 12 are used to complement the misuse cases by providing more detailed information on how a threat can be realised from the attacker’s point of view. The root node of the tree is the overall goal of what the attacker wants to achieve. Sub-goals in the tree must be fulfilled in order to achieve the overall goal, and have either a disjunctive or conjunctive relationship between siblings. SeaMonster uses the SHIELDS SVRS in a similar manner here, by retrieving reusable threats as sub-goals. By creating an attack tree for a service, the attack tree defines which attacks may have negative consequences for it.
3. An example

The model shown in Figure 13 has been created for a service module that manages orders and shipment of goods in a warehouse. It involves 4 legitimate actors, 3 illegitimate actors, a number of business level functionality, a number of threats to the system, vulnerability classes that the system must avoid, and possible mitigating security activities. More details on how to implement an attack, e.g. Change RFID information, was modelled as attack trees (not shown here). If this service module becomes exposed to some of these threats, e.g. that someone starts sending a lot of false orders, then this would affect the whole system and appropriate actions must be taken. All the threats, vulnerabilities and security activities are linked to and can be found in the SHIELDS SVRS, where additional metadata, such as descriptions, references and related elements can be found as well.

![Figure 13. The Warehouse data management example service.](image)

4. Additional references


1.2 TECNALIA Security Framework (TECNALIA)

1. Description

TECNALIA’s Security Framework is an Eclipse\(^2\) Based tool for the creation and automatic deployment of Security Policies. In fact, the tool is an Eclipse Rich Platform (RCP) application, built upon the RCP framework of Eclipse. The objective of this tool is two folded: (i) to provide a graphical environment for the modelling of WS-SecurityPolicy compliant policies and (ii) to provide a runtime environment to test the designed policies.

This way, after the policy has been modelled through the editor, it can be easily attached to a Web Service, and deployed into an Apache Axis2 application server.

2. Detailed description of basic operation

The specification exposes that WS-Policy defines a framework for allowing web services to express their constraints and requirements. Such constraints and requirements are expressed as policy assertions. WS-SecurityPolicy defines a set of security policy assertions for use with the WS-Policy framework with respect to security features provided in WSS: SOAP Message Security (WSS10, WSS11), WS-Trust and WS-SecureConversation. The intent is to provide enough information for compatibility and interoperability to be determined by web service participants along with all information necessary to actually enable a participant to engage in a secure exchange of messages.

The following diagram shows the actors and the relationships that may be typically involved in a network security scenario:

![Diagram](image)

**Figure 14. WS-SecurityPolicy typical scenario**

The diagram shows the possible interactions that may occur between actors in any given scenario, although, in general, depending on the policy specified by the recipient, only a subset of the possible interactions will actually occur in a given scenario. Note that the Issuing and Validating Authorities, may, in general be either a WS-Trust Security Token Service (STS) or other authority.

The next figure below explains the high level design of the tool:

---

The TECNALIA Security Framework tool is designed as a component oriented application. The core component defines the WS-SecurityPolicy metamodel, with the knowledge of the OASIS specification. Here, we must note that every implementation of the specification has its own characteristics, so the core component needs to be extended with another metamodel, introducing this domain specific knowledge. The design allows the creation of extensions in an easy process. In the current implementation, the developed extension is for the Apache Axis2 application server.

As mentioned above, the security policy is modelled through a graphical user interface provided by the tool. As the policy is being modelled, attributes and options are available to the user so that he is able to design and configure it. Afterwards, the model can be exported to an Axis2-compliant configuration file, by the introduction of some domain specific parameters, such as the key store type, paths, and passwords. The Apache Axis2 enforces the fulfilment of the policy in the IT infrastructure where it is deployed. Actually, the tool provides a way to deploy Web Services which could be secured by the created policies.

The way to proceed with TECNALIA Security Framework is described below:

- The user (usually the company’s IT Policy/Security responsible) models the policy with the User Interface. The attributes the user can put into a policy depend on the type of model created, this is, thanks to the model checking, only the possible attributes are shown to the user, making impossible to introduce structural errors into the model. The resultant of this process is a policy aligned with the WS-SecurityPolicy standard.

- The last but one point in the process is the extension of the defined model with the domain specific information, needed by the application server to correctly enforce the policy. This phase is semi-automatic, because few parameters are required before the extension is performed. These parameters can not be inferred by the tool, because are completely coupled to each specific infrastructure (e.g. key store names, passwords, etc…). When these parameters are introduced, another instance of the metamodel will be automatically created integrating all this new information.

- Finally, the user is able to use the export function in order to generate the policy file which contains the modelled policy. At this point, this policy file is able to be linked to a Web Service in order to introduce the security requirements’ the policy represent.

The figure below shows graphically the process described:
3. An example

The picture below shows an example of a policy for a small scenario where:

There is a Web Service providing certain business process, without any security requirement enabled. Then, the enterprise wants to introduce security requirements in order to make the service more robust, as they are going to manage critical information. The security requirements provided are mutual authentication, confidentiality and integrity of the exchanged information.

These security requirements will be implemented as follows:

- **Mutual authentication**: with tokens including a mutual client/server authentication where the server authenticates the user but the user performs the same authentication to the server. In the example, this process is done by the use of X.509 certificates.

- **Confidentiality**: all the exchanged SOAP messages will be encrypted (actually, the body of the messages).

- **Data integrity**: all the exchanged SOAP messages will be signed (actually, the body of the messages).

Using the TECNALIA Security Framework tool, the resultant model is the following:

![Figure 17. Example of a security policy edition (i)](image-url)
to introduce. A pop-up window will allow the user easily type this information. The next figure shows this pop-up:

![Figure 18. Example of a security policy edition (ii)](image)

Now, the model (extended model in the terminology) has all the necessary data to be attached to the Web Service. In the next picture, the resulting policy is showed:

![Figure 19. Example of a security policy edition (iii)](image)

Finally, using the Web Service creation wizard, user can attach the policy to the service, by pointing the file that has just created. This process is showed below:
1.3 Modelling Evolutionary Requirements (UNITN)

1. Description
The methodology for evolutionary requirements is developed to model and analyze changes in security requirements. This methodology addresses the problem of accommodating a change in the security requirements and guaranteeing the correctness and the fitness to purpose of such a change. To this end, the methodology proposes a model-driven approach to represent security requirements and eventual changes on them. It also provides an automated reasoning to manage changes and the argumentation about them. The ultimate purpose of this methodology is to guarantee that a change fits to its purpose. The methodology consists of:

- A metamodel that incorporates requirements, security requirements, and evolution of security requirements.
- Argumentation model and automated support to systematically and iteratively manage argumentation between different stakeholders about the satisfaction of security requirement stated in a requirement model.
- Specifying rules for evolutionary changes based on model transformation.

The requirements metamodel builds on two mainstream approaches in requirements modelling; the Tropos methodology, and the Problem frames approach. Both models are problem driven and share the basic ideas that software is a means to reach user requirements in specific environment. The model takes benefit from extensions of the two requirements models to accommodate security; namely Secure Tropos, and Abuse Frames, to model security requirements. Moreover, the model includes an explicit notion of the world in which the system operates. The evolution in security requirements happens as response to change. The change is, basically, a transition from one situation (before) to another (after). The work gives a metamodel for the change in a situation. The primitive changes are the addition, modification and removal of an element (entity or a relationship). On the basis of these primitive changes, a composite change can be constructed.
2. Detailed description of basic operation:
The envisioned workflow of maintaining design and requirement models through evolution requires the continuous adaptation of the models to react to changes. The adaptation is iterative. Thus, applying an evolutionary action represents itself a situation change which may trigger another evolutionary rules or required manual intervention.

- Situation change: changes of situation are initially caused by external factors (environment context) of the system.
- Evolution rule(s) triggered: situation changes can trigger evolution rules that perform automatic adaptation (evolutionary action).
- Manual intervention: if none of the rules was automatically triggered, then analysts intervention is required to specify what and if an evolutionary action to be taken.
- Loop exit: if none of the evolutionary rules was triggered or the analysts stated that no evolutionary action is to be taken.

These steps are summarized in the following figure taken from D3.2 of the SecureChange project (http://www.securechange.eu/content/deliverables)

![Figure 21. A continuous process to maintain the requirements](image)

3. An example
The modelling language, developed by UniTN and partners as a part of SecureChange project, makes it possible to model requirements, security requirements and eventual change in security requirements. To give a simple example about changes in security requirement within a service-oriented computing paradigm, let’s consider an agency of trip arrangements (tripOrganizer). The agency needs to contact with hotels, airlines, car renting and other agencies within an SOA and acts on behalf of an end-user. Suppose that due to some new rules/laws, the agency needs to authenticate the identity of an online end-user (traveller) before making the online reservation. This authentication can be done through a governmental organization. The new service offered by a governmental organization to authenticate data will interact with tripOrganizer and access its data to verify if the personal data given by a user X is correct and if it matches with a real person. The addition of this new service necessitates introduces a change in the security requirements and requires re-analysing the whole system to detect potential security vulnerabilities. The first step towards this analysis is the modelling of requirements and the changes occurring from time to time in a way that facilitate the reasoning.
1.4 Goal Risk Model (UNITN)

1. Detailed description of basic operation

The Goal Risk Model is a modelling and reasoning framework that considers risk during the early phase of the system development. It supports modelling of uncertain events, mainly risks, that can influence the fulfilment of one or more goals, managing the effect of risks as well. The framework is used to analyze risk at the organizational level, offering a number of techniques which help the analysts identify and enumerate relevant countermeasures to mitigate the evidenced risks. Therefore it comprises two basic steps: risk analysis and risk mitigation.

The analysis starts identifying a relevant event or an uncertain circumstance. An event can influence several goals, it could be a risk for some goals, while it could be an opportunity for others.

The analysis of the model takes place after the model is fully developed, eliciting the most appropriate solution that will satisfy stakeholders’ goals and certain preferences they might have. The steps of the analysis are:

- Find alternative solutions
- Evaluate alternative solutions against relevant risks
- Assess countermeasures to mitigate risks.

Once the events are analyzed, countermeasures needed to mitigate the risks are identified and analyzed. The framework elicits a solution that is not only based on the stakeholder’s goals, but also anticipates the existence of malicious events by offering several treatments to manage their impacts. Treatments include prevention, detection, alleviation, etc. The analyst must understand the characteristics of the model before choosing one of the measures to undertake. Therefore, the obtained solution will be more robust.

2. An example

An example of the application of GRM is the London Ambulance Services (LAS) case study presented in Asnar and Giorgini, 2006 [14].
A strategic objective of LAS is to reach the location of Accident and Emergency in time. In Tropos goal model, the goals are decomposed and refined until they become tangible (i.e., there is an actor who can fulfill the goal). In GRM, each entity has a separate layer of analysis.

Event identification:
The risk of failure in contact centre computer system is a risk that obstructs the achievement of the goal allocate ambulance properly because LAS cannot know the status of the current ambulance.

On the other hand, this event can be seen as an opportunity for the goal having reliable manual standard operating procedure (SOP) for A&E, since it can give the opportunity to exercise manual SOP.

Countermeasures:
The countermeasure maintaining ambulances regularly reduces the risk of having ambulance is broken after receiving A&E dispatch.

All alternative solutions are evaluated based on a criteria, let us suppose ‘minimal cost’. The candidate solution that has the minimal cost and the necessary countermeasures is going to be implemented. GRM helps find this final optimal (w.r.t the specified criteria) solution.
1.5 Secure Tropos (UNITN)

1. Description
The Secure Tropos methodology, based on the SI* (secure i*) modelling language, is an agent-oriented security requirements engineering methodology. Secure Tropos allows for the design of secure information systems starting since the initial development stages (early requirements). It has been conceived from the observation that a system cannot be secure if security is not taken into account since the beginning of the development process. The language it is based upon, SI*, takes a subset of the concepts of i* (actor, role, agent, goal, resource, task, softgoal) and complements it with a number of constructs to deal with security issues (objectives, entailments, capabilities, trust of execution and of permission, delegation of execution and of permission). SI* is formally represented and Secure Tropos comes with different automated reasoning techniques. The Secure Tropos methodology consists of two main phases: early requirements (before the system-to-be is introduced) and late requirements (after introduction).

2. Detailed description of basic operation
In order to explain Secure Tropos in more detail, we proceed as follows. First, we show the basic elements of its metamodel. Second, we briefly describe the methodology that explains how it is applied.

The first figure shows the basic relations between actors and “services”. There are two types of actor: role and agent. An agent can play zero or more roles; a role can be played by an arbitrary number of agents. The services in Secure Tropos are goals (e.g. “send report to boss”), tasks (e.g. “send report via e-mail with Thunderbird”), resources (e.g. “monthly report”). Actors are linked to services in three different modalities. Entitlement is represented by the “Own” relations; objectives are represented by the “Request” relation; capabilities are represented by the “Provide”.

![Figure 23. Basic elements of Secure Tropos metamodel](image)

The second figure focuses on trust and supervision relations. Secure Tropos allows to define these concepts in terms of roles. Some roles supervise others, which are subordinate ones. There are two different types of trust: trust of execution and trust of permission. Trust links two roles and a service: the trustor trusts the trustee for a trustum (a service). Trust of execution refers to the achievement of a goal, the execution of a task, the provision of a resource. Trust of permission refers to the entitlement of the trustee to deliver some service. The meta-model supports both trust and distrust, via the boolean value for attribute “condition” in classes Trust of Execution and Trust of Permission.
The third meta-model shows dependencies and delegations. The dependency relation links a role to another to represent the dependency for the achievement of a goal, execution of a task, provision of a resource. Delegation of permission refers to the passage of privileges between roles.

The chunks of the meta-model we have shown above result in Secure Tropos models. Different models are created in different phases of the Secure Tropos methodology. An overview of the methodology is shown in the following figure.

The methodology is an iterative process composed of the following conceptual phases:

- Requirements Acquisition Process, in which the above presented modelling activities are used to produce different kinds of diagrams that represent different views of the system requirements model.
- Organizational Requirements Model Instantiation, in which the requirements model of the organization is instantiated at individual level.
- Requirements Verification and Validation, in which the compliance of the model with security properties and the consistency of requirements specifications are verified.
- Security Mitigation, in which the requirements model is revised by introducing security measures to cope with the violation of security properties.

The following figure shows more details concerning Requirements Acquisition.

![Figure 27. Requirements Acquisition within Secure Tropos methodology](image)

**Actor modelling** consists of identifying and analyzing the actors of the environment, and the system’s actors and agents. In the early requirements phase, actor modelling focuses on modelling the application domain stakeholders along with their objectives, entitlements and capabilities. Agents are also described in terms of the roles they play. During late requirements phase, the actor modelling focuses on the definition of the system-to-be as an actor. This modelling activity produce the actor diagram that represents the actors involved in the system along with their objectives, entitlements, capabilities, and agent-role assignments.

**Social modelling** consists of two phases:

- **Trust modelling** consists of modelling the expectations of actors about the achievement and use of goals by other actors. Such expectations are modelled using (dis)trust of execution and (dis)trust of permission links. In early requirements analysis, the focus is on the modelling of (dis)trust relations between social actors of the organizational setting. During late requirements analysis, trust modelling focuses on analyzing the (dis)trust relations involving the system-to-be actor. This modelling activity produce the trust diagram that enriches the actor diagram by representing the trust network.

- **Supervision modelling** consists of modelling the structure of the organization in terms of supervision relations. This modelling activity produce the organization chart.

**Goal modelling** proceeds in order to enrich the model with further details. Specifically, goal modelling rests on the analysis of actor’s goals, conducted from the point of view of the actor, by using three basic reasoning techniques:

- **Goal Refinement** analyzes and refines root goals to build a finer goal structure in terms of an AND and OR decompositions.

- **Contribution analysis** studies the impact of the achievement of goals on the achievements of other goals.

- **Means-end Analysis** aims at identifying tasks that provide means for achieving a goal.

**Execution Dependency modelling** consists of identifying actors who depend on other actors for the achievement of a goal, execution of a task, and supply of a resource. Essentially, this activity identifies assignments of responsibilities between actors. Such assignments are modelled using execution dependency links. During early requirements analysis, functional dependency modelling focuses on assignments of responsibilities among social actors of the organizational setting. During late requirements analysis, the focus is on transfers of responsibilities involving the system-to-be actor. This modelling activity produce the execution dependency diagram that enriches the actor diagram by representing the execution dependency network.

**Permission Delegation modelling** consists of identifying actors who authorize other actors to achieve goals, execute tasks, and access resources. Essentially, this activity identifies transfers of authority.
between actors. Such transfers are modelled using delegation of permission links. During early requirements analysis, authorization modelling focuses on delegations of authority between social actors of the organizational setting, whereas during late requirements analysis the focus is on delegations of authority involving the system-to-be actor. This modelling activity produce the permission delegation diagram that enriches the actor diagram by representing the permission delegation network.

**Task/Resource modelling** is complementary to goal modelling. It rests on reasoning techniques analogous to those used in goal modelling, namely, means-end analysis, contribution analysis, and task refinement.

- **Task Refinement** analyzes and refines root task to build a finer task structure in terms of AND and OR decompositions.
- **Contribution analysis** studies the impact of tasks on the achievements of goals and execution of other tasks.
- **Means-end Analysis** aims at identifying resources that are consumed or produced by tasks.

An important step in Secure Tropos is the definition of Security properties that can be checked against existing models. A list of basic properties is shown in the figure below.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pro1</td>
<td>capture untrusted delegation. An actor delegates a service to another, but the delegatee does not have permission for that service.</td>
</tr>
<tr>
<td>Pro9</td>
<td>capture the need to know principle. An actor has permission for a service even though he needs no such permission.</td>
</tr>
<tr>
<td>Pro10</td>
<td>capture trust conflicts for permission, where two permission chains go from an actor to another for the same service, each chain having different modality: trust vs. distrust.</td>
</tr>
</tbody>
</table>

**3. An example**

Suppose a customer C depends on a bank B for an online banking service. Also suppose the bank, due to the overloading of its servers, delegates the provision of the online banking service to an external hosting company H. C trusts B for the provision of online banking, but does not trust H for the same service.

Using the Secure Tropos methodology, a system designer can (i) represent such setting using the concepts of agent, goal, delegation of execution, trust of execution; (ii) identify an untrusted delegation using automated reasoning techniques (e.g. verifying the security properties described above). See the description of SI* tool for more details.
The following figure shows a different scenario graphically modelled using Secure Tropos. The scenario concerns handling of private data of a patient. Bob, the patient, delegates the permission of managing his data to the Health Care Center (HCC). Bob owns such goal. Bob delegates the execution of goal “provide medical services” HCC. Bob requests such goal. In order to manage the patient data, the HCC decomposes such goal to subgoals provide patient data and collect patient data. The scenario shows a number of relations between actors involved: the Medical Emergency Rescue Center (who handles patient requests), a doctor named Charlie, a social worker Allison in charge of delivering medicine.
1.6 Si* tool (UNITN)

1. Description

Si* Tool is a graphical tool where it is possible to draw Secure Tropos models and to perform the
effective formal analysis of Secure Tropos specifications. The tool is provide as an Eclipse's plugin
and use XML as its document format. Formal analysis is based on logic programming. Si* Tool allows
to different systems based on Datalog to analyse Secure Tropos specification.

The main objectives behind the design of the tool are:

- Graphical environment: a framework to assist requirements engineers in the creation of graphical
  models of early security requirements using Secure Tropos concepts.
- Formalization: support for translating graphical models into formal specifications.
- Analysis facilities: a front-end to state-of-the-art, off-the-shelf ASP solvers that support the
  analysis of these specifications to ensure that desirable security properties are satisfied.

Though originally designed for Secure Tropos, the Si* Tool supports other dialects of the i*/Tropos
requirements engineering notation. In particular, it supports (i) modelling and reasoning about risks,
via the goal-risk framework; (ii) organizational planning, where an organization is composed of a set
of actors and a set of goals these actors have to achieve, via their own capabilities and via interaction
(delegations); (iii) defining security patterns (solutions to recurrent security issues) and applying these
patterns to a certain setting.

2. Detailed description of basic operation

The Si* Tool is an Eclipse-based tool. As a consequence, the way to use it is in line with the usage of
typical Eclipse tools.

Create a Serenity project, which is a container that allows for modelling several settings;

Create a diagram. The current version of the Si* tool supports four types of diagram: Si*, goal-risk
(GR) framework, organizational planning, and security patterns.

For Si* and GR models, the designer can draw the models using the palette;

For organizational planning, the input has to be provided through a set of tables;

For Security patterns, two subdiagrams have to be defined: the context and the solution. These
diagrams represent the generic setting where the pattern can be applied, and the modification that are
effected when applying the pattern, respectively;

Run analysis / Apply the pattern. After the model has been created, the tool can perform automated
reasoning techniques. The outcome of these techniques depends on the model that has been created
Si* models can be analysed to verify whether some security property holds or not. For example, one
might look for untrusted delegation of execution relations, where one actor delegates some activity to
some other actor he does not trust;

Goal risk models can be analysed to check the risk level in a certain setting, on the basis of the risks
and countermeasures in the setting;

For organizational planning, the tool allows for running the planner in order to find solutions that
ensure organizational goals are achieved by performing actions by the various actors;

Security patterns can be applied to a scenario. Applying one pattern means (i) checking whether, in the
scenario, the pattern can be applied – the context is identified; and (ii) changing relations in the
scenario, as prescribed by the pattern, so that the security problem is solved.
3. An example
We demonstrate how the automated reasoning concerning Si* diagrams works. Suppose a Serenity Project has been created, and the following Si* diagram has been drawn, where agent7 delegates the execution of goal9 to agent8.

![Figure 30. Goal execution delegation diagram](image)

Now, we want to verify whether this diagram violates any security property. Therefore, we right click on the diagram file, and we select “Serenity → Run ASP Analysis”. After that, the Si* tool calls an external solver (based on disjunctive datalog) and the results are shown in the diagram as follows:

![Figure 31. Goal execution delegation result diagram](image)

The analysis has detected an untrusted delegation of execution, for agent7 doesn't trust agent8 for the achievement of goal9. Now, the designer can update the organizational model by adding a trust of execution relation or making agent7 rely on another (trusted) agent.

Examples on the usage of the other functionalities can be seen in the Si* tool user guide: [http://sesa.dit.unitn.it/sistar_tool/request.php?33](http://sesa.dit.unitn.it/sistar_tool/request.php?33)
1.7 Organizational S&D Patterns (UNITN)

1. Short description
Organizational S&D patterns are a specialization of the concept of S&D pattern introduced by the Serenity project. A pattern is a solution to a common problem; design patterns are probably the most popular example of pattern. S&D patterns are a kind of patterns that deals with security and dependability. Organizational S&D patterns talk of S&D at the organizational level, and describe how an organization should change in order to cope with a specific S&D issue.

An organizational S&D pattern is composed of different fields:

A context where the pattern applies. The context is expressed as an Si* organizational structure, namely a set of actors with their own goals and connected by delegation and trust relations.

A requirement that expresses, in natural text, the kind of S&D property that the pattern provides.

A solution: the set of changes that should be applied to the organizational structure to actuate the pattern. This might be adding or removing goals, delegations, trust relations, and so on.

A set of consequences: a textual representation of the consequences (including side effects) that applying such pattern will provide.

Organizational S&D patterns have been successfully applied to represent legal patterns as well.

2. Detailed description of basic operation
In order to use Organizational S&D patterns, a series of steps should be followed. These steps are typically supported by a tool. For example, the SI* Tool supports this technique on Secure Tropos models.

Step 1. Create the patterns
The first activity is the creation of one or more patterns. Usually, patterns need not be created for every considered scenario. Patterns should be stored in repositories so that they can be used whenever necessary.

In order to create one pattern, all its fields should be filled. Context and solution have to be expressed in terms of the same language. Using Secure Tropos a pattern context is a set of goal-oriented actors that interact via delegations. A pattern solution can be specified using two equivalent notations: (a) a Secure Tropos diagram showing the final organization after the pattern has been applied; or (b) the set of atomic operations that should be performed to apply the pattern (add actor, decompose goal, delegate goal, …)

Step 2. Draw the scenario
The organizational scenario under consideration should be modelled using the same language used to create the patterns.

Step 3. Apply the solution
Whenever a problem in the organization is identified, either by manually listing the goals that are problematic or by automatically identifying such risky situations via pattern matching techniques on the organizational scenario, the solution can be applied.

Applying the solution consists of modifying the organizational structure of the scenario under consideration as prescribed by the pattern solution. Importantly, all the operations at the organizational level should be bound to concrete actions, so that the changes applied to the model result in actual changes in the organization.

3. An example
We provide an example taken from the paper published in AI&Law. Such example represents the access control pattern. Three actors are involved: a data requester, a data provider, and a data owner.
**Context.** The Data Requester wants access to certain data and depends on the Data Provider for it. The Data Owner has full disposition on who can access them. The Data Owner delegates the permission to disclose data to the Data Provider. The Data Provider, however, can disclose the data only after the Data Requester has provided the necessary credentials (issued by the Data Owner). Shown in the part (a) of the figure.

**Requirement.** The Data Requester shall access information if he is an authorized actor.

![Figure 32. Access control pattern](image)

**Solution.** Once the Data Provider receives an access request, he requests an authorization certificate from the Data Requester. The Data Requester forwards this request to the Data Owner. If the Data Owner provided the required evidence, the Data Provider discloses the data to the Data Requester. A SI* diagram representing the solution is presented in part (b) of the figure where the added elements and relations are represented in blue.

**Consequences.** The pattern solves the problem of granting the permission to access data to the Data Requester. However, new issues may arise after the application of this pattern. For instance, the application designers should verify whether the Data Requester actually needs that permission to achieve his duties. Other issues may arise due to the expectations of the Data Owner about the use of the data by the Data Requester.
1.8 Contextual Requirements Modelling and Analysis (UNITN)

1. Description
Adaptability to context, the environment in which the system operates, is an essential feature for emerging computing paradigms such as mobile and ubiquitous computing. Contextual adaptation should be taken into account since requirements analysis, as a strong mutual influence between requirements and context exists. On the one hand, context is a main factor to determine relevant requirements, the alternative ways to meet requirements, and the quality of each of such ways. On the other hand, in order to meet its requirements, a system has often to make changes in the context. We develop contextual goal model as a requirement model that weaves together variability of both context and requirements. We associate between the goal satisfaction variability and context variability. We also provide constructs to analyze context and identify concrete facts that imply it. Moreover, we develop a set of reasoning mechanisms to analyze contextual goal models addressing various problems:
- Detecting inconsistency in context specification.
- Providing runtime rationale for the derivation of requirements in a way adaptive to context and user preferences.
- Detecting conflicts between requirements happening as a consequence of changes in the context they lead to.
- Economizing the model by deriving a set of requirements that leads to a system developed with minimum costs and operable in all of the analyzed contexts.

2. Detailed description of basic operation
Modelling and reasoning about contextual requirements consists of the following steps:

Goal analysis: in this activity, high level goals are defined and refined in top-down way. Each refinement step is followed by a context analysis.

Context analysis: this activity weaves goal modelling with context aiming to link the requirements, at the goal level, to the context in which they are activated and adoptable. Context analysis activity is composed of: identifying contextual variation points at the goal model, and analyzing the context specified at these points.

Specifying logical relations between contexts: after the refinement of each context, the logical relations (implications and contradictions) between it and the previously refined contexts need to be specified. These relations are essential for the forthcoming reasoning about contextual goal models.

Reasoning about contextual goal models: this activity is supported by our developed prototype tool (RE-Context). The tool allows reasoning about contextual goal models for different reasons. It analyzes a contextual goal model in order to detect inconsistency in contexts specified on it and potential conflicts among its executable processes (tasks). Moreover, the automated reasoning tool allows us to validate whether the model reflects stakeholders’ requirements. To this end, this reasoning derives and shows to stakeholders the goal model variants that reflect a given context and user priorities. It also elicits variants to goal satisfaction that are of minimum costs and able to reach top-level goals in all analyzed contexts.

Identifying monitoring requirements: after context analysis and reasoning terminate, the analyst can identify the monitoring requirements. Monitoring requirements are fundamental to develop systems adaptive to context. We identify these requirements in terms of the data to collect from the system environment and the equipments needed to collect them. The source of these requirements is the context analysis we do at each contextual variation point.
3. An example

In a health care institute for people with dementia, a caregiver may have the goal to "involve patient in social activities" (Goal_1) whenever "the patient is feeling bored" (Context_1). The caregiver can reach Goal_1 by "taking the patient for a trip in the city" (Goal_2) or by "asking a relative of the patient to come" (Goal_3). Goal_2 is adoptable if "the city is not crowded" (Context_2), since people with dementia usually get anxious in crowded places. Goal_3 is adoptable if "the patient has relatives or friends that can come" (Context_3). The requirements model of a smart home designed to support the patient should operate to meet the caregiver's goals Goal_1, Goal_2, and Goal_3, reflecting his rationale in meeting goals in context-adaptive manner: (i) Goal_2 OR Goal_3 → Goal_1 and (ii) if Context_1 AND Context_2 then Goal_2 and (iii) if Context_1 AND Context_3 then Goal_3.

An example of the context analysis is shown in the following figure. Where context is specified as and OR formula of world predicates. For an actor, possibly the system, a world predicate could be verifiable (fact) or not verifiable (statement). A statement S can be supported by a formula of world predicates if that formula provides enough evidence to the truth of S. The target of context analysis is to identify a formula of facts that reduces the primary context specification which may contain statements.

Figure 33. An example of the context analysis
1.9 Modelling Interaction between Agents via Commitments (UNITN)

1. Description

Social commitments are a conceptual abstraction that represents social relationships between different agents. Commitments are made in a socio-legal context, which defines the rules of engagement for agents in this context, such as penalties and compensations. A commitment is a promise (a contract stipulated in the socio-legal context) between a debtor and a creditor to bring about a certain state of the world. A conditional commitment is a conditional promise: the debtor will do something if some state of the world is brought about before (typically by the creditor). Commitments are an adequate abstraction to represent service-oriented settings, for they represent—at the business level—the interface that service providers (debtors) offer to service consumers (creditors).

Commitments have been used for different purposes. First, they have been proposed as a social abstraction to represent agent communication which exists independently from mentalistic notions that are of one agent (e.g. beliefs, goals, intentions). Second, their combination in protocols has been devised as a way to represent flexible inter-organizational business processes: they capture the business rationale for interaction between participants and do not rely on inflexible business process specification in terms of sequence of actions and messages. Third, they have been explored in the context of service-oriented settings, where an agent having some goals and some holding commitments to fulfil wants to understand if he will succeed in their fulfilment by exploiting its capabilities and making/getting commitments to/from other agents.

2. Detailed description of basic operation

A commitment is of the form \( C(\text{debtor}, \text{creditor}, \text{antecedent}, \text{consequent}) \), where debtor and creditor are agents, and antecedent and consequent are propositions. A commitment \( C(x, y, r, u) \) means that \( x \) is committed to \( y \) that if \( r \) holds, then it will bring about \( u \). If \( r \) holds, then \( C(x, y, r, u) \) is detached, and the commitment \( C(x, y, T, u) \) holds (\( T \) being the constant for truth). If \( u \) holds, then the commitment is discharged and doesn’t hold any longer. All commitments are conditional; an unconditional commitment is merely a special case where the antecedent equals \( T \).

Examples 1–3 illustrate these concepts. In the examples, EBook is a bookseller, and Alice is a customer; let \( BNW \) and \( S12 \) refer to the propositions \( \text{Brave New World has been delivered} \) and \( \text{payment of } S12 \text{ has been made} \), respectively.

Example 1. (Commitment) \( C(\text{EBook}, \text{Alice}, S12, \text{BNW}) \) means that EBook commits to Alice that if she pays \( S12 \), then EBook will send her the book \( \text{Brave New World} \).

Example 2. (Detach) If Alice makes the payment, that is, if \( S12 \) holds, then \( C(\text{EBook}, \text{Alice}, S12, \text{BNW}) \) is detached. In other words, \( C(\text{EBook}, \text{Alice}, S12, \text{BNW}) \land S12 \Rightarrow C(\text{EBook}, \text{Alice}, T, \text{BNW}) \).

Example 3. (Discharge) Suppose now EBook sends the book (if \( BNW \) holds), then both \( C(\text{EBook}, \text{Alice}, S12, \text{BNW}) \) and \( C(\text{EBook}, \text{Alice}, T, \text{BNW}) \) are discharged. That is to say, \( BNW \Rightarrow \neg C(\text{EBook}, \text{Alice}, T, \text{BNW}) \).

Importantly, an agent can manipulate commitments by performing certain operations (technically, speech acts). The commitment operations are reproduced below. Create, Cancel, and Release are two-party operations, whereas Delegate and Assign are three-party operations.

Create\((x, y, r, u)\) is performed by \( x \), and it causes \( C(x, y, r, u) \) to hold.

Cancel\((x, y, r, u)\) is performed by \( x \), and it causes \( C(x, y, r, u) \) to not hold.

Release\((x, y, r, u)\) is performed by \( y \), and it causes \( C(x, y, r, u) \) to not hold.

Delegate\((x, y, z, r, u)\) is performed by \( x \), and it causes \( C(z, y, r, u) \) to hold.

Assign\((x, y, z, r, u)\) is performed by \( y \), and it causes \( C(x, z, r, u) \) to hold.
We introduce Declare($x, y, r$) as an operation performed by $x$ to inform $y$ that $r$ holds. This is not a commitment operation, but may indirectly affect commitments by causing detaches and discharges. In relation to Example 4, when Alice informs EBook of the payment by performing Declare(Alice, EBook, $12$), then the proposition $12$ holds, and causes a detach of C(EBook, Alice, $12$, BNW).

A deductive strength relation can be defined between commitments: $C(x, y, r, u)$ is stronger than $C(x, y, s, v)$ if and only if $s$ entails $r$ and $u$ entails $v$. So, for instance, a detached commitment $C(x, y, T, u)$ is stronger than the commitment before detachament $C(x, y, r, u)$. This means that an agent that commits for something will also commit to deliver something less for the same price. Equally, an agent that commits to do something at a certain price will also commit to deliver the same at a higher price.

A commitment arises in a social or legal context. The context defines the rules of encounter among the interacting parties, and often serves as an arbiter in disputes and imposes penalties on parties that violate their commitments. For example, eBay is the context of all auctions that take place through their service; if a bidder does not honour a payment obligation for an auction that it has won, eBay may suspend the bidder’s account.

Commitments are grounded in interaction via messaging. This means that, given a message exchanged between two agents in a socio-legal context, an observer can objectively derive the meaning of the message in terms of commitments. The table below exemplifies such notion showing a mapping between domain-specific and commitment-oriented messages. An Offer message corresponds to the creation of a commitment; an Accept message corresponds to the opposite commitment. A reject message corresponds to releasing the debtor. Deliver and pay messages correspond to declaring that certain states of the world have been brought about.

<table>
<thead>
<tr>
<th>Domain-Specific Message</th>
<th>Commitment-Oriented Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offer(E, A, $12$, BNW)</td>
<td>Create(E, A, $12$, BNW)</td>
</tr>
<tr>
<td>Accept(A, E, BNW, $12$)</td>
<td>Create(A, E, BNW, $12$)</td>
</tr>
<tr>
<td>Reject(E,A, $12$, BNW)</td>
<td>Release(E, A, $12$, BNW)</td>
</tr>
<tr>
<td>Deliver(E, A, BNW)</td>
<td>Declare(E, A, BNW)</td>
</tr>
<tr>
<td>Pay(A, E, $12$)</td>
<td>Declare(A, E, $12$)</td>
</tr>
</tbody>
</table>

From the perspective of an agent, reasoning can be performed about the goals of the agent. Given some role in a protocol and some goal that the agent wants to achieve, goal support verifies whether an agent can potentially achieve his goal by playing that role. A goal can be supported by an agent in several ways:

The agent has a capability for it. Consider Alice’s goal payment. *Alice supports the goal if she has a capability for it.*

The agent can get an appropriate commitment from some other agent about the state of affairs that the goal represents. Consider Alice’s goal BNW. The commitment $C($merchant, Alice, payment, BNW$)$ from some merchant supports the goal, but only if Alice supports payment. The intuition is that Alice won’t be able to exploit the merchant’s commitment unless she pays.

The agent can make a commitment to some other agent for some state of affairs (presumably one that the latter would be interested in) if the latter brings about the state of affairs that the goal represents. *Consider EBook’s goal payment. He can support this goal by making an offer to some customer, that is, by creating $C($EBook, customer, payment, BNW$).*

An and-decomposed goal is supported if all children are supported

An or-decomposed goal is supported if at least one children is supported.
3. An example

We show how we can model a service-oriented setting and reason from the perspective of one agent. We take a setting concerning flight tickets purchase via a travel agency. Four main roles participate in this protocol: travel agency, customer, airline, and shipper. Customers are interested in purchasing flight tickets for some reason (e.g., holidays or business trips), travel agencies provide a tickets-selling service to customers by booking flight tickets from airlines, shippers offer a ticket delivery service.

![Figure 34. An example model of a service oriented setting](image)

The figure above describes the protocol in the travel agency scenario. The protocol is defined as a set of roles (circles) connected via commitments; the commitments are labeled (C_i). Commitments in the figure are represented graphically. The arrows go from the debtor to the creditor. So, for instance, commitment C_5 should be understood as C(airline, customer, ticketsShown, boardingAllowed).

The following figure shows the situation where agent Fly has adopted the role travel agency in the protocol of Fig. 4; the other roles are not bound to agents. Fly has one top-level goal: selling tickets (ticketsSold). In order to support it, three sub-goals should be supported: tickets should be obtained, tickets should be delivered to the customer, and the service should be paid. Tickets can be obtained if the tickets are reserved and they have been paid. Fly is capable of goal ticketsPaid. There are two ways to deliver tickets: either electronic tickets are e-mailed or tickets are posted. In order to send tickets via mail, Fly has to ship the tickets and pay for the shipping. Fly is capable of eticketsEmailed. E-mailing tickets contributes positively (+S) [18] to softgoal costsKeptLow, whereas sending via shipping contributes negatively (--S) to such softgoal.

![Figure 35. An example of goal and sub-goals](image)
We present now some queries concerning goal and commitment support that can be run against the specification of previous figure.

Query 1. Can Fly support goal ticketsSold?
The answer to this query is yes. Fly can support ticketsObtained by using its capability for ticketsPaid and getting C₄ from some airline. Fly supports ticketsDelivered via its capability for eticketsEmailed. Fly can support servicePaid by making C₂ to some customer.

An alternative solution involves sending tickets via shipping. Fly could support ticketsShipped and shippingPaid if it makes C₃ to a customer (which supports servicePaid and shippingPaid) and get C₁ from some shipper (to support ticketsShipped).

Another solution includes supporting both eticketsEmailed and ticketsSent: both C₂ and C₃ are made to the customer.

Query 2. Can Fly support goals ticketsSold and costsKeptLow?
This query adds an additional constraint to Query 1: supporting softgoal costsKeptLow. The only solution is when tickets are e-mailed: eticketsEmailed contributes positively to costsKeptLow and the softgoal gets no negative contribution. Posting tickets does not work: ticketsSent contributes negatively to costsKeptLow.

1.10 Thales Risk Assessment Domain Specific Modelling Language (THALES)

1. Description
System security engineering classically involves: 1) the analysis and assessment of security risks encountered by the system, 2) the specification of requirements for security measures to address those risks, and 3) the design, development, integration and validation of a security architecture, functions and mechanisms that address those requirements.

Our present work is focusing on security engineering activities 1 and 2 above. Our objective is to provide adequate and efficient tooling to security engineers for an effective integration of security engineering in the process of critical system design; this will enable a better targeting of security specifications.

2. Detailed description of basic operation
The Thales Security Domain Specific Modeling Language (DSML) tool builds upon model-based engineering methods and techniques and provides risk models for a security risk analysis and management. The main functionalities of the targeted tool will be:

- a security risk analysis method and tool, inspired from [EBIOS] and [MEHARI], and adapted to Thales’ specific context, and based on system model views.
- libraries and reference models, which constitute a first knowledge base that serves the development of the models;
- automation support, which shall enhance the productivity of the tool by providing: automatic recommendations, e.g. vulnerabilities attached to a type of component of the system based on knowledge from a vulnerability library.

The risk management offered by the tool follows the process described here below:
3. An example

The picture below shows a very simple example of a risk analysis performed on a system composed of a Web browser and a Web Server (in blue) considering an Admin database password as a data (in green) transiting through the communication channel between the browser and the server (in grey).
1.11 Thales SOA Modelling Suite (SMS)

1. Description
The purpose of the Thales SOA Modelling Suite is to capture the different concerns related to SOA architectures specifications and implementations. Ultimately, it shall provide its users with domain specific languages (DSLs), as well as their corresponding graphical representations, that allow specifying efficiently SOA concerns. Such DSLs are designed by capturing the concepts associated with SOA standards, technologies and Thales engineers’ specific requirements.

2. Detailed description of basic operation
Users can create a project and use the modeller to edit SOA models, eventually generating the appropriate documents. Multiple features are available including high level service and message type specification, logical view specification, physical view specification, BPMN 2.0, enterprise integration patterns, etc.

3. An example
Not significant - as the Thales security assessment domain specific modelling language (DSML) is not yet integrated.

1.12 Property and Policy Tools (LJMU)

1. Description
In order to understand and analyse the nature of composed services, properties of the individual services that make up the composition must be either determined or predefined. In addition, a method is also needed for the definition of policies that are to be applied, either to individual services or to the composition as a whole.
The tools described here allow such descriptions to be constructed using a collection of straightforward user-interfaces. The resulting properties are output in a simple XML file format that can be read in by other tools.

Two tools are described specifically. These are the PropertyInterface and the CompositionClient property selection interface. In future development we also expect a further tool, the PolicyInterface, to complement these. They each serve a different purpose as follows.

The PropertyInterface allows property sets to be defined. These specify a range of possible properties that a service or device might support. The tool outputs Property Set files.

The CompositionClient property selection interface accepts Property Set files and allows specific properties to be selected from those that are available. It outputs Property files.

The PolicyInterface tool will allow the creation of policies that amount to logical and arithmetic expressions with variables and constants taken from the properties available. The tool will also allow policies to be resolved against specific sets of properties to determine policy satisfaction. The tool will accept Property Set files (for policy creation) and will generate Policy files as output. It is expected to accept Policy files and Property files for policy testing.

2. Detailed description of basic operation

When considering the security properties of interacting services, a variety of different types of information are required. Most notably, in order to evaluate the properties of the overall system (especially if this is to be done in a distributed manner), the most important pieces of information are likely to be the properties of the individual services, and the manner in which these services interact.

In addition to this, the security of a system will be a function of the policy (or policies) in force. Methods are therefore needed to allow properties and policies to be understood, transferred between services, defined and so on.

Some of this functionality can be provided by the property tools described here.

To tackle the issue of property definition, we have developed a property interface tool (as shown in Figure 38) that allows node properties to be defined and saved in a simple XML format. These files can then be automatically transferred between nodes to help identify the level of security in a specific scenario.

The interface allows sets of properties to be defined based on various criteria. The purpose is not to define specific properties that would apply to a given service, but rather a range of property classes that might apply. The actual properties that apply to a given service would then be selected from within this range (where some of these properties might also be dynamic and change over time).

Properties are defined based on type; for example they might be numerical properties, string properties, boolean properties, enumerated properties and so on.

Having specified a type and name for a property, its nature can then be changed using the interface. For example, a numerical property might be assigned bounds, or the values for an enumerated property can be specified.

Properties can be defined from scratch, or selected from a list of previously defined properties. For the simplest operation, the tool can act as a simple menu: different property types can simply be selected from a list.

The output from the tool is an XML file that specifies the property set. An example of the simple format used for such files can be seen in Figure 39.
Figure 38. Property Interface for defining service properties.

```xml
<?xml version="1.0" encoding="utf-8"?>
<PropertySet>
  <Property id="Name" type="string" />
  <Property id="Type" type="option">
    <Option enum="1">Ambulance Service</Option>
    <Option enum="2">Bank</Option>
    <Option enum="3">Dept. of Transport</Option>
    <Option enum="4">Event Organiser</Option>
    <Option enum="5">External Node</Option>
  </Property>
  <Property id="External" type="bool" />
  <Property id="Sensitivity Level" type="int" rangeMin="0" rangeMax="9" />
  <Property id="Staff Skills" type="option">
    <Option enum="1">Low</Option>
    <Option enum="2">Medium</Option>
    <Option enum="3">High</Option>
  </Property>
  <Property id="Firewall" type="bool" />
  <Property id="Asset Value" type="option">
    <Option enum="1">Low</Option>
    <Option enum="2">Medium</Option>
    <Option enum="3">High</Option>
  </Property>
  <Property id="Risk Level" type="float" />
</PropertySet>
```

Figure 39. Example Property Set file.
In order to define the specific properties that relate to a specific service or device, property set files can be used as a template. In our current implementation, property set files are transferred onto devices or made available to services. The user is then required to select the precise properties that apply for a particular node using the CompositionClient tool.

More generally, properties might be generated automatically or as part of a formal verification process. At present however, allowing the user to select the properties provides a simple way to experiment with different properties that might hold for different interacting services.

**Figure 40. CompositionClient property selection interface**

Figure 40 shows an example of a property selection interface that might be presented to the user of the CompositionClient tool. The interface has been generated directly and automatically from a Property Set file, in order to allow the user to easily choose from all available options. The interface will therefore be different depending on the property set in use.

Once the relevant properties have been selected by the user via this interface, the CompositionClient then generates a Property file, an example of which is shown in Figure 41. As can be seen from the file structure, this property definition file is similar to that of the Property Set file, the main different being that actual values are specified for each of the options, in addition to the range of possibilities available.
3. An example

The tools can be used to define property sets and properties that might apply to individual services.

With the current configuration of tools, the process might work along the following lines.

During creation of a service a Property Set file is generated using the PropertyInterface based on the possible properties that the service might satisfy.

During execution a Property definition file is generated for a given service. This file might be created by a user with the CompositionClient in a testing environment. For real deployment, this file would be more likely to be generated automatically.

The Property definition files for a collection of services might be sent to a policy checking service (such as MATTS as described in Section 6.2) that compares them against the specified policy to determine whether or not the service composition is compliant with the composition policy in force.

The steps above provide an example of how the tools might be used. However, the tools do not necessarily need to be used together, and could be used independently for a variety of tasks.

2 Trustworthiness prediction module

2.1 Role-based Trust Management with quantitative notion of trust (CNR)

1. Description

The proposed tool/technique, Role-based Trust Management with quantitative notion of trust (RTML with weights), is a framework including a language that enable to express the attributes (or roles) that a principal assigns to a subject with quantitative notions of trust, the policy to infer new (derived) attributes from the existing one, along with a deduction engine that performs the process of deriving the new attributes of a subject exploiting the original attributes and the policy. The attributes assigned to subjects are represented through credentials issued by the principal that embody the role and the weight that expresses the measure of trust. The trust metrics are expressed through semi rings.

This approach brings together in a hybrid framework two usual notions of trust:

- Trust based on credentials, policies,
- Trust based on recommendation/reputation.

The framework is extremely flexible being parametrized by the appropriate semiring.

2. Detailed description of basic operation

Role-based Trust Management (with weights)
The Role-based Trust Management framework RTML (Li et al., 2002; Winsborough et al., 2003) provides policy language, semantics, deduction engine, and concrete tools to manage access control and authorization in large-scale and decentralized systems. RTML combines the strength of Role-Based Access Control (RBAC) and Trust-Management (TM). RBAC was developed to manage access control in a single organization in which the control of role membership and role permissions is relatively centralized in a few users. RTML takes from RBAC the notion of role to assign permissions to users. TM is an approach to distributed access control and authorization in which access control decisions are taken on the base of policy statements made by multiple principals, e.g., Grid sites. From TM, RTML takes the principles of managing distributed authority through the use of credentials, as well as some notation denoting relationships between those authorities.

The main concept in RTML is the notion of roles: each principal has its own name space for defining roles, and each role is compounded by the principal name and a role term. For example, if $A$ is a principal and $r$ is a role term, then $A.r$ denotes the role $r$ defined by principal $A$. Only $A$ has the authority to issue policy statements defining the members of the role $A.r$. Roles may be parametrized, e.g., a basic credential of the form $A.r(p) \leftarrow D$ means that $A$ assigns to $D$ the role term $r$ with parameter $p$.

In the following credential, organization IIT assigns the role of IIT researcher to Alice, whose distinguished name is "CN=Alice, OU=IIT, O=CNR, L=Pisa, C=IT'.

$$IIT.\text{researcher('CN=Alice, OU=IIT, O=CNR, L=Pisa, C=IT')} \leftarrow Alice$$

In (Martinelli et al., 2007) a basic set of RTML credentials has been enriched with trust, in order to express not only the fact that an authority assigns to someone a certain role, but also that a principal trusts someone for performing some functionality $f$, or for giving a recommendation regarding a third party able to perform that functionality. This follows the interpretation of trust encoded in the transitive trust model of (Jøsang et al., 2003; Jøsang et al., 2006), according to which trust is always linked to a purpose. The most natural situation is when one trusts another for performing a certain function/task. It is often common that principals ask other principals for suggesting/recommending a third party able to performing that function or task.

The following language enriches RTML with trust, by also specifying a trust weight $v$, i.e., a quantification of the confidence one places in the positive outcome.

Simple member. $A.r(p,v) \leftarrow D$. The role $A.r(p)$ has weight $v$.

Simple containment. $A.r(p,v) \leftarrow v_2 A_1.r_1(p_1,v_1)$. According to $A$, all members of role $A_1.r_1(p_1,v_1)$ with weight $v_1$ are members of role $A.r(p,v)$ with weight $v = v_1 \otimes v_2$. $v_2$ is a constant filtering $A_1$'s authority with $A$'s authority.

Linking containment. $A.r(p) \leftarrow A_1.r_1(p_1).r_2(p_2)$. If $B$ has role $A_1.r_1(p_1)$ with weight $v_1$ and $D$ has role $B.r_2(p_2)$ with weight $v_2$, then $D$ has role $A.r(p)$ with weight $v = v_1 \otimes v_2$. This works as a sort of role-based delegation.

Intersection. $A.r(p) \leftarrow A_1.r_1(p_1) \cap A_2.r_2(p_2)$. This statement defines that if $D$ has both roles $A_1.r_1(p_1)$ with weight $v_1$ and $A_2.r_2(p_2)$ with weight $v_2$, then $D$ has role $A.r(p)$ with weight $v = v_1 \oplus v_2$.

Weights are not explicitly expressed in the linking and the intersection containment statements. Operators $\otimes$ and $\oplus$ combine the trust measures in the composed credentials expressed by the simple, the linking, and the intersection containment. For the sake of readability, weights are not explicitly expressed in the linking and the intersection containment statements.

Generally speaking, $\otimes$ combines opinions along a path, i.e., A's opinion for B is combined with B's opinion for C into one indirect opinion that A should have for C, based on what B thinks about C. $\oplus$ combines opinions across paths, i.e., A's indirect opinion for X through path $path_1$ is combined with A's indirect opinion for X through path $path_2$ into one aggregate opinion that reconciles both. To work
properly, these operators must form an algebraic structure called a c-semiring, (Theodorakopoulos & Baras, 2004).

3. An example

We present sets of user credentials that could be evaluated with respect to an access policy by the RTML authorization service in real job requests.

Trust Management example. The Center of Electronic Computation (CCE) offers a computational service to university students which are carrying out a stage at ABC Company. To access the service, a user need to supply a credential issued by the ABC Company, asserting the stage, and a credential issued by the university, granting him the role of student. Also, a credential chain is required to verify that this university is admitted by CCE. Each student with right credentials can access the computational services, even if her identity has not been registered by CCE.

Let us suppose that Alice, which is a student of University of Pisa and a collaborator at the ABC Company, wants to access the CCE computational service. She supplies the following credentials:

\[
\begin{align*}
\text{UniPI.Student}(&\text{university='University of Pisa', department='CS', id='1999s131', firstname='alice', lastname='black'}) \leftarrow \text{Alice} \\
\text{ABC.Collaborator}(&\text{role='stage', firstname='alice', lastname='black'}) \leftarrow \text{Alice} \\
\text{MIUR.University}(&\text{name='University of Pisa'}) \leftarrow \text{UniPI}
\end{align*}
\]

The Access Policy stored by CCE is the following:

\[
\begin{align*}
\text{CCE.University(name=?) } & \leftarrow \text{MIUR} \\
\text{CCE.Student(} & \text{university=refuni, department=?, id=?, firstname=?, lastname=?) } \leftarrow \text{University(name=refuni)} \\
\text{CCE.ABCGuest } & \leftarrow \text{ABC.Collaborator(role=?, firstname=reffirst, lastname=reflast)} \\
\text{CCE.Student(} & \text{university=?, department='CS', id=?, firstname=?, lastname=?) } \leftarrow \text{CCE.Student(} \text{university=?, department='CS', id=?, firstname=?, lastname=?)}
\end{align*}
\]

Symbol ‘?’ denotes a parameter whose value is not specified.

In the first and second credential, principals UniPI and ABC assign to Alice the attributes of, resp., student and collaborator, with the specified parameters. The owner of the third credential is UniPI and not Alice. This statement could be used to infer information about Alice's attributes. Indeed, CCE considers as universities the principals that are considered Universities by MIUR, the Italian Ministry for University and Research.

The credentials and the access policy are retrieved by the RTML PDP from the repositories, and passed to the RTML Engine. The RTML Engine transforms them in the following set of Datalog statements:

\[
\begin{align*}
\text{Student(} & \text{UniPI, Alice , x1, x2, x3, x4, x5) } : x1 = 'University of Pisa', x2 = 'CS', x3 = '1999s131', x4 = 'alice', x5 = 'black' \\
\text{Collaborator(} & \text{ABC, Alice, x1, x2, x3, x4) } : x1 = 'stage', x2 = 'IS137', x3 = 'alice', x4 = 'black' \\
\text{University(} & \text{MIUR, UniPI, x1) } : x1 = 'University of Pisa' \\
\text{University(} & \text{CCE, Y, x1) } : \text{University(MIUR, Y, x1)} \\
\text{Student(} & \text{CCE, Y, x1, x2, x3, x4, x5) } : \text{University(CCE, X, y1), y1 = uniname, Student(X, Y, x1, x2, x3, x4, x5), x1 = uniname} \\
\text{ABCGuest(} & \text{CCE, Y) } : \text{Collaborator(ABC, Y, x1, x2, x3, x4), x3 = namefirst, x4 = namelast} \\
\text{Student(} & \text{CCE, Y, y1, y2, y3, y4, y5), y4 = namefirst, y5 = namelast} \\
\text{Guest(} & \text{CCE, Y) } : \text{Student(CCE, Y, y1, y2, y3, y4, y5)}
\end{align*}
\]
The RTML PDP reads from a configuration file that either attribute *ABCGuest* or *Guest* is required for the service requested by Alice. Then, the RTML PDP invokes the Prolog Engine with the previous statements to perform the evaluation of the following goals:

? :- ABCGuest(CCE, Alice)
? :- Guest(CCE, Alice)

A positive evaluation of both the goals allows Alice’s access to the computational service. If the second credential is not owned by Alice because she is not an ABC collaborator, the second goal is not verified, and she can only access the service with the *Guest* role. This role may have some limitation with respect to the privilege granted to an ABC collaborator.

4. References

3 Monitor trustworthiness module

3.1 xESB (UNITN)

1. Description
xESB is an instrumented JBI ESB (Enterprise Message Bus) for the enforcement of security policies that are organization wide. xESB is able to enforce both access and usage control policies. The rich enforcement semantics of xESB allows not only to reject ESB messages that violate a policy but also to compensate that violation. xESB also introduces and supports indicators, which aim at helping security administrators analyze and derive useful information about policy violations and their impact to the overall security of the organization.

2. Detailed description of basic operation
xESB is an enhanced version of ESB, used to monitor and enforce preventive and reactive policies, both for access control and usage control policies, and both inside one domain and between domains. The ESB itself is a piece of middleware that is placed between the various services of a SOA application offering a layer of communication and integration logic to mitigate technology disparities between communication parties. JBI (Java Business Integration)1 standardizes deployment and management of services deployed on an ESB. It describes how applications need to be build to be integrated easily.

The enforcement process behind xESB comprises message interception, decision making, and action enactment based on decisions taken in the second step. The runtime enforcement process starts when a message, to which at least one policy applies, is intercepted. Once a policy – relevant message is
intercepted, it needs to be evaluated against the applicable policy. The decision component performs policy matching by examining all policies in the policy repository and evaluating them against the current intercepted message. Once a verdict is reached, an action takes place. The authors have implemented five basic enforcement actions to approach usage control enforcement. Actions can be differentiated as preventive or corrective. The first ensure that a policy violation will not happen, whereas the second try to compensate already happened violations. We can conclude that, the rich enforcement semantics of xESB allows not only to reject ESB messages that violate a policy, but also to compensate that violation.

xEsB also introduces and supports indicators aiming to help the security administrator analyze and derive useful information about policy violations and their impact to the overall security of the organization.

3. An example
A mobile telecommunications company must be able to enforce certain policies while offering services to its clients, such as ‘Log starting time and duration of calls’ or ‘Hide initiators number in outgoing calls’. For the first policy, a simple mechanism should signal the start of a call and its duration. Capturing and logging these events should leave the application unaltered. What is therefore required is a control that is able to filter messages and to duplicate them to a logging service.

For the second policy, a mechanism is needed to filter outgoing calls from incoming ones. This can be achieved simply by looking at the type and parameters of the event (either a service invocation or a service response) and then modifying those that identify the initiator.

In composite services, a user might require to avoid services offered by a certain provider.

3.2 Trust Manager (TSSG)
1. Description
The Trust Manager receives alerts and metrics from components that monitor QoS and other properties of entities and stores ratings locally in a database. It can compute trust levels based on those ratings and forward the trust level to interested parties.

2. Detailed description of basic operation
The architecture of the Trust Manager is illustrated in Figure 42. It receives input from components monitoring the QoS and functional metrics for agents and computes ratings based on those metrics. The ratings are then time-stamped, categorised and stored. Categories of ratings refer to the types of metrics received. Trust evaluation is based on past behaviour and the reputation of each agent through direct experience, recommendation, referral, and roles. The trust level consists of a score that is calculated based on weighted ratings given to different categories of metrics. The trust level also includes a confidence degree which is affected by the recency of the scores and the quantity and variability of the ratings on which the trust score was based.
3. An example
The tool works as part of monitoring middleware to monitor run-time QoS (e.g. security, performance) and reputation (consumer feedback) of services and provide estimates of trust levels. QoS metrics, SLA violations, and other monitoring data are received by the Trust Manager, running in an application server, from other monitoring and event processing components. The trust manager creates ratings for agents based on the data and trust calculation uses the ratings in evaluating and predicting an agent’s trust. The prototype has been demonstrated in testing CoMiFin middleware for financial critical infrastructures [103].

4 Contract negotiation module

4.1 Security by Contract with Trust for mobile systems (CNR)

1. Description
The notion of Security-by-Contract (SxC) denotes an agreement between a mobile application and a platform (Mobile Device). This agreement specifies the permissible interaction between the application and its host device. To accomplish the contract, the application has to disclose its security-relevant features to any platform where it is deployed. The platform (the mobile device) has to check these features against its security policy and decide upon whether to sign the contract and accept the instalment of the application. It is also possible that the platform accepts some features and neglects others. This selection of features leads to customizing the mobile application to be consistent with its platform choice.

At runtime, the operation of the application is monitored by its platform to ensure and enforce security specified in the contract. This work also provides mechanisms to establish a run-time monitoring to both
- Enforce the security policy specified in the contract
Evaluate the mobile code compliance with the contract after the permission to some resources is given.

The trust level of an application may change by time according to its compliance with the contract. The experience a platform has with a mobile code will influence the trust level of that code and consequently its operation.

Currently, SxC is being extended to infer security-related behaviour from the program code itself instead of relying on the developer’s declaration.

As the SxC, the Security-by-Contract-with-Trust (SxCxT) is developed in order to have a unified framework for dealing with both security and trust on (i) mobile devices and (ii) distributed systems.

The notion of SxCxT denotes a paradigm with a trust management mechanism able to guarantee that a mobile application can be safely executed on mobile devices, according to applications trust recommendations.

The SxCxT extends the SxC with a run-time contract monitoring for allowing to manage the application trust level. In particular, the contract monitoring checks if the actual behaviour of the application is compliant with its contract and the trust management rewards and penalises the application depending on the answer of that check.

The main advantage of applying contract (instead of policy) monitoring is the higher efficiency deriving from using less security rules.

2. Detailed description of basic operation

The SxCxT framework runs on the customer’s platform and is designed to be efficient enough to be applied to any host configuration (e.g., minimising the computational requirements). Furthermore, a fundamental aspect of our framework is the trust relationship management. Indeed, malicious applications and providers must be penalised for future requests in order to prevent possible attacks.

Thus, we propose a mechanism for managing trust feedback according to the concept of mobile application criticality. Indeed, the SxCxT framework is able to adapt dynamically its security and trust behaviours according to the degree of criticality of the running application. The degree of criticality reflects how much an application may be considered critical with respect to security and trust aspects according to its type and category (i.e., according to which kind of data it may access or which resources it uses).

Our mechanism manages trust by rewarding and penalising mobile application’s provider according to its trust recommendation.
The S×C×T framework enforces security informed by trust according to the following steps:

Step 1 - Trust Assessment: Once a mobile application is downloaded on the mobile device, before executing it, the trust module decides if it is trusted or not according to a fixed trust threshold.

Step 2 - Contract Driven Deployment: According to this trust measure, the security module defines if just monitoring the contract or both enforce the policy and monitoring the contract going into one of the scenarios described in Step 3.

Step 3 - Contract Monitoring vs Policy Enforcement Scenarios: Depending on the chosen scenario, the security module is in charge to monitor either the policy or the contract and save the execution traces (logs).

Step 4 - Trust Feedback Inference: Finally, the trust module parses the S×C×T produced logs and infers trust feedback.

3. An example

Let us consider a Mobile Applications Marketplaces, like, for instance, Apple AppStore, Cydia, Android Market, in which mobile applications are released by a provider and customers can download and install them on their devices.

Let us also suppose that a trust recommendation is associated to each mobile application. Recommendation is managed by Mobile Applications Marketplaces from customers’ opinions.

Generally speaking, we consider a customer, owner of a mobile device, who needs to add a certain functionality to his own device. The customer asks for an application for doing that. Such mobile application is downloaded from the Mobile Applications Marketplaces.

Each mobile application comes with its contract released by the provider of the application itself. Let us suppose that a security policy is embedded on the customer’s device. It can be set by the customer himself or by the manufacturer of the device.

The customer decides to download or not the mobile application according to the given recommendation and the trust threshold he has set on his device. Indeed, let us consider to measure the level of trust of the application according to if:

- The application code respects the application contract
- The application contract respects the device policy.

The first statement is referred to the level of trust with respect to the application developer, while the second statement is referred to the level of trust of the Mobile application marketplace that is in charge of choosing the mobile application to provide to the owner of the device according to his/her specific requests.

Hence the mobile application level of trust is the composition of the level of trust of both the developer and the marketplace.

If the application is downloaded, whenever it is executed, some security mechanisms are needed for guaranteeing that the device’s security policy is satisfied.

In this scenario, we have applied the Security-by-Contract-with-Trust in order to guarantee that a downloaded application is executed without violating the security policy required by the customer.

One of the end-users of mobile device policies could consist of 2 rules:

1) The mobile application could access contact list but cannot automatically call.
2) The mobile application can not automatically send more than 5 SMS per day.

Consider a MIDlet, (kind of mobile application) that is developed to help for scheduling a meeting between the mobile device user and a set of his contacts. To do the work, the MIDlet may require access to the contact book of the mobile phone and send voice mails or SMS reminders. This means that the MIDlet security-relevant behaviour may not be consistent with the policies of the end-user.

In SxC, the MIDlet has to expose its behaviour and feature and the required access to the mobile phone resources, so that the contract can be signed or declined by the mobile device accordingly. This
contracting might be part of the deployment process so the user agree or deny some features, or can be automated. If the user do not allow for autonomous contacting of his contacts then this feature can be dropped of the installed application. During Runtime the behaviour of the MIDlet will be monitored to ensure compliance with the contract. For example, if the contracting confirmed the 1st rule then the application will be blocked/stopped when it tries to make a call. If the MIDlet has respected for a period of time this rule and no violation of the contract was reported then the level of trust in the MIDlet will be increased and vice versa.

4.2 SLA Management framework (ATOS)

1. Description

Service Level Agreements are probably the most important documents in every business-aware framework – at least they should be. Every distributed piece of software faces the problem to assure a decent Quality of Service. An elegant solution is to describe the QoS within an SLA, linked to a legally binding contract. The BEinGRID solution offers an implementation supporting the negotiation, evaluation and accounting of SLAs. SLAs are of vital importance for modern business models, such as Utility Computing, SaaS or IaaS, due to the requirement for observing a certain QoS when providing a service.

![The BEinGRID SLA Life Cycle](image)

2. Detailed description of basic operation

The BEinGRID SLA framework provides software components for:

- **SLA negotiation**: The parties involved, the client and the providers, have to agree on the terms of the SLA binding them. They also have to detail the liberties of each partner, with consequences to violations. This negotiation can be a long process, a well-known fact to sales representatives. An interface highlighting the debated issues is proposed in order to attain a faster compromise with respect to the conflicting parts of the SLA.

- **SLA Monitoring and Evaluation**: Once a provider has started providing access to its resources, it should monitor the operating resources. The monitored information is then used to prove the QoS agreed within the SLA is being respected. Some good practices in monitoring and output formats may be proposed. Knowing that a job is running correctly is not the only information of interest for managers and users. Other data like contract violations or global statistics are needed in order to verify the SLA. Evaluation is the process of analysing the previously monitored information. An evaluation daemon may be proposed, based on the monitoring tools developed.

- **SLA accounting**: The use of a resource generates an accounting sheet which describes the resources used and aligns them with the billing rules agreed in the SLA. This is not an invoice, but a base to draft the real financial exchange, which can be in disfavour of the provider in case of failure to comply with the compromised QoS. This subject is very sensible, and the development of tools for such themes should not be taken lightly.
3. An example

The following diagram reflects how the Negotiation module provides the initial interface for the rest of functionalities to be exposed to the requesting party. After Negotiation, Evaluation and Accounting requirements are satisfied, the Resources are made available to the user.

![Diagram of BEinGRID SLA components]

**Figure 45. The BEinGRID SLA components**

4.3 Security-Oriented Service Composition and Evolution

1. Description

This methodology is used as a security-oriented negotiation approach to service composition and evolution. The key building blocks of the framework are specifying the security objectives and properties at the service and system levels, the negotiation and re-negotiation technique for service composition and evolution, and checking the security compatibility between services and the satisfaction of system-level security goals. It focuses on developing methods that allow system developers to design required security into service compositions with predictability and to maintain or adapt service compositions in changed security context.

2. Detailed description of basic operation

Figure 46 shows the basic elements of the framework for security-aware service composition and evolution in the general context of system composition and evolution: system architecture design, system instantiation and composition, and system execution and evolution. In system architecture design, system requirements are defined and a system architecture is created. In system instantiation and composition, the system is implemented with various services that are typically created and maintained by third parties. The services are located, selected and used by the system. In system execution and evolution, the composed system is in operation and delivers the services to the clients. At this stage, the application may undergo changes, including replacing existing services, modifying system requirements, and so on.
In relation to system architecture design, there is a need for the specification of security objectives and properties for the composite system and individual services. Service instantiation and composition requires establishing security-oriented service contracts through negotiation. Service execution and evolution involves re-establishing security-oriented service contracts through re-negotiation. The three security-related key aspects are:

1. **Definition of security objectives and properties**: A two-level formal model and notation for describing and publishing high-level security objectives and detailed security properties of services and service compositions. A high level security objective is the abstract representation of a goal of a security policy. Security policies have certain security objectives which are essentially defined to withstand identified security threats and risks. A security function enforces the security policy and provides required capabilities. Examples of security objectives are integrity, confidentiality, authenticity, authorisation, non-repudiation from a user’s perspective. Detailed security properties consider security measures such as passwords, private keys, public keys, secret keys, shared keys and digital signatures among others which are typically derived from security functions. Security properties are a collection of security elements that are used to realize security objectives.

2. **Security contract negotiation for service composition**: Automated negotiation techniques for service composition, including the checking of security compatibility between interacting services and satisfaction of system-level security requirements. The first step during architecture instantiation is to select a collection of candidate services that can potentially satisfy the security objectives of the overall system. The candidate services are selected on the basis of conformity of their security objectives with the system level security objectives. Typically, each service might have a number of security properties that, individually or in combination with others, can fulfil its security objectives. The (required and ensured) security properties of a service need to match with those of the interrelated services in a composition and it may be that only some of their combinations can collectively satisfy the system-level objectives.

3. **Security contract re-negotiation for system evolution**: Automated re-negotiation mechanisms for system evolution aiming to re-establish security compatibility and security requirements satisfaction in a changed context. As a consequence of system evolution, a service composition may require adaptation. For example, some original composition contracts
between individual services may no longer be complied with. Certain system-level security objectives may no longer be achievable. New system-level objectives may be required. Certain services may need to be replaced with new ones. Even the original system architecture may need to be changed. In any case, compatibility between individual services in the system needs to be achieved or re-established, and system-level objectives need to be delivered. All these need to be reflected in a set of contracts that exist, adapted or new. Again, the satisfaction of security requirements is part of this adaptation process. In general, system evolution results in one or more of the following scenarios in terms of establishing service contracts:

1. Adapt an existing service contract under different terms.
2. Establish a service contract with a newly selected service.
3. Change the existing composition architecture, and consequently instantiate it by adapting certain existing service contracts and establishing new contracts with new services.

Re-negotiation can be triggered by changes in the security requirements at the service and/or system levels. In any case the existing security contracts are analysed to determine the affected services and identify the required contracts for re-negotiation. If the security objectives of the existing services satisfy the new security objectives at the system level, re-negotiation on the security properties is carried out in a similar way as the initial negotiation.

3. An example

For illustration purpose, Figure 47 shows a simple example of the artefact model of security objectives at the architecture level, as well as security objectives and properties at the service level for a healthcare system. The example has a very simple system architecture consisting of three service types offered by three different service providers. The system as a whole delivers a service called HealthCare(). It provides a patient with a diagnosis report based on his/her test results such as blood test and CT scan. The system also offers a prescription based on the diagnosis report from another specialist service. This system-level service has three associated security objectives that are made public. The figure also shows that each service in the architecture offers a service, namely, SetTestResult(), SetDiagnosis(), or SetPrescription(); each has a security objective, i.e., confidentiality of the patient name, integrity of the diagnosis report, and non-repudiation of the prescription respectively. These objectives are published along with the services by the service providers.

![Figure 47. A simple example of defining and specifying security objectives and properties](image)

The paper (Han and Khan, 2006 [113]) has given three examples of defining system-level security objectives. In a healthcare system involving multiple services, (a) patient names should be kept
confidential in all transactions between services, (b) prescriptions must carry the verifiable identity of the issuing services, and (c) diagnosis reports must not be modified or tampered with by any unauthorised services. These security requirements should be codified into security objectives in a way that they are machine readable, and can be reasoned about by other entities. A security objective is represented in the following form: \textit{security objective (entity)} where \textit{security objective} can be any security goal such as confidentiality, integrity, non-repudiation, authorisation, authenticity, availability, and so on. The \textit{entity} can be any business object such as a report, a file, a person, a data, etc. By applying this simple format to the above three examples, there are three security objective descriptions:

(a) \textit{Confidentiality (patient name)};
(b) \textit{Non-repudiation (prescription)};
(c) \textit{Integrity (diagnosis report)}.

These security objectives are associated with specific functionalities offered by the system, and spell out the ultimate security goals about the entities of these functionalities, but ignoring the lower level details.

In addition to security objectives at system level, the framework (reference above) also defines security properties at services level. These properties are required to achieve the security objectives, and are often kept \textit{private} but are exploited for service negotiation to satisfy compositional security requirements. It formulates a security property as follows

\textit{Security property (element1,\ldots,element n)}

Where \textit{security property} represents a particular type of security function such as encryption, digital signature, key generation and so on. By applying this format to health care examples, the following security property specifications are obtained:

(a) \textit{Encrypted (patient name, shared key)};
(b) \textit{Signed (prescription, secret key)};
(c) \textit{Encrypted (diagnosis report, public key)}.

The above properties state that the \textit{patient name} is encrypted with a \textit{shared key}, the \textit{prescription} is signed with a \textit{secret key} and the \textit{diagnosis report} is encrypted with a \textit{public key}.

5 Security property determination module

5.1 Modular plans for secure service composition (CNR)

1. Description

Web services behave differently according to the way they compose at runtime. Whenever a request is performed, the actual server must be chosen among a list of concurrent providers. A fundamental criterion for deciding which interactions are allowed is the respect of the security policies defined by the services.

Security policies are defined and included in the structure of a service at development time. Clearly, each service has its own security requirements and is usually unaware about the others’ policies.

Our technique aims at verifying which compositions are valid for a service. In this way services can compose dynamically only using safe interactions. This result is obtained by model checking the formal descriptions of the services, i.e. their contracts, against the involved security policies.
Moreover, through a partial evaluation step, we produce special requirements that the services expose to their clients in order to obtain a faster verification process. In this way we also improve the standard security-by-contract (SxC) technique with a policy negotiation phase that extends the security requirements of a service over its clients.

2. Detailed description of basic operation

When a service is activated it extends the network with one or more new functionalities. The state of the network, i.e. the services composing it, is then used to verify which compositions are valid for the new entity. Indeed, each service is registered in a public repository, e.g. a UDDI. Each entry describes the interface, namely the invocation signature, and the behaviour, i.e. the contract, of a service.

The new service interacts with the rest of the network by invoking some of the already existing services through service requests. A request is done by specifying the interface of the needed service. We extract from the repository the list containing all the compatible services and we verify which of them can be safely invoked. To do that, we match the contracts of the services with the security policies holding on the invocation. Once the service has been integrated and published in the repository it becomes available to the clients, including other services.

When a service is invoked its execution can perform some security relevant action entailing the actions already fired by the execution of its client. Hence, if the service poses a security requirement on its clients, we must guarantee that this composition is safe. This is obtained by checking whether the history of the client satisfies the security requirement.

Since requirements may depend on the underlying network behaviour, they cannot be specified ad development time. Our approach uses a partial evaluation algorithm to automatically generate security requirements starting from a standard security policy. This process is performed when a service joins the network, or when the repository is refreshed, reducing the costs of verifying whether an invocation is safe.

3. An example

Imagine a simple service network for travel booking where services are organized in a hierarchical way such that the services of a certain level can only invoke services from the above levels. The top level services are travel agencies (TAs). The TA (we assume to have only one instance in the network) books both flights and hotels. Booking is done through proper, middle level services. Each booking service (B) can reserve a single resource, i.e. flights or hotels, so that we have dedicated booking services for hotels (BH) and flights (BF). The booking process also involves a payment step. Payments are performed by other, bottom level services (Ps).

![Figure 48. An example of a service network for travel booking](image)

Customers contact a travel agency providing a credit card number for payments and receive back a receipt. Every execution of the travel agency books exactly one room and one flight by making two requests to the booking services. Note that, since every instance of booking service implements the interface B, the two requests are syntactically undistinguishable. Each booking service receives a card number and uses it for paying a reservation. Payment services are in charge for authorizing a purchase.
A payment service charges the needed amount on the credit card (possibly after checking the credentials of the account).

Each service has its own security requirements expressed through a corresponding security automaton. For example, a booking service would like to “perform at least one availability check (action check) before each payment (action charge)”. This property can be modeled with a 3-state automaton where every action but check and charge is allowed without changing the security state. After a check, any action is permitted, including charging. Instead, a charge action performed before a check leads to a security violation (denoted by a special, offending state). Additionally, a booking service can be forbidden to redundantly check, that is performing the check action more than once per execution. Similarly, the travel agency can declare different rules focusing on different aspects, e.g. “never book (action book) two times the same resource (hotel or flight)”. Clients are unspecified, so the network is open. However, clients do not affect at all the security policies introduced so far. We can therefore check secure this open network. As a matter of fact, services only put security constraints on their own traces and on those of the services they invoke. In particular, we can verify that the travel agency requests to booking services are effectively dispatched in the correct way, which is one for booking a room and one for the flight. In this way the correct behaviour of the service is guaranteed even if the underlying structure of the network is unknown to the service developer.

5.2 A Framework for Quantitative Security Analysis of Complex Business Systems (CNR)

1. Description

The framework considers a three-layered model of an enterprise. Indicators for each atomic element on each layer of the model are combined according to the structure of the enterprise forming an overall indicator for a specific business process. Such aggregation provides an objective way of aggregating indicators. Also such model specifies the sources of data required for the analysis and indicates the stakeholders (e.g., administrators) who have the deepest knowledge in specific parts of the system and who should be the provider of required data.

The framework takes into consideration that the flow of business process affects aggregation of security indicators and includes an explicit way of analysing security of the business process structure. The framework also supports outsourcing and allows correcting values according to the belief of client in genuine fulfilment of the agreed requirements by the contractor.

On the level of enterprise and business process we can specify where alternatives are possible: some applications may be run on various servers or different sequence of steps can fulfil the same part of the business process. The proposed analysis allows efficient comparison of alternative designs and selects the design for the service which has the best protection, i.e., has the best assurance indicator. All design alternatives are evaluated at once by the efficient algorithm created for this concrete task.

Next to the core analysis which works with one simple security indicator we propose others which make the analysis more practical. First improvement deals with several assurance indicators at once. The second modification allows conducting the analysis when it is hard to find a precise value of indicator and fuzzy analysis is more suitable.

2. Detailed description of basic operation

In the sequel, we will focus on structure of the business process and omit the part about the structure of enterprise as less relevant for Aniketos project.

An analyst models the business process with Business Process Modeling Notation (BPMN) in a hierarchical way (every tier contains only one structural activity (flow, sequence, parallel, and loop) or an outsourcing relation. The only difference with BPMN is that we add a new gateway in the notation. We a bit extended the notation to capture design or deployment alternatives to fulfil an activity (sub-processes) which accomplish the same functional goals but have different qualities. We added a
special construct to the notation to model the alternatives (a gateway with letter “D” inside). An example of such model is shown in Figure 49. At the end of modelling only one of the alternatives should be left and the final model will follow only the original BPMN.

The model is automatically transformed to a hypergraph used for further analysis (see Figure 2). Nodes of the hypergraph correspond to the appraisals for sub-processes and atomic activities of business processes. Hyperedges are decomposition edges indicating the lower sub-processes and activities required for operation of the higher sub-processes. Every hyperedge connects only the activities combined with one structural activity. Every alternative has its own hyperedge.

As an indicator for our analysis we use rate of occurrences. This indicator is an essential part of risk analysis technique which is widely used for security assessment. Values of the indicators for atomic activities can be determined by analysis of corresponding history of events. The rate of occurrences for the whole process is unknown.

Since activities are often used for other processes we should consider only the impact on the process in question. Therefore, we need to add weights which are found by counting relative time of execution of lower sub-processes. For example, if for a sequence structural activity connecting two sub-processes we know that the first sub-process requires 70% of time (and then switches to another process) and the second one requires only 30%, then the weights will be 0.7 and 0.3. The weights for outsourcing hyperedges are determined from the trust level assigned to a partner \( w=1/T_p \), where \( T_p \) is a trust level of partner \( p \). The idea here is the less we trust a partner the more we thing that the agreed level of service will be satisfied and adjust the expected indicators correspondingly.

![Figure 49. The loan origination process modelled with BPMN](image)
When premise values for atomic activities and weights are assigned we can compute the value of the indicator for the whole process. Moreover, the algorithm we propose selects only the alternatives which provide more secure service (the “shortest” path). Thus, the more secure structure of the business process can be selected. Furthermore, small changes in the structure of the business process (e.g., change of premise values or connection with a partner) can be efficiently taken into account and new structure can be selected on the fly.

The core analysis uses only one indicator for aggregation. A more realistic situation is when several indicators are required (e.g., risk is usually computed as a sum of losses due to a number of threats). We also adapted our core algorithm for contemporary analysis with several indicators. Pareto-optimality principle has been used for selection of the most secure alternative. This principle allows using only a partial order for comparing alternative designs of the business process. Therefore, at the end of the analysis we still have several alternatives. The selection of the most secure design is performed by applying Annualised Loss Expectancy (ALE) analysis.

The proposed technique is supported by a home-made tool, which 1) creates a hypergraph using a BPMN model created with BPVA 2.4. 2) helps to assign required values (premise indicators and weights) 3) conducts three types of analysis. Though the tool is not mature enough to be used by an inexperienced user, but it can be improved if required.

3. An example

Consider a bank holding company which outsources some services to several semi-independent subsidiaries. There is a specific subsidiary (BBB bank) belonging to the holding which provides some of these services. There are also several independent organisations which provide specific services. For example, in our scenario two Credit Bureaus collect financial information about people (e.g., loans taken, money paid back, etc.) from various sources (e.g., banks). This information is then aggregated and trustworthiness rating is computed for each person. The rating can be used by banks which need the information for performing some operations (e.g., issue loans).

The holding company is aware of a huge number of losses caused by frauds. In the banking sector a typical problem is asset misappropriation: the usage of organisation’s assets (especially cash) by an
employee either directly or indirectly for the employee’s own benefit. Asset misappropriation accounted for approximately 90% of all frauds. Therefore the holding wants to be sure that it is well protected against this type of losses. Knowing that total elimination of threats is impossible the holding company requires that the subsidiary should have not more than 5 frauds in 10 years.

Figure 51. The hypergraph for loan origination process with weighs

Figure 52. Premise values

Suppose weights and premise values are as shown in Figure 51 and Figure 52. Now we can aggregate indicators using the weighting function when one hyperedge combines appraisals and select the most secure alternative if two or more disjoint hyperedges lead to one node.

For example, selecting the most suitable provider of trustworthiness rating we see that though Credit Bureau 1 guarantees a more secure service ($q_7=7$) we select Credit Bureau 2 ($q_8=8$) since it has a higher trust level ($1.1 < 1.4$). In this case we have: $w_7q_7 > w_8q_8$ ($7*1.4 > 8*1.1$). Thus $q_6 = 8.8$.

We also follow the same strategy selecting the most secure way of payment. Payment by automatically withdrawal is the most secure among three alternatives: $13*1 > (6*0.5 + 13*0.3) > 6*1$. Thus $q_{10} = 6$.

The following aggregation requires only applying the weighting function. For example, $q_9 = 10*0.3 + 8*0.7 = 7.4$. Continuing computation we can find that expected amount of asset misappropriations in 10 is going to be 3.055 which is less than required 5 cases in 10 years. Now, we can compute the losses caused by asset misappropriation threats in loan origination process using ALE analysis. Assume that every threat occurrence results in cost 100 000 euro. Then the expected amount of losses in 10 years is 305 500 euro.

5.3 Platform for Run-time reconfigurability of security (ATOS)

1. Detailed description of basic operation

This asset is a system and platform that provides applications and systems with security solutions during run time, upon request. PRRS supplies the necessary and proper security solution to applications, at any given time, being aware of the changes in the context and environmental attributes of the execution.

This dynamic provision of Security and Dependability is particularly focused in Internet of Things ecosystems scenarios. It is not limited to run-time contextual monitoring, it also performs design-time management of the mechanisms and constraints required in each context. To automatically apply these mechanisms at run-time, PRSS has precise specifications of how to adapt them to the constraints and monitor their execution.

PRRS implements the SERENITY concepts of S&D patterns and mechanisms.

2. An example

The SRF Console is the interface for the S&D authority to control the run-time execution and to manage the configuration of the tools. Through this interface, end-users can express their requirements and monitor the adaptation of the mechanisms in their execution.

The interface is composed of 3 parts:

- A tree view of the S&D classes, patterns and implementations available to the system in the Library
- The configurations, rules, preferences and events management of the services
- The monitoring of the execution

The following pictures, taken from the PRSS documentation in the SERENITY website, show the Console Overview, the Monitoring of the Service List and Monitoring Logs.
Figure 53. Console Overview

Figure 54. Monitoring the Service List
5.4 STACS / Scalability Testbed for Access Control performance of Services (TSSG)

1. Detailed description of basic operation

STACS is a client-server application. The client component submits access requests that are routed to the server component which incorporates the Policy Decision Point under test. The start and end times of policy processing are recorded. Subsequently, other tools (external to STACS) can be used for offline statistical analysis to build simulation models to test various scenarios, such as changing the mix of access requests, etc.

2. An example

Assume a traveller wishes to employ a geolocation data service and to share her location data with selected family members and friends. Such a user might use Facebook to manage their social network – assigning people to groups with different access privileges and other characteristics. They might also wish to mashup this location data with other social media, notably blog posts and/or Twitter updates. Clearly, they will need to decide on who can access their data, with what frequency etc. Such access control rules can be encoded (in the background) in a policy language by the service(s) they use. The services will need scalable policy based access control, given the large numbers of users and frequency of location updates.

The service provider would need to test the scalability of the (fundamental) access control services they provide, particularly since timely location data is what is needed. Access control is an important part of their service offering, given recent adverse publicity regarding the privacy of personal data in services such as Facebook. Furthermore scalability might depend on the performance characteristics of several service providers. All of these requirements present performance challenges, putting scalability in doubt.

STACS would support offline experimentation to ascertain the scalability of service providers’ access control systems. Service providers would provide (mockups of) critical components of their infrastructure for STACS to collect timing measurements and simulate scenarios of varying complexity, and thereby to identify and manage service bottlenecks.
5.5 Composing Security-Aware Software

1. Detailed description of basic operation

The framework’s main objective is to generate computational reflection to let components and their developers identify and capture the various security properties of the other components with which they cooperate. Security properties and behaviours of a software system are categorized into 11 classes in ISO/IEC-15408 Common Criteria. These classes are made of members, called families, based on a set of security requirements. To demonstrate the idea, the framework considered only a subset of these security classes i.e. user data protection. The publishable security properties related to user data protection of any atomic component can be categorized as required—a precondition that other interested parties must satisfy during development to access the ensured security services—or ensured—a post-condition that guarantees the security services once the precondition is met. Security properties are typically derived from security functions (the implementation of security policies). And the security policies are defined to withstand security threats and risks. A simple security function consists of one or more principals (a principal can be a human, a component, or another application system), a resource such as data, security attributes such as key or passwords, and security operations such encryption. Based on these, three main elements characterize an ensured or required security property: security operations executed by the components to enforce security properties, security attributes required to perform the operation, and application data manipulated in a compositional contract. Furthermore, these elements are formulated into a simple structure to characterize the security requirements and assurance of individual components:

\[ f (O_i, K_j, D_k) \]

Where \( f \) represents a security objective formed with three associated arguments; \( O \) is the security-related operation performed by the principal \( i \) in a compositional contract; \( K \) is a set of security attributes used by the principal; subscript \( j \) contains additional information about \( K \) such as key type, the key’s owner, and so on; \( D \) is an arbitrary set of data or information that is affected by the operation \( O \); and the subscript \( k \) contains additional information regarding \( D \) such as whether a digital signature is used or not. The following examples represent a required security property \( RP \) (protect_in_data) and an ensured security property \( EP \) (protect_out_data) of a component \( P \):

\[ RP = \text{protect\_in\_data} (\text{encrypt}Q, \text{key}P+, \text{‘amount’}) \]

\[ EP = \text{protect\_out\_data} (\text{encrypt}P, \text{key}Q+, \text{file1}P−.\text{digi\_sign}) \]

In this example, component \( P \)’s required property \( RP \) states that the data is to be encrypted by any component \( Q \) with component \( P \)’s public key. A plus sign (+) after \( P \) denotes public key. The ensured property \( EP \) states that component \( P \) encrypts the data file with the public key of any component \( Q \). The data is also digitally signed by \( P \) with its private key, denoted by the minus sign (−) after \( P \). This format is specific to a particular type of security function related to user data protection. This notation, or a similar one, can be standardized for all components. However, alternative structure might need to be formulated to represent other security classes such as authentication, security audit, trusted path, privacy, and so on.

The methodology also establishes a global security characterization of a composite system. It specifies the composite component security by required and ensured properties based on the structure and principles defined for the autonomic components. The final and important issue in the framework is
how to make components’ security profiles available to other components. It is based on a three-phase automatic negotiation model for component composition:

A component publishes its security properties attached with functionality to the external world. The component negotiates for a possible component security concerns at runtime with other interested candidate components. If it succeeds, the negotiation results are used to configure and reconfigure the composition dynamically.

2. An example

The framework considered a fictitious distributed-system topology as an example of how our proposed active interface would work in a distributed environment. Consider an e-health care system that regards all clinical information passing among the stakeholders, such as the general practitioners, specialists, patients, and pharmacists, as confidential. Assume a focal component \(Y\) running on a machine at a GP’s office connects with a trusted candidate component \(S\) chosen from among many such systems running at various specialists’ offices. \(Y\) provides a patient’s diagnosis report to \(S\) to get a prescription. After receiving the prescription from \(S\), \(Y\) sends it electronically to a candidate component \(P\) residing on a pharmacist’s system for a price quotation. Developers would independently develop many such \(P\)s and \(S\)s and make them available from their various distributed sources, potentially able to deliver the functionality that \(Y\) wants. However, component \(Y\) not only is interested in specific functionality but also wants to know upfront the security properties that those components provide.

Assume component \(Y\) exposes the following required and ensured security properties:

```
SECURITY {
    REQUIRED\(Y\) \(\{ R_Y = \)
    protect_in_data (encrypt\(S\),
    key\(S^-\), ‘prescription’\\.digi\\_sign))
    ENSURED\(Y\) \(\{ E_Y = protect_out_data\)
              (encrypt\(Y\), key\(S^+\), ‘diagnosis’)).
```

The ensured property states that \(Y\) will provide a diagnosis report of a patient to a specialist component. \(Y\) would encrypt (encrypt\(Y\)) the report with \(S\)’s public key (key\(S^+\)). In return, \(Y\) requires from \(S\) that \(S\) digitally sign (‘prescription’\\.digi\\_sign) and encrypt (encrypt\(S\)) the prescription it sends with its own private key (key\(S^-\)). Now assume that in response to the event \(Y\) broadcasts for the functionality Get\\_prescription, it receives responses from components \(S1\) and \(S2\) offering that functionality. \(S1\) and \(S2\) run on different machines for different specialists; they are independently developed and serviced by different developers and have their own security requirements and assurances. \(Y\) also reads the certification information, origin, and identity of the components from the interfaces of \(S1\) and \(S2\). \(Y\) first queries \(S1\). \(S1\)’s interface exposes its security properties.
SECURITY {
  REQUIRED$_{S_1}$ $(R_{S_1} =$
  protect\_in\_data $(\text{encrypt$_{Y}$,}$
  key$_{S_1}$, $\text{\textquoteleft diagnosis\textquoteright})$)
  ENSURED$_{S_1}$ $(E_{S_1} =$
  protect\_out\_data $(\text{encrypt$_{S_1}$,}$
  key$_{Y}$, $\text{\textquoteleft prescription\textquoteright})$).
}

According to these security properties, component $S_1$ requires that component $Y$ encrypt the diagnosis report with $S_1$’s public key. In return, $S_1$ would encrypt the prescription with $Y$’s public key, but $S_1$ would not digitally sign the prescription data. $Y$’s active interface now generates the component security concern (CSC) between $Y$ and $S_1$ based on

$$C_{Y_{S_1}} = (E_Y \Rightarrow E_{S_1}) \land (E_{S_1} \Rightarrow E_Y)$$

In the generated CSC, $S_1$’s ensured security property has not fully satisfied $Y$’s required security property, because $S_1$ does not provide the digital signature with the prescription, as $Y$ requires. After making a similar query to $S_2$, $Y$ reads $S_2$’s disclosed security properties as

SECURITY {
  REQUIRED$_{S_2}$ $(R_{S_2} =$
  protect\_in\_data $(\text{encrypt$_{Y}$,}$
  key$_{S_2}$, $\text{\textquoteleft diagnosis\textquoteright})$)
  ENSURED$_{S_2}$ $(E_{S_2} =$
  protect\_out\_data $(\text{encrypt$_{S_2}$,}$
  key$_{S_2}$, $\text{\textquoteleft prescription\textquoteright,}$
  s2.digi\_sign))
}

Component $S_2$ requires $Y$ to encrypt the diagnosis report with $S_2$’s public key. In return, $S_2$ ensures that it would digitally sign (‘prescription’s2.digi\_sign) and encrypt the prescription with its private key. $Y$ can decrypt the message using $S_2$’s public key to verify the signature. Based on these security properties, the generated CSC is consistent with the requirements of $Y$ and $S_2$. $Y$ can finally be combined with $S_2$.

5.6 Enforcing Secure Service Composition

1. Description

Enforcing Secure Service Composition (ESSC) is proposed to study secure composition of software. It extends the $\lambda$-calculus with primitives for invoking services that respect given security requirements. Security-critical code is enclosed in policy framings which enforce safety and liveness properties of execution histories. The actual histories that can occur at runtime are over-approximated by a type and effect system. These approximations are model-checked to verify policy framings within their scopes. This allows for removing any runtime execution monitor, and for selecting those services that match the security requirements.
2. Detailed description of basic operation

ESSC tackles the problem of modelling composition of services in the presence of security constraints. A security policy in ESSC is a property over a statically determined abstraction of the behaviour of a service. Safety and liveness properties have shown effective to reason about concurrent systems and security. For example, history-based access control can be modelled in terms of safety properties, while liveness properties can be exploited to formalize denial-of-service and brute-force attacks of cryptographic keys. More generally, a suitable combination of safety and liveness properties can express contract agreements in terms of enforce and guarantee paradigm.

ESSC introduces a typed extension of the λ-calculus to describe services as program expressions, and to compose them under security constraints. The calculus assumes a set of primitive access events, that abstract from activities with possible security concerns. The security policies are regular properties of execution histories (i.e. sequences of access events). Given an expression e, a safety framing φ[e] enforces the policy φ at each step of the execution of e. A liveness framing ψ[e] prescribes that the evaluation of e must eventually respect the policy ψ. ESSC exploits a static analysis technique to ensure the desired behaviour, checking both kinds of properties at compile time. Note that, while safety properties can be enforced by an execution monitor, liveness properties cannot. Also, liveness cannot be reduced to safety in general.

The static analysis over-approximates program behaviour through history expressions. These represent sequences of access events together with the scope of safety and liveness framings. A history expression is valid when all the histories it denotes respect the security policies – according to the scopes and the intended semantics of safety and liveness framings.

Services are modelled as expressions with a functional type \( \tau \xrightarrow{H} \tau' \). Intuitively, when supplied with an argument of type \( \tau \), the service evaluates to a value of type \( \tau' \), and it generates a history belonging to the statically inferred history expression H.

A service invocation is modelled by an expression requirement \( \tau \xrightarrow{\phi[\psi][e]} \tau' \). It means that ESSC will look for a service of type \( \tau \xrightarrow{H} \tau' \), where H satisfies both the safety framing enforcing \( \phi \) and the liveness framing guaranteeing \( \psi \). Intuitively, the liveness framing \( \psi[] \) can be seen as the duties the invoked service must fulfill. Instead, the safety framing \( \phi[\cdot] \) says how the caller protects itself from the service.

A typed service \( e : \tau \xrightarrow{H} \tau' \) is assumed to be published in a trusted repository, which collects the type, in particular guaranteeing that H represents all the possible behaviour of e.

The second technical contribution of ESSC is the definition of a type and effect system that extracts from a well-typed program a sound approximation of its possible runtime histories, represented by a history expression. Remarkably enough, ESSC exploits the information about types and effects in order to detect and select those services only, that match the security constraints required by the service invocation. In this way ESSC models the fulfillment of a contract agreement. The type and effect system enjoys the following type safety property: if the history expression of a program is valid, then there will be no runtime errors. Also, only the services that respect the security properties required will be chosen. Therefore, execution monitoring is needed no longer.

The third contribution of ESSC is to develop a static technique to verify validity of history expressions. This is done by model checking Basic Process Algebras (BPAs) with Büchi automata. A history expression is rendered as a BPA, while an automaton models the security properties subject to the scope of safety and liveness framings.

Summing up, a user can invoke services and put over them constraints that enforce and guarantee security. The type and effect system predicts the actual behaviour of programs, including the security framings they must respect. Validity of behaviour is model checked over BPAs and Büchi automata. If the effect of a program \( e \) is proved valid, then \( e \) can be executed with no runtime monitoring and it will never go wrong.
6 Security verification module

6.1 Usage Control Authorization System (CNR)

1. Detailed description of basic operation

Usage Control Model

The Usage Control model (UCON) is a model that encompasses and extends the existing access control models. Its main novelties are that rights are dynamic, and that subjects' and objects' attributes can be mutable over time, thus requiring continuous enforcement of the security policy during the access time. The UCON core components: subjects, objects, attributes, authorizations, conditions, obligations.

Subjects and Objects: the subject is the entity that performs the accesses on objects. An object, instead, is an entity that is accessed by subjects through access operations.

Attributes: describe subjects' and objects' features. An attribute is mutable when its value is updated as a consequence of accesses performed by subjects on objects, e.g., the user's reputation. Mutable attributes can be updated before, during, or after the execution of the access action. Immutable attributes instead, are the classical attributes, that can be updated only through an administrative action, e.g., the user's identity.

Authorizations: are predicates that evaluate subjects' and objects' attributes to decide whether to allow subjects to perform actions on objects. The evaluation of the authorization predicate can be performed before executing the action, or while the action is in progress.

Conditions: are environmental or system decision factors that do not depend upon subjects or objects. The evaluation of conditions can be executed before or during the action.

Obligations: are used to verify whether the subject has satisfied some mandatory requirements before performing an action, or whether the subject continuously satisfies these requirements while performing the access.

The mutability of subjects' and objects' attributes introduces the necessity to execute the usage decision process continuously in time because, while the access is in progress, the attribute values that previously authorized the access could have been changed in a way such that the access right does not hold any more. In this case, the access is revoked.

Security Policy Language

The language we use to express security policies, POLPA, is operational, and it is based on a process algebra. This language represents the allowed behaviour of users by defining the sequences of actions that are allowed on the system, and which authorizations, conditions and obligations must hold before, during and after the execution of each action. To define the sequence of action, POLPA exploits a set of operators. For example, the sequential operator allows to execute an action B only after another action A, the parallel operator allows to execute two actions A and B in any order, and the alternative operator allows to execute either an action A or an action B.

The approach we used to define the sets of actions that model UCON is inspired to Zhang et al. (2005). POLPA defines two distinct sets of actions: security relevant actions and control actions. Security relevant actions are the ones performed by the subject that is controlled by the authorization system. These actions are intercepted by the authorization system, and the decision process is performed to decide whether they can be actually executed or not. For example, let us suppose that the authorization system protects the file system of a machine. In this case, the security relevant actions are represented by the accesses of the users to the files stored in the file system, to perform operations such as open, read, write, append or close. Control actions, instead, are actions that are executed by the authorization system itself, as consequence of the decision process. An example of control action is the interruption of the execution of an access that is in progress.
Martinelli et al. (2010), describes in detail the POLPA language and shows that the expressiveness of POLPA is sufficient to model the basic features of UCON model.

Architecture
The architecture of the system is based on a Policy Enforcement Point (PEP) and a Policy Decision Point (PDP), such as most of the common authorization systems.

![Figure 56. Architecture of Usage Control Authorization System](image)

The PEP should be integrated in the component to be monitored, and should be able to intercept the invocations of security relevant operations performed by the user, to avoid their execution, and to interrupt them while in progress.

The tryaccess(s,o,r) command is transmitted by the PEP to the PDP when the security relevant operation is intercepted, where s represents the user, o the object that is accessed and r the action that is performed by the user on the object. The PEP allows the execution of the operation only after a positive response from the PDP, represented by the permitaccess(s,o,r) action. Moreover, once an access has been permitted and is in progress, the PEP should be able to detect when it terminates to issue the endaccess(s,o,r) command to the PDP. The specific components of the system architecture where the PEP should be integrated depend on the resources that should be monitored and on the set of security relevant accesses we are interested in.

The PDP is the component of the authorization system that performs the usage decision process. The PDP, at first, gets the security policy from a repository, and builds its internal data structures for the policy representation. The policy is expressed with the language previously described. The PDP is invoked by the PEP, that sends it the tryaccess(s,o,r) command every time that the subject s attempts to access a resource o. The PDP exploits its internal representation of the policy to determine whether the access is allowed or not. If the policy is satisfied and the permitaccess(s,o,r) action is sent to the PEP, that executes the access. Otherwise, the PDP returns denyaccess(s,o,r) to the PEP, that enforces it by skipping the execution of the access. The PDP is also invoked by the PEP every time that an access that was in progress terminates, with the endaccess(s,o,r) action. However, the PDP is always active, because if required by the policy, the PDP continuously evaluates a set of given authorizations, conditions and obligations while an access is in progress, and it could invoke the PEP to terminate it through the revokeaccess(s,o,r) action.

The other components of the architecture are the managers for attributes, conditions and obligations, that are exploited by the PDP for performing the policy evaluation process.

2. An example
Let us consider the following policy that is applied by a web server and allows a user to access the web site www.siteA.it, and after this access the same user can also access another web site, www.siteB.it, that is hosted on the same system. Supposing that get_site("www.siteA.it") is the action performed by users to access the web site www.siteA.it, the policy is the following:
This policy, to combine actions, exploits only the sequential operator, that is represented by the dot (.) symbol, where A.B means that the action B can be executed only after the action A. Moreover, each access is represented by the triple \((s,o,a)\), where \(s\) is the subject that perform the action, \(o\) is the object that is accessed and \(a\) is the action executed on the object \(o\).

In this policy, lines 1 to 4 allow a user to access the network resource to perform a \(\text{get\_site(site)}\) action when the value of the parameter site is equal to \(\text{www.siteA.it}\). In particular, line 1 of the policy includes the command \(\text{tryaccess(fuser, net, get\_site(site))}\), that represents the request received by a user, whose identifier is saved by the variable fuser, to execute the operation \(\text{get\_site(site)}\). This access is permitted by line 3 of the policy, i.e. by the command \(\text{permitaccess(fuser, net, get\_site(site))}\), only if the predicate in line 2, \((\text{site} = \text{"www.siteA.it"})\), is satisfied. As a matter of fact, the command in line 3, that permits the access, can be executed only if the condition in line 2 is verified. When the access is terminated, the control action \(\text{endaccess(fuser, net, get\_site(site))}\) (line 4 of the policy) is received.

Lines 5 to 8 are very similar to lines 1 to 4, and allow a user to perform a \(\text{get\_site(site)}\) action when the value of the parameter site is equal to \(\text{www.siteB.it}\). However, the execution of this second access is permitted only after that the access to \(\text{www.siteA.it}\) is terminated, because the lines of the policy related to \(\text{www.siteB.it}\) are placed in the sequence after the command in line 4 that represents the termination of the access to \(\text{www.siteA.it}\).

3. Additional references


6.2 MATTS (Mobile Agent Topology Test System) (LJMU)

1. Description

MATTS is a program that has been designed primarily to allow modelling of systems-of-systems scenarios and testing of secure component composition analysis techniques. The program integrates together a number of tools:

A secure, sandboxed, virtual machine.
A mobile code framework allowing communication between agents.
A composition analysis engine.
A formal analysis tool.
MATTS allows scenarios to be created interactively and comprised of a mixture of both real and virtual nodes or services. Separate software (called the CompositionClient) has been developed for use on mobile devices that, when deployed, allows mobile devices to appear in the modelled visualisation. As the scenario evolves and changes, the properties and connections between services are dynamically analysed to determine the security properties of the overall composition of services. The analysis process is based on a scripting technique.

2. Detailed description of basic operation

The MATTS server acts as the central analysis tool and provides a visualisation of the network structure. We can split its functionality into a number of categories, the most important of which are scenario visualisation, security/composition analysis and networking functions.

Figure 57 shows the basic interface. This allows services to be added, either as ‘virtual nodes’ or real devices. Each of these shows up in the same way on the interface.

The analysis process does not distinguish between modelled and real nodes/services, hence the tool provides a suitable environment for modelling and testing secure composition analysis techniques.

Figure 58 shows the interface using a visualisation based on the logical network interactions between nodes.
Figure 58. Scenario visualisation showing logical network structure.

The CompositionClient is another part of the MATTS system that runs on other devices (PCs, laptops or Windows Mobile/Pocket PC devices including phones and PDAs) and allows them to interface with the MATTS server via the network. It allows for the properties of a device to be assigned, for connections to be made to the server, and then for the execution of certain network operations such as sending data between nodes (either real or modelled).

CompositionClient nodes also appear in the MATTS interface and can be manipulated in the same way as the modelled services. The properties of both the modelled and real nodes are incorporated into the analysis process. Figure 59 shows part of the CompositionClient interface as it appears on a PC and a mobile device.

Figure 60 provides a very brief overview of the property analysis process. This is based on a special scripting language that allows the incorporation of both service properties and the interactions between services to be considered.
3. An example

Access control provides an important security mechanism for preventing unauthorised access to data and retaining data privacy. It can, however, present problems for interoperating systems, since access control mechanisms may not be available for all services and even when available, may not provide uniform access control methods to support interoperation. An access control mechanism that can be layered on top of existing services may therefore provide an effective way to impose access control without the need to significantly modify underlying systems.

Moreover, access control mechanisms often fail to account for potential data leakage that can occur as information is passed between distributed, interoperating systems. If an authorised service passes private data to a second receiving service on a different platform, how can the sending service be certain that the receiving service will not pass the data onwards to a further unauthorised actor, even if
D1.1 Consolidation of existing work

The receiving service is itself authorised to have access to the data? Using MATTS we can easily enforce access control whilst simultaneously preventing data leakage of the nature just described. We achieve this by directly analysing federated services based on their dependencies. Each file has access control meta-data associated with it that stipulates the level of access required in order to read from or write to the file. These levels may be set by the creator of the file or their agent and may take the form of an access control list as with discretionary access control, or an access level as with mandatory access control. The origination of this data is independent of the process described here, and may take any of these forms. In addition, each service must also have a stipulated set of access rights. Again, the standard method for this is for the access level or rights of a service to be set as the same as those of the instigator of the service. In other words, if the service is spawned by another service, it inherits the same access rights as its parent; if it is started by a user, it assumes the same access rights as that user. Again, the exact process is independent, the importance lies in the requirement that all files and services have access levels or rights assigned to them.

The primary aim is to ensure that read access to data from a file is granted only to those services with sufficient access rights, and similarly for write access.

The benefit provided by the dependency information is that data flow across services can be predetermined, and a model constructed of the data flow throughout the federation of services. This model can be established before service deployment, and re-analysis can take place whenever dependencies change.

A simple MATTS script describes the requirement that data may only flow into or out of a component if that service has sufficient access rights to read from or write to the originating or destination file respectively.

In addition to the dependencies between interoperating services, the MATTS script can also utilise slicing properties of components that allow internal dependencies between inputs and outputs to be established. Space constraints prevent a full explanation of this, but the consequence is that the access restrictions that would otherwise be imposed on interoperating systems may on a number of occasions be safely relaxed without affecting the overall security of the system.

As an example, consider a number of services \( S_1, \ldots, S_6 \) that interoperate via distributed middleware as shown in Figure 61. The dependencies (shown as solid arrows) between services can be established through instrumentation and analysed to ensure there are no access control violations prior to deployment. We assume that different access rights are required in order to access the data contained in files \( A \) and \( B \). The result of the analysis will tell us that service \( S_6 \) must have sufficient rights to access file \( A \); service \( S_5 \) must have sufficient rights for access to file \( B \); services \( S_3 \) and \( S_4 \) must have right to access both files \( A \) and \( B \); whilst the read access rights of services \( S_1 \) and \( S_2 \) do not matter, since no data is read into the service. Note the internal dependencies are marked with dotted arrows in the figure.

The framework described here is able to automatically establish these results and, based on the access rights information supplied about each component and file, establish whether any access violations or data leakage is likely to occur after deployment.

Timing results for an analysis of a simple linear configuration of services is shown in Figure 62. In this case, with simple dependencies between services, we can see that analysis time is linear in the number of components. These timings represent times for pure analysis and do not take into account delays during instrumentation readings or caused by network overheads. Nonetheless these results are
encouraging, since analysing the services as a complete system using slicing would be liable to result in an exponential time process in the general case.

6.3 Avantssar platform for validating trust and security properties (SAP)

1. Detailed description of basic operation

Exposing services in future network infrastructures entails a wide range of trust and security issues. Solving them is extremely difficult since making the service components trustworthy is not sufficient: composing services leads to new and dangerous vulnerabilities due to interference between component services and policies, the shared communication layer, and application functionality. Thus, even when individual services themselves are secure, the validation of their composition into secure service architectures is of vital importance.

The AVANTSSAR project developed a tool-supported methodology for the formal specification and the automated validation of trust and security in service-oriented systems. Within the AVANTSSAR platform (see Figure 1), service compositions are specified using in ASLan. ASLan is a formal language, developed within the AVANTSSAR project, for specifying trust and security properties of services, their associated policies, and their composition into service architectures. The formal tools suite (e.g., OFMC, SATMC, CL-Atse; for details see http://www.avantssar.eu) allows for statically analyzing the specified service composition for potential security (secrecy, privacy) problems.
2. An example
The interested reader is referred to the basic examples provided online as part of the AVANTSSAR Validation platform: [http://regis.sci.univr.it/avantssar_platform/output.php](http://regis.sci.univr.it/avantssar_platform/output.php) and in particular the “Deliverable D2.2 ASLan v.2 with static service and policy composition” (available at [http://www.avantssar.eu/pdf/deliverables/avantssar-d2-2.pdf](http://www.avantssar.eu/pdf/deliverables/avantssar-d2-2.pdf)) of the AVANTSSAR project.

6.4 Static Verification Framework Tool

1. Description
The main scope of the Static Verification Framework Tool (SVF) is to support the design and analysis of abstract behavioural system specification with respect to key security attributes.

The overall aim is broken down into the following objectives:

- To provide support for the construction of design models representing abstract behavioural system specification
- To provide support for the construction of abuse cases
- To provide support for the specification of security and general application properties to be verified against the abstract behavioural system specification
- To statically verify system specifications against security and general application properties
- To visualize the results of the static verification process

Following from normal software development process, the view is that the Static Verification Framework interacts with the Design and Architecture Framework, since the former will use the requirements specifications, architecture models, and high-level design models generated by the Design and Architecture Framework, in order to support the construction of the abstract behavioural specifications and properties to be verified against these specifications.

2. Detailed description of basic operation
The following figure depicts the architecture of the SVF tool.

![Figure 64. Architecture of the SVF tool](Image)
The SVF Tool is part of the Development Framework together with the Design and Architecture Framework of the PEPERS Framework. Following from normal software development process, the view is that the Static Verification Framework interacts with the Design and Architecture Framework since the former will use the requirements specifications, architecture models, and high-level design models generated by the Design and Architecture Framework in order to support the construction of the abstract behavioural specifications and properties to be verified against these specifications. The SVF Tool consists of the following components:

- The Design Model and Security Protocols Constructor component, which is responsible for the construction of abstract design models of the system and the specification of security protocols, through the use of UML and UMLSec
- The Abuse Case Constructor, which supports the definition of UML use cases and state machine diagrams representing scenarios of the attackers of the system.
- The Property Editor and Selector, which allows the user to build, browse and select the properties (specified using an extended version of linear temporal logic - LTL) to be verified by the SVF. These properties can be classified into two main groups: general properties, concerned with mobile peer-to-peer applications, and security properties, concerned with security requirements of the system.
- The Static Verification Tools, which perform model checking as the main verification technique. The model checking languages used are SPIN, as the tool to support the verification of design models of the system, and AVISPA to support the verification of security protocols. A Result Visualisation Tool is used to present the results of the verification process.
- The Translators, which support the mappings from UML design models into models for verification purposes.

3. An example

Sample Model: Instantiation and Simulation

Model Instantiation: Two Peer objects (a and b) and one RoleManager object (manager). The RoleManager assigns role Sender to a, and role Sender to b. The object with role Sender (a) sends a message containing variable value to the object with role Receiver (b).

Object "a" always reaches its final state "eventually state (a,Final)" but object "b" sometimes gets stuck at state Receive. This is due to a synchronisation problem at state "Branch" which is solved by adding the condition "Peer.counter=2" to both transitions leaving state "Branch".

**Property: Availability**

Readiness of a system to provide a correct service.

First formula (ok): After executing state "Branch" peer "a" should at some point invoke "message" on "b" - "always (state(a,Branch) implies eventually call(a,b,message))".

Second Formula (fails): Always object "b" reaches state "Receive" before object "a" reaches state "Send" - "always (state(a,Branch) implies eventually state(a,Send))".

**Property: Integrity**

Absence of unauthorised modification of information. Let’s assume that "a" cannot modify field "value" after being assigned its role.

First formula: "always (state(a,Branch) implies always (!write(b,value)))" fails because "b", with role Receiver, writes into field value.

Second formula: "always (state(a,Branch) implies always (!write(a,value)))" reports no errors.
7 Security policy monitoring module

7.1 Risk-aware Usage Decision Making in Highly Dynamic Systems (CNR)

1. Description

The main goal of any access control is to guarantee that a resource is used as it is stated in established policies and to prevent unauthorized users from accessing or corrupting the resource. A policy decision point entails the most complete, up-to-date and trustworthy information to make credible decisions. Obtaining such information is a difficult task since the mechanisms collecting authorization context are procedural, error-prone, rely on software and hardware, and, thus, have a risk of failure. Moreover, there is always a risk that the mechanisms are compromised by a malicious entity. Decisions made using low quality, vague, or obsolete information may result in major losses.

An authorization context received by a policy decision point can be imprecise because of two types of causes: unintentional and intentional ones. Unintentional causes appear because the system is imperfect and inherent risks are always present (e.g., delays, noise, loss of connection, etc.). Intentional causes are connected with deliberate alteration of authorization context by a malicious data provider. Note, that a malicious data provider may use unintentional causes to hide its actions.

All these drawbacks relevant for the access control model are even more relevant for its successor, the usage control model, where correctness of policies has to be checked not only during the first authorization, but also afterwards, i.e., during usage of a resource. We can check a policy only in discrete moments of time and there is no guarantee that the policy holds between adjacent checks. In highly dynamic systems this may lead to great losses. The reliability of controlling mechanisms is also limited by presence of design and implementation errors, vulnerabilities of execution environment, errors caused by users, etc. Thus, correct values of attributes may alter while the compromised controlling mechanisms do not detect this change.

We propose a basic risk-based approach which helps in making decision for usage control model when a number of uncertainties are present. We provide a probabilistic approach based on Markov chains to model mutability of authorization context. The approach can be used to solve different problems caused by presence of uncertainties. We employ risk analysis in order to make the most rational decision and be as flexible as possible.

2. Detailed description of basic operation

The proposed approach consists of the following steps:

1) For each attribute create a Markov chain for modelling changes of attribute values.
2) Compute the probability that the policy which uses the attribute is violated at some point of time.
3) Compare costs if further access is allowed or denied.
4) Apply a mitigation strategy to reduce the risk

The first step is to create a Markov chain, which indicates how a value of an attribute changes. States in this chain are possible values of the considered attribute and transitions are possible changes of the attribute. States can be combined if a fine-grained analysis is not required for simplicity.

We assume that transition probabilities do not change during some period of time (strict-sense stationary systems). Moreover, the change of these probabilities occurs slowly and infrequently. This assumption allows us to observe these values in the past and use them for prediction of nearest future.

In practice, transition probabilities can be estimated using statistics about past operations. The best way is to find the statistics for a specific partner and derive the probabilities about it directly. This approach is useful if we consider a long term interaction with a partner. Another way is to get the value from other sources (e.g., other data providers) or a central authority (similar to credit bureaus in banking). Finally, if interactions with partners are very short (e.g., web server and users) then similar users can be grouped and statistics can be collected for the whole group.
Now we compute the probability of failure of a policy. We need to know the amount of transitions in the period between the time when we know exact value of an attribute and the current time. In the simplest case we know the amount of completed transitions (e.g., the auction server knows the exact amount of accomplished deals). In this case we use discrete time markov chain. We also can apply the same theory to continuous attributes making an assumption that the average time between changes of attribute value exponentially distributed with the rate parameter $v$. This assumption allows modelling the behaviour of attribute values using a continuous-time Markov chain.

In both cases (using Discrete or Continuous-time markov chains) we use absorbing states to find the probability that a policy has been violated since the last check of the attribute. This probability is required for making a rational decision about the further access.

There are two possible decisions about a usage session: to continue or to revoke the session. The decision depends on the attributes received by a decision-point. There are four possible results:
1) continue usage when it should be continued;
2) continue usage when it should be revoked;
3) revoke usage when it should be revoked;
4) revoke usage when it should be continued.

Each result should be evaluated before a decision is made. The value of the decision is a combination of possible risks and benefits that are connected with a particular decision (see Table 3). The positive cost values in the table represent gains when negative values are losses connected with incorrect decision.

<table>
<thead>
<tr>
<th>Continue access</th>
<th>Satisfied policy $(1 - p_i(t'))$</th>
<th>Failed policy $p_i(t')$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revoke access</td>
<td>$-C_i^{RS}$</td>
<td>$C_i^{RF}$</td>
</tr>
</tbody>
</table>

Table 3 Decision matrix with costs

Decision theory provides a number of well developed methods to make a decision under risk and uncertainty. Decision making under risk means that we know exact probabilities of policy failure. We apply a simple probability-weighted utility theory for analysis of alternatives (using Equation 1). The idea behind the analysis is to compare the benefits of allowing access and revoking it (i.e., alternative decisions). If we know that the probability of failure of a policy $i$ at time $t_0$ is $p_i(t')$ then the access should be allowed if:

$$(1 - p_i(t')) * C_i^{CS} - p_i(t') * C_i^{CF} > p_i(t') * C_i^{RF} - (1 - p_i(t')) * C_i^{RS}$$

(1)

Frequently, a policy consists of a number of complex usage rules. A complex rule contains several atomic rules, which constrain different attributes of a subject, an object, and an environment. We consider only the following three operators for aggregation of rules: conjunction (AND), disjunction (OR), and negation (NOT). These are basic operators, though the approach can be extended for specific operators if it is needed. 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 which needs several attributes. The first possibility is when four costs for the decision matrix (see Table 3) 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 formulas from the probability theory.

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. $C^{CS}$ is assigned to the whole policy because we get this benefit only if all rules are satisfied. It also does not matter why access to an honest user has been revoked. Therefore, this loss ($C^{RF}$) is also rule-independent and should be assigned to the whole policy. We also assume that there is no difference why we revoked access of unauthorised person ($C^{RS}$). The rule-dependent cost is the cost of a violation when the access has not been prevented (but is had to be). If one rule is more important than another one, we have to consider different losses ($C^{CF}$) caused by violation of corresponding rules. Thus, we combine risks to tackle this issue.

Now we should discuss what to do when the condition of Equation 1 fails. Naturally, the simplest solution which comes to the mind is to revoke the further access. This simplest solution is not, however, is applicable in all situations and, often, by far not the best one. Note, that we make a decision based on probabilities and this means that we can be wrong.

Other mitigation strategies are possible. First of all, the current session can be simply suspended unless a fresh value is received and a solid decision can be made. Another possibility is simply to ask for a fresh value right in the moment when Equation 1 fails. When none of the proposed strategies are applicable an additional attribute may be requested, which somehow mitigates a possibility of granting the access to an unauthorised subject. One more strategy is to rise an alarm which notifies a responsible person that a suspicious operation has taken place. This could be a message to an administrator or a marked event in a log file.

3. An example

As a running example, we consider a research and development (R&D) department of a small corporation, which develops and produces prototypes of novel electronic devices. The department consists of a laboratory, an assembly shop, a library, a coffee bar, and the corridor, which connects all these rooms. There is an additional direct door from the laboratory to the shop. The structure of the company is presented in Figure 65. In the example, we consider an engineer who works on a new project and uses a tablet computer for this purpose.

![Figure 65. Structure of R&D department](image)

In order to protect secrets about new devices from a possible leakage of this information, the personal is not allowed to access and use sensitive information outside of the laboratory and the assembly shop. The engineer is allowed to use his device outside of the laboratory and the assembly shop for any other purposes but for working on the project. The position of any person which has access to trade secrets is controlled by a location tracking system. A special sensor is implanted into the tablet laptops which sends the information about the position of the device every 15 minutes. It has been found that often the engineers go to the coffee bar to take some coffee and do not close the usage session of projects. Thus, sometimes they have access to the secret information outside of the allowed rooms, while sessions are still active and the next position check will be only several minutes afterwards. Reducing the period between position checks results in more power and bandwidth consumptions. Thus, there is a need for rational determination of the time when the check has to be scheduled.

The location attribute in the R&D department is a Markov chain of five states (see Figure 66). However, the number of states can be modified, because the access to the database should be
forbidden if the researcher is in the library, the coffee bar, or the corridor. Thus, these states could be replaced with one absorbing state $a$. The modified Markov chain is presented in Figure 67.

![Markov chain for User Location](image1)

**Figure 66. Markov chain for User Location**

![Markov chain with absorbing states](image2)

**Figure 67. Markov chain with absorbing states**

The example of the decision matrix is presented in Table 4.

<table>
<thead>
<tr>
<th>Satisfied policy</th>
<th>Failed policy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Continue access</strong></td>
<td>20</td>
</tr>
<tr>
<td><strong>Revoke access</strong></td>
<td>-100</td>
</tr>
</tbody>
</table>

**Table 4 Example of the decision matrix**

The probability of policy violation after $t'_1 = 7$ minutes is $p_1 = 0.0330$. When we apply Formula 1 we see that $-46.6 > -96.7$, and the usage session can be continued. When we consider this inequality after $t'_2 = 14$ minutes ($p_2 = 0.0659$) we see the opposite situation: $-113 < -93.4$, and the usage session should be revoked or some mitigation strategy should be applied.

The start of a process from different states can lead to different decisions about the usage session. The probability of policy violation after $t'_3 = 10$ minutes is $p_3 = 0.0471$ if the process start from the state 1 (the engineer is in the laboratory), in this case $-75.1 > -95.3$ and the access can be continued. The probability of policy violation after $t'_3 = 10$ minutes is $p'_3 = 0.0658$ if the process starts from the state 2 (the engineer is in the assembly shop), in this case $-113 < -93.4$ and some mitigation strategies should be applied.
7.2 Model Driven Security/OpenPMF

1. Description

Model Driven Security (MDS) allows security policy to be defined at a high level of abstraction before being converted into lower level technology-specific policies. This is useful in distributed, agile environments such as service oriented architectures, since the high level policy can be enforced across the entire SOA environment.

OpenPMF™ product provides a way to configure business-driven security policies (esp. authorization, incident monitoring) for applications and processes. It also automatically enforces and monitors security, and it minimizes the security maintenance cost for agile applications and workflows. OpenPMF™ therefore also supports SOA governance. The enforcement is done through a pull-down menu within their development tools (e.g. Eclipse, Intalio BPMS).

In addition to allowing the high- to low-level policy translation, OpenPMF™ also provides a means to directly apply the policy by adding security components into an SOA deployment. The system makes use of the CORBA Component Model (CCM) to allow introspection and navigation of component features in order to achieve this.

2. Detailed description of basic operation

The aim of Model Driven Security is to allow security policies to be defined in terms of high-level organisation-centric, business-centric, information-centric terminology, but so that these can be translated into low-level IT-centric policies in an automated way. In addition, the high-level policies generally constitute broad policy requirements that hold across collections of systems. The specific IT- and technology-centric security policies to be enforced in practice tend to be applied at the level of individual services. The MDS approach described here also provides a means for achieving this.

The authors claim there to be a number of important benefits with the MDS approach to applying security in SOAs.

The approach is able to regulate information flows between services and users. It does this in a fine-grained policy-driven way that allows it to be applied across large heterogeneous environments.

Changes to the security policy can automatically filter down from the high level policy into the technology-centric policies, which are then applied to the system itself. This reduces the effort required to manage system security.

The system is agile, so that changes in the policy or environment can result in an adaptation of the security measures at run time.

MDS ensures that business security requirements are aligned with policy-driven technical and IT security enforcement.

The approach matches the security requirements with the overall business enterprise architecture.

The technology is built up from a number of different systems and components. At the highest, most abstract level, security policies are described using a domain specific Security Policy Language (SPL). This is based on the meta meta model used to describe the abstract definition of the system. For example it could be based on Unified Modelling Language (UML), Business Process Modelling Notation (BPMN) or some form of Domain Specific Language (DSL). The SPL policy is then translated into a lower-level technology-specific policy in Policy Description Language (PDL) format. The translation can be automated, by extracting details about the functional aspect of the system from the meta meta object model that describes it. This process is shown diagrammatically in Figure 68.
The policy in PDL format is similar to a policy written in the OASIS eXtensible Access Control Markup Language (XACML). However, PDL extends XACML and presents it in a human-readable form. Nonetheless, PDL is not really suitable for large-scale policy definition since PDL policies tend to be made up of large numbers of fine-grained technical enforcement rules at a low-level of abstraction.

The OpenPMF™ framework provides capabilities for these PDL policies to be distributed to centralised Policy Decision Points (PDPs), and then further to Policy Enforcement Points (PEPs). In this way a single consistent security policy is applied across the whole application with a single administrative console. The actual policy enforcement is achieved using the PEPs which can, for example, control the invocations or data flow between services. In the event that a policy violation occurs, a notification is sent back to the administrative console.

3. An example

As explained above, the MDS and OpenPMF™ approach starts with a high-level policy description that applies across a federated set of services. This description is created in a Domain Specific Language that relates to the language used to describe the meta model of the system (e.g. UML, BPMN, etc.).

---


For example, in a healthcare scenario, the security/privacy requirement applied at the high-level might be the following:

“Every doctor is only allowed to access the patient record of the patient they are currently treating (unless the patient is treated in a crisis context, or the patient consents).”

This requirement would be captured in a modelling tool in a customised, customer-specific way. This is then transformed into a potentially large number (possibly hundreds of thousands) of technology-specific protection and authorisation policy rules that are then applied at multiple layers within the system. Such generated rules might take the following form.

“allow information flow if ‘caller X.509 cert. id doctor1’ via ‘firewall IP...’ calling ‘file patient1’ on ‘database IP...’ and from ‘hospital IP...’ and ‘doctor1 is treating patient1’ and/or ‘patient1 crisis’”

These many technology-specific rules are generated automatically based on the higher-level requirements and the model of the system.

In the paper referenced the authors also explain how the tool was applied as part of the AD4 ATC simulation system\(^6\). In this example various roles are defined (such as controller, administrator, etc.), and security is applied to a system comprised of several components. This includes a radar station (sensor), which sends data to a ControlWorkingPosition component. The ControlWorkingPosition component collects data from the radar station and presents it to flight controllers. An AdminWorkingPosition component allows control of the radar station, and finally an actuator component is an example actuator that communicates with the ControlWorkingPosition component. This configuration is shown in Figure 70.

![Figure 70. Extract from the ATM system design\(^7\).](image)

Combined with a given security policy, the OpenPMF™ system is able to automatically attach roles to the system in order to define a configuration including security information such as that shown in Figure 71.

---


\(^7\) op cit.
Figure 71. Example ATC Assembly\textsuperscript{8}.

\textsuperscript{8} ibid.
8 Threat response recommendation module

8.1 Architecture for Self-Adaptive Socio-Technical Systems (UNITN)

1. Detailed description of basic operation

The basic operation of the architecture requires two major phases: (i) creating the architecture for an existing socio-technical system (STS); (ii) executing the architecture in the STS.

Phase 1: Creating the architecture

Such activity is described in Figure 72 via a SPEM 2.0 diagram. The main objective of such phase is to define the models that the architecture uses at runtime for diagnosis and variant generation. Also, during this phase traceability links should be established between models and the existing system, so that events are bound to specific states of requirements.

The first task is Application Domain Analysis, which corresponds to acquire knowledge about the STS in terms of autonomous human or software agents and non-autonomous entities such as sensors and actuators. The resulting artefact is the set of agents and entities in the application. Also, such activity provides a list of domain-specific messages that are exchanged between participants in the STS.

Three tasks should be concurrently performed after the domain analysis: Requirements Analysis, Context Analysis, and Interaction analysis. The output of these activities consists of three models: a context model; goal models and domain assumptions; and commitments models. Each output feeds a subsequent activity. Context and requirements analysis are typically performed iteratively: the corresponding models are not isolated, given that requirements models are specified on the basis of context. For example, domain assumptions are implication over context entities, whereas the status of goals (active, started, done, failed) is characterized as contextual events and states. Given the tight connection between these models, changes in the context (requirements) model often requires to modify to the requirements (context) model.
After requirements and context analysis are completed, *Traceability Establishment* is performed. This task defines what to monitor at runtime in order to determine requirements satisfaction and violation. The objective of this step is to ensure traceability links between implementation and requirements. We propose to carry out this step by associating a specification—in terms of monitorable events expressed over the context model—to goal model tasks. In such a way, the status of a task is computed by monitoring the status of its specification in terms of events, whereas the status of a goal is derived from the status of the tasks that operationalize it. For what concerns commitments, traceability establishment consists of the mapping between domain-specific messages and commitments. In particular, it might be useful to exploit a middleware that observes interaction and interprets the exchanged messages according to the mapping.

After establishing traceability, the next steps are to select tolerance policies and to define reconfiguration mechanisms. Task *Tolerance Policies Selection* specifies tolerance policies for failures and under-performance. Some failures have to be addressed through reconfiguration, some can always be tolerated, some can be tolerated under certain circumstances. Policy definition is first enacted at design-time, but in most cases policies need to be redefined or adjusted at runtime. For instance, system administrators might realize that too many failures are considered, therefore the system performance is affected by frequent reconfigurations.

Task *Reconfiguration Mechanisms Selection* defines how failures should be addressed by the architecture. The activity produces two output artefacts: (i) *Compensation Plan* are intended to revert the effects of the failed strategies, and (ii) *Reconfiguration Strategies* describe a possible alternative to achieve current goals. Both steps depend on the actuation capabilities of the existing application: possible reactions are scenario-dependent for the architecture needs to control actuators or communicate with agents.

**Phase 2: Executing the architecture**

After the set-up activities of phase 1 have been performed, the architecture is ready to execute and support subsystems in an STS. Figure 73 provides an overview on the way our architecture operates.
The architecture is founded on a Monitor-Diagnose-Reconcile-Compensate (MDRC) cycle. Monitoring collects data about the state of the environment and the agents participating in the system from a variety of sources; Diagnosis interprets these data with respect to requirements models to determine if all is well, if not, diagnose the problem-at-hand; Reconcile searches for a new variant that best deals with the problem-at-hand; Compensation takes necessary steps to ensure that the new plan can be executed. Our architecture is designed for socio-technical systems, which are inherently decentralized, distributed and heterogeneous systems. Therefore, our architecture has to take into account the interaction between its participating agents and functional components as well as the supervisory MDRC cycle.

2. An example

We show here the architecture in action on a firefighting scenario, where the mission of the architecture is to support a fire chief that should organize a rescue team to put off a fire. After creating the requirements models as described in the basic operation section, the architecture is deployed and almost ready to run. The last activity before running the architecture is to set the adaptation policy that the architecture should consider. The frame used to specify these options is shown in Figure 74. First, the selection criteria metric should be defined. Here, two options are viable: consider the overall cost for the variant, or consider how much the variant differs from the current configuration.
Then, the system administrator can specify whether the new solution should differ from the current one, if threaten and violated capabilities and commitments should be avoided. Capabilities are an abstraction to represent the existence of a plan to achieve a goal. The architecture supports also compensation cost, which refers to the operations that are needed to revert the effects of plans that are currently executing and will not be in the new variant.

Importantly, the adaptation triggers can be specified. Adaptation triggers say which are the reasons for adapting. In the current implementation, these triggers are specified in terms of event types. We distinguish between two types of triggers: risks and opportunities:

- **Risks** represent events that might result in system failure. Risks are always compensated when detected.
- **Opportunities** represent events that might allow for better performance. Opportunities lead to adaptation only if the cost of the new variant is lower than a certain threshold, that can be specified through the interface.

Figure 75 shows the architecture in action, where the fire chief jim is trying to extinguish a fire, and a commitment it is relying upon is threaten. This means that the service provider might have notified jim it won’t be able to bring about the promised service. The top-left part of the screenshot includes the buttons to create a new scenario, open an existing one, saving the current one, and opening the adaptation policy frame, respectively. The main part of the figure shows the goal model for the considered agent. Colours represent the state of goals (e.g. green = active, white = not active, red = failed, yellow = threaten). Below the goal model are the set of commitments from other agents (bottom-left) and the states of the world that currently hold. On the right part of the frame are the variants that the architecture generates. On bottom-right is the current query: the goals that the agent should achieve. Currently, jim should extinguish fire.
After detecting such threat, the architecture responds to it because such event type (commitment threaten) has been selecting as an adaptation trigger (see Figure 74). The architecture generates a set of variants and selects the one having the lowest cost. Then, such variant is enacted. Figure 76 shows how the architecture has reacted to the threat. On the right side of the screenshot, two variants are shown. The second one has been chosen (it is selected in the screenshot), for its cost is lower. Such variant corresponds to replacing `tanker1` with a substitute service provider `tanker2`. At the bottom-left of the screenshot commitment `c4` is coloured green, to represent that such commitment is active.
Figure 76. The architecture reacts to the threat by generating alternative strategies (right part of the screenshot) and adopting the less expensive one.
9 Service threat monitoring module

9.1 MASTER Design Toolkit (UNITN)

1. Description

MASTER methodology supports the realization of Information Governance-Risk-Compliance initiatives within an organization, which includes a set of stages that lead an organization through the planning of control objectives, the implementation of risk responses and the monitoring of these risk responses.

MASTER Design Toolkit is a set of tools that comprises three main parts: workbench, repository, and verification & validation (V&V) tool.

The (graphical) workbench facilitates the analysis and modelling process. It provides a means for the analysts to model, analyze, and design the MASTER system (i.e., control objectives, control processes, and indicators). Its graphical design environment allows users to build a design model for the business process that is being analyzed.

The repository provides support to the MASTER analysts to store, checkout, and manage versioning of the development artifacts (e.g., models, policies). Moreover, the repository is also useful for a knowledgebase system that helps the analysts in analyzing and designing the MASTER system.

Finally, the toolkit is also equipped with a set of V&V tools to verify and validate particular properties of a control process model before implementing the design in terms of executable control processes.

2. Detailed description of basic operation

The MASTER Design Toolkit supports the definition and refinement of control objectives according to risk assessments. Control objectives state requirements related to a to a certain business process. They are identified to mitigate the risk and provide assurance for the achievement of business objectives. Basically, high-level objectives are first defined and then they are further refined into specific/detailed control objectives according to risk assessments. For each detailed control objective, control activities are identified, describing the actions needed to be taken to manage risks. Finally, technical specifications for the implementation of control activities are given through control processes. Control processes are designed (using gap analysis), tested and implemented along with quantitative indicators. Indicators serve as a metric to measure the level of compliance and control of business processes.

Using existing business processes, controls and control activities, a Design Model is obtained. The Design Model describes the business process, control processes and the supporting IT infrastructure.

The model documents the interactions between these processes. After the appropriate Controls Objectives have been identified and a Design Model of the system has been constructed, a Verification Model is built. The Verification Model is a formal description of the Design Model with a clearly defined semantics. Each Control Objective is tested against the Verification Model to ensure that they are being fulfilled and the results are recorded.

Graphical Workbench provides a graphical design environment to build the Design Model (DM) for the business process being analysed.

The repository contains previous designs and best practices. The simulation tool allows users to explore the design space and investigate the appropriateness of control processes and design decisions. The verification and validation tools (theorem assistant, model checker and simulator) allow the user to assess a proposed design and test if each Control Objective is satisfied. The verification tools allow the Analyst to determine if the system properties, and hence the Control Objectives, are satisfied by the Verification Model. When the verification tools determine that all the system properties are
satisfied by the Verification Model, then the system properties can be used by the Graphical Workbench to generate the initial operational policy files.

3. An example
Suppose a credit insurance company has to generate a credit report as depicted in next figure. In such a situation, using MASTER methodology we can first identify regulatory requirements and then perform risk assessment. (See details of this example in MASTER Handbook [11]).

Business Objective:
- Financial losses are limited and cash-flow is optimized

Regulatory Requirement:
- All data must be obtained from a certified resource.

![Diagram of credit report process](image)

**Figure 77. Example Business process: Generate credit report**

Risk assessment:
- Unauthorized user access: impact on the availability, confidentiality and integrity of credit report information
- Unauthorized access to supporting Information Technology infrastructure: impact on the availability, confidentiality and integrity of credit report information

Afterwards, to mitigate the risk and provide assurance for the achievement of business objectives, control objectives are identified.

Control Objectives:
- System’s security is appropriately implemented, administered and logged to safeguard against unauthorized access to or modifications of programs and data, that result in incomplete, inaccurate, or invalid processing or recording of financial information.
- The configuration of programs and systems security during change management is appropriately managed to safeguard against unauthorized modifications to programs and data that result in incomplete, inaccurate, or invalid processing or recording of financial information.
Data is appropriately managed to provide reasonable assurance that credit reports are complete, accurate, and valid.

Control objectives are further refined in:
- CO3: Authenticate users for each request
- CO4: Authorize users for each request
- CO5: Check the quality of self requested data

Control objectives are formalized in a specification language to be verified.

To meet the control objectives authenticate users for each credit report request and authorize users for each credit report request, an access control mechanism is employed to authenticate and authorize users. The control activity resulted is as follows:

**Control Activity:**
Name: Authenticate users and verify their authorization
Description: Users are authenticated and their authorization is verified before they use services
Control objectives: CO1, CO3

To implement a control process, analysts need to design in detail how the relevant control activity will be performed or executed.

![Diagram](image.png)

**Figure 78. Description of Business processes and control activities**

In parallel with the identification, refinement and verification of control processes, the key indicators are calculated for each business process. They ensure that everything is under control (checking the level of compliance, correctness, and coverage of the business process w.r.t its specification) and it is not a case that things work properly.

To summarize, the Design Toolkit helps in building a design model for the business process that is being analyzed. This model is compared to previous designs and best practices offered by the repository and is completed with a sufficient level of detail. Once completed, it is translated into a verification model which checks if the control objectives are satisfied.
10 Community support module

10.1 OAuth

1. Detailed description of basic operation

The typical “3-legged” (authentication and authorization) OAuth workflow can be seen on the figure below.

![Figure 79. OAuth authentication workflow](image)

An example of a “3-legged OAuth” protocol run (entities are in bold, services are in italics):

1. **Consumer** and **Provider** negotiate a shared secret (or an RSA key pair) ahead of time; both of them also know each other’s unique identifier. **Provider** has specified the API calls used to **initiate** the protocol, **authorise** transactions, **request tokens**, and **access Resources**.

2. **User** wants **Consumer** to access a **Resource** accessible through **Provider** and invokes the appropriate API call on **Consumer**.

3. **Consumer** connects to **Provider**’s **initiate** service and sends its identifier, a timestamp, a nonce, a **callback** URL, and a signature.

4. The **Provider** responds with an authorisation token and a token-specific secret.

5. The **Consumer** redirects the **User** to **Provider**’s **authorise** service with the appropriate token.

6. **Provider** authenticates **User**.

7. **User** obtains a verification token and forwards it to **Consumer**’s previously-specified **callback** URL.

8. **Consumer** connects to **Provider**’s **request tokens** service and sends its identifier, a timestamp, the authorisation token, a nonce, and a signature.

9. **Provider** sends **Consumer** a new authorisation token.

10. **Consumer** connects to **Provider**’s **access** service and sends its identifier, a timestamp, a nonce, the new authorisation token, and a signature.
11. **Consumer** can access **Resource**.

The signature of a request is the signed hash of a string (HMAC-SHA1 or RSA-SHA1) that contains:

- The HTTP action (such as GET, PUT, POST, DELETE)
- The “Host” header
- The target URI
- All parameters – including the contents of the “Authorization” header and the message body, if applicable – in alphabetical order

Therefore – since it effectively signs the HTTP action, the target URI and the message body – OAuth can also be used to realise authentication between REST-based web services (“2-legged OAuth”). Its main issue is that it does not officially support body content outside application/x-www-form-urlencoded.


It is important to note that the confidentiality of requests is not protected by the protocol, therefore eavesdropping attacks are possible. In order to resolve this, transport layer security should be employed.

2. **An example**

During a recomposition event, sensitive user data may need to be handed off from one service to another. Users could confirm such hand-offs securely by employing OAuth; since the service identifiers and shared secrets (or public/private key pairs) were defined before the recomposition took place, the security of the hand-off would be ensured.
11 Threat repository module

11.1 Security Vulnerability Repository Service (SEARCH)

1. Description

The SVRS is a repository of security-related resources (i.e. vulnerabilities, threats, security activities, security models) and the links between them.

Content consumers (software developers) download resources from the SVRS either by special development or testing tools, or via a web interface where they are presented in a human-readable and intelligible format. To access the content stored within the SVRS, consumers can use multiple search interfaces (free text-based or ontology-based).

Besides the aforementioned repository functions, the SVRS also provides community services. Community services include communication functionality related to security in general (e.g., articles, comments, forum) and the SVRS (feature requests, bug reporting).

2. Detailed description of basic operation

The SVRS is built on a standard three-tier architecture consisting of the storage layer, application logic layer and presentation layer. Figure 80 shows the high-level SVRS architecture along with its interfaces and interactions with various kinds of users.

![Figure 80. SVRS high-level architecture]

Different actors can use the SVRS for different purposes and by using different interfaces; they can use various tools or the user interface to download content, submit content, submit usage statistics, and gather feedback.

The interfaces of the SVRS are based on open standards and are well-documented. The machine interface of the SVRS is a RESTful web service, minimizing the effort required on the parts of tools to interact with it. A special SVRS instance (the ‘sandbox’) allows tool developers to integrate their tools without affecting the operation of the SVRS. The user interface of the SVRS is a standard servlet- and JSP-based web application.

The SVRS stores a large variety of content:

Abstract security knowledge such as vulnerabilities, causes, threats, security activities and security goals. These are called core elements.
Categories to organise core elements into groups and create ontologies inside the SVRS. For example, it is possible to create categories that organise core elements according to their relevant aspects: programming language (“Java”), phase of the software development lifecycle (“Implementation phase”), type of affected software (“Media player”).

Artefacts for use by SHIELDS Compliant security tools, such as detection or testing rules. Each such rule deals with a specific core element (such as detecting a threat or eliminating a vulnerability).

Artefacts for human use, such as source code examples or human-readable documentation relevant to a specific core element.

Security models elaborated within the SHIELDS project, such as Security Goal Models, Security Goal Indicator Trees, Vulnerability Detection Conditions or Misuse Case Diagrams. Each of these models deals with a specific core element.

The individual components of security models– such as sub-trees, graph nodes, or individual logical statements – can also be stored within the SVRS; this allows contributors to reuse parts of existing models.

All content inside the SVRS is strongly interconnected, and it is possible to traverse the SVRS using these connections. The UI rendering of an example SVRS resource – and its connections – is shown in Figure 81.

The SVRS itself is responsible for proper management of resources, ensuring the integrity of each resource, and enforcing access control. Part of the content may only be available to paying customers, which is also enforced by the SVRS. Users may also want to subscribe to resources in the SVRS and be notified whenever they are modified. All resources are versioned, and it is possible to view the state of the SVRS (resources and their relations) as it was at any given time.

In order to ensure that content inside the SVRS is of a consistently high quality, the repository also supports a workflow for addition of new content. Content providers (i.e., security experts) upload new resources or propose modifications to existing ones. A special group of administrators verifies these submissions for semantic correctness and validity, and endorses them. Once a submission passes the endorsement process, it becomes available for download.

Figure 81. Resources inside the SVRS
Usage of SVRS content is logged. Usage data – combined with vulnerability occurrence information and security activity usage information collected by security tools – can be used to generate statistics, providing valuable input for security researchers. Once validated, all such data is anonymised to ensure privacy: the SVRS retains no information that could be used to identify the uploader or the specific software the tool was used on.

The SVRS can be deployed as the central element of a centralised architecture; it was used as such in the SHIELDS project. It is also possible to deploy SVRS instances locally; such SVRS instances operate independently from each other.

3. An example

A security tool that performs validation of secure service behaviour can use the SVRS to keep its local database – containing vulnerability definitions, for instance – up to date. They can also get new content applicable to the tool by performing search queries on the SVRS and traversing the connections between SVRS resources appropriately.

If the security tool deals with a vulnerability, it can submit appropriate statistical data back to the SVRS. Others can use this statistical data to find which vulnerabilities or threats are the most dangerous at the time and also find ways to deal with them by finding security models or artefacts connected to them.
12 Marketplace

12.1 Composition Repository (ATC)

1. Detailed description of basic operation

The Composition Repository enables users to store and manage the compositions that will be created by the use of the SODIUM composition suite, comprising of a storage area for compositions and a web enabled user interface for the management of this storage.

The Repository supports:

- Storage of Composition files
- Versioning of Compositions
- Easy location of Compositions through the use of metadata
- Composition Role Access control (RBAC)
- Remote Access (Web based Application)

The tool has been built with CodeCharge Java servlets, MySQL and is running on a Tomcat Server. It provides a storage area and a web interface is available as depicted on the following figure:

![Composition Repository by ATC](image)

**Figure 82. Composition Repository by ATC**

2. An example

This tool exhibits an interface through which a developer may provide the appropriate filters and identify the most appropriate database records that suit to his/her queries. The tool facilitates for answering simple MySQL calls and maintains on the background a database for storing the specifications of the composition of services.
13 Training material module

13.1 Automated Training Material Generator (SEARCH)

1. Description
The ATMG is a tool and set of techniques for creating customized training material subsets (PowerPoint slides, hand-outs, etc.) for various target audiences from a large training material components pool. It is able to automatically generate different “flavours” of the training slides from the same base components depending on the needs of specific audience groups.

The system consists of the ATMG software, the configuration descriptions for the individual trainings, and the base training material component pool. We also established a methodology about how the system should be best utilized in everyday practice.

The content creators are contributing to the component pool, while the training manager is in charge of defining the requirements for an individual training until the training material has been generated. Then the presenter receives a fully customized, presentation-ready material package for his particular training session.

2. Detailed description of basic operation
The ATMG is built on the concept of programmatically building and modifying the content of modern (OpenXML-based) Microsoft Office documents. The most important content differences between various training materials typically include:

- target audience specifics: which set of slides should be included or left out
- differences between individual training sessions: date/venue/presenter

These differences are described by the individual training configuration files – together with other customizable properties.

The training material components pool consists of documents provided by various content creators, with each section of these documents tagged according to its intended use. The proper tagging is a key concern from the perspective of an efficient training generation algorithm. The current methodology of tagging suits the business needs of SEARCH-LAB, but can be modified in order to better suit the needs of training concepts within Aniketos.

Based on the configuration files, the ATMG software creates the final training material from the tagged components, customized to fit the individual training’s purpose.

There are several advantages of this approach other than being an automated way of generating the final, customized training material:

- It can handle large and complex training situations, such as several different training content providers, different presenters and various target groups; the training material for each training session would need slightly different content but the topic of the training sessions would be similar.
- Both the source materials and the training configuration files used to generate the training material can be easily managed via a version control system.
- There will not be any wasted effort from manually compiling training material that is only slightly different from existing material.
- It is easy to identify (and give credit to) each of the original content providers in all generated training material.
3. An example
SEARCH-LAB had been conducting a security vulnerability training for developers for years when we got a request to conduct a similar training for testers as a different audience. After analysing the requirements, we concluded that most of the several hundred slides (the training is a five-day course) were also relevant for the new audience. However there were slides that we either wanted to leave out or change for this particular training; we also needed several entirely new slides.
Therefore we appropriately changed the tagging of the original component material based on the new training requirement, produced a few new slides, and left the rest of the work to the ATMG, which generated the new training material.
Since the material for both courses – developer, tester – is generated from a common source (the component materials), we are able to avoid having to manually update and sync the material of both training courses if anything needs to be changed in the component materials in the future.

14 Service composition Framework

14.1 Internet of Services: USDL/SDF

1. Description
The Unified Service Description Language (USDL) originated from the German lighthouse project Theseus/TEXO. USDL builds on models for describing business and technical services, and creates a unified description of related research efforts. It is not meant to replace other specifications in the technical service stacks, but aims to complement them by adding necessary business information. On the other side, it was not designed for targeting automated services only. USDL is generic enough to be used for the description of manual services that have no technical implementation.
The Service Delivery Framework (SDF) provides a dedicated way through which services can be provisioned and delivered beyond “firewalls”, out to global business networks. Beyond the classical access and orchestration considerations of SOA, the SDF will allow services to be discovered, repurposed and operationally optimized on a global scale, be it for: exposure beyond software registries to service marketplaces; new service innovations and channels through third-parties potentially unknown to original providers; re-hosting through low-cost cloud environments for SMEs and untapped consumer segments, outsourcing interoperability and other service delivery support functions through specialists like gateways and integrators – among various provisioning considerations.

2. Detailed description of basic operation
The Unified Service Description Language (USDL) The Unified Service Description Language (USDL), which produced two iterations of the language. Meanwhile other research projects in Europe and Australia have heavily contributed to what is now the 3rd iteration, called USDL v3.0.
The general design principle was to create a unifying entry point into the overall set of service metadata, which in the end, comprises several artefacts in different formats. The role of USDL is to enable a number of Internet of Services (IoS) use cases, among them discovery/matchmaking, aggregation and bundling. In other use cases, where it was clear that a more specialized format will be used by components of the SDF, USDL only provides an abstraction and a link to the specialized artifact (e.g. WSDL, BPEL or WS-Policy).
The USDL defines a model to describe services in the universe of discourse Internet of Services, and services computing in general. Capturing relevant concepts in a structured way is the main challenge of any model design. While working on USDL, three general groups of concepts were identified:
Foundational concepts: These concepts cannot be associated uniquely with a single aspect of service description. They capture common entities that occur in the context of different aspects. Examples include time and location concepts, as well as organizational concepts. The foundation also comprises of general characteristics common among entities, e.g. name, description or unique identifiers.
Generic service description (horizontal) concepts: These concepts apply to a broader range of services without being specific to an application or a domain that exists in the Internet of Services. Examples include concepts like pricing, service availability or legal parameters.

Domain-specific service description (vertical) concepts: These concepts capture description aspects that are rather special, i.e. are particular to a certain application or industry vertical.

USDL is conceived as a generic language and therefore covers concepts from the first two groups. Even though USDL is thereby limited to the generic part of service description, the set of concepts to model is still quite extensive. In order to ease readability and simplify maintenance of the model, it was decided to separate it into subsets of concepts that are logically related. Relatedness in this context means: being primarily concerned with the same aspect of service description. Thus, subsets, henceforth called modules, are aligned with the different aspects identified.

3. An example

15 Service Runtime Environment

15.1 MUSIC studio and middleware (SINTEF)

1. Description
MUSIC provides application developers with methodologies and an open platform for the development of self-adaptive context-aware mobile applications, which includes a runtime environment (MUSIC middleware), development tools (MUSIC studio) and pilots. MUSIC can be deployed on Windows, Linux, Windows Mobile and Android devices. A public demonstration held in the subway of Paris demonstrates the success of the project. The software, documentation, and tutorials are available.

MUSIC targets component-based applications that may be dynamically composed and reconfigured at runtime. MUSIC combines component-based software engineering with service-oriented architectures to allow applications to benefit from services in the service landscape by adapting to the service availability. It supports both component-oriented and service-oriented adaptation, and facilitates the building of “systems of systems” where self-adapting applications collaborate through dynamic QoS-aware service discovery and binding and adapt in a coordinated way driven by service level negotiations. The adaptation planning is based on the utilities of alternative configurations in response to context changes, and considers components installed on its computer to populate the roles of the composite system model, service providers located by dynamic service discovery to bind dependencies on external service providers, and takes into account service level agreements with consumers of provided services. The properties of components and services (e.g., service levels, non-functional properties including security properties) are described in property predictors, which are evaluated at runtime and used as input to calculate the utility of a composite configuration.

2. Detailed description of basic operation
The MUSIC architecture defines two main building blocks: the MUSIC Studio, providing a methodology, notation and tools supporting the modelling and design of context aware self-adaptive applications, and the MUSIC Middleware, providing a distributed platform supporting the efficient execution of these applications in ubiquitous environments.

The MUSIC studio (Figure 83) is a set of tools integrated as plug-ins on the Eclipse platform which support a methodology based on Model Driven Development. In particular, the MUSIC modelling tool is based on UML profile and implemented in Eclipse Papyrus. The modelling tool models the application variability model, context elements and data structures, as well as service functionalities.
and QoS-properties at an abstract and platform-independent level. MUSIC allows developers to specify application configurations (typically compositions based on components and services) as well as their properties from the adaptation point of view. Property predictors (also called property evaluators) and utilities are specified using the modelling tool. Property predictors are expressions over the context, the resources and the properties of collaborating components, and in the case of composite components, also the properties of the constituting components. They predict the values of the non-functional properties of a component or composition for a given execution context. The utility function is an expression over the predicted properties of the system configuration and the properties of the current context. Property predictors can model both functional and non-functional properties, including security properties. Figure 84 gives an example property predictor specified using MUSIC modelling tool. Property predictors and utilities are evaluated at runtime by the adaptation middleware aiming to adapt the running systems so as to maximize the overall utility. MUSIC studio provides also a visualisation tool for testing and simulating context changes and adaptations. MUSIC has also ontology support for services.

![MUSIC studio overview](image)

Figure 83. MUSIC studio overview

![Example property predictor](image)

Figure 84. Example property predictor (evaluator) specified using MUSIC modelling tool

MUSIC middleware (Figure 85) provides context monitoring and reasoning, adaptation reasoning and reconfiguration supporting dynamic self-adaptation (reconfiguration or (re)composition). In particular, MUSIC middleware provides discovery and remoting services for dynamic QoS-aware service advertisement, discovery and binding. MUSIC middleware supports also service level negotiation and monitoring at runtime. MUSIC is based on a flexible plug-in architecture where plug-ins can be implemented and integrated to the middleware, such as discovery and negotiation plug-ins based on different discovery and SLA protocols.
MUSIC has used security-oriented property to demonstrate self-adaptation by runtime aspect weaving. A short description of how services, service level negotiation, agreement, and monitoring have been integrated into the adaptation process in MUSIC is given below:

*Modelling service offers and dependencies:* MUSIC modelling tool supports the modelling of applications with regard to service offers and dependencies;

*QoS aware service publication and discovery:* When adapting an application the middleware will publish or withdraw service offers in accordance with own adaptation decisions, both regarding services offered and provided service levels. When a consumer discovers a service, the advertised service level is used to create a service plan\(^9\), which can later be evaluated by the middleware when computing the utility of the available compositions;

*Service selection and negotiation:* The middleware selects the most appropriate service offerings among the set of available service providers and different service levels available, each considered as a variant, by deciding if the variant with its service level can contribute to the composition configuration that gives the highest overall utility. If the reasoning process selects a particular service variant, a negotiation process is initiated with the provider. If the negotiation succeeds, an SLA will be created and the service will be provisioned with the corresponding service level. In case of negotiation failure, the middleware selects another variant and eventually re-negotiates;

*Service level monitoring:* the conformance of SLAs is checked according to the predicted property values defined in the property predictors leveraging the MUSIC planning mechanism;

*Service level violation and re-negotiation:* A service level violation discovered by service monitoring will trigger the adaptation process, which may select another service provider and re-negotiate.

\[\text{Figure 85. The reference implementation of the MUSIC middleware}\]

3. An example

We have used InstantSocial (IS) trial application [221] to demonstrate MUSIC concepts and platform for self-adaptation. IS is a media sharing platform for transient user groups that allows members to tag, to comment, and to search for text and images. The application is modelled using MUSIC modelling tool, automatically code generated using MUSIC transformation tool and deployed on mobile phones running MUSIC middleware.

\[\text{Figure 85. The reference implementation of the MUSIC middleware}\]

---

\(^9\) A service plan is a recipe to create the binding to a service provider, including details and QoS properties of the service.
D1.1 Consolidation of existing work 219

Figure 86. Design of the InstantSocial application

Figure 86 shows the design of the IS application 10: Variant configurations, components, their properties and associated property predictors. The CR (Content Repository) component type provides routing service (rs) and content access service (ca). A Full configuration of IS has both an UI and a CR, while a Mini configuration has only UI and uses the ca service provided by another node with Full configuration. The property types defined are avy (availability of the content), rut (resource utilisation of a node) and rsu (routing service utilisation), since these properties are considered during dynamic adaptation. Property predictors for the components and configurations are also shown. The utility function of a configuration is defined as:

\[ \text{utility} = c_1 \cdot \text{avy} + c_2 \cdot (1 - \text{rut}) + c_3 \cdot (1 - \text{rsu}) \]

where \( c_1, c_2, \) and \( c_3 \) are relative weights of the properties and they sum up to 1.

Figure 87. Network layout before (left) and after (right) node A joins the existing InstantSocial network

Consider two cases for runtime adaptation: 1) Dynamic composition of services. If a new node A wants to join an existing IS network where nodes B-E all run Full configuration (Figure 87), the MUSIC middleware on node A will discover the rs and ca services provided by other nodes and compute the utilities for different composition configurations based on the property predictors and runtime context. Suppose node A has enough resource to run Full configuration, then after the best combination of services is identified (e.g., provided by C and B), the nodes negotiate SLAs for the routing services. After successful negotiations, A connects to B and C, and the rs.conn property of A’s provided rs service is updated. 2) Dynamic reconfiguration and service recomposition. Assume that B is the first node noting this change by discovering the new rs service provided by A. Therefore node B re-adapts and its utility function indicates that it is better to disconnect from D in order to connect to A. After negotiating with and connecting to A, B also updates its rs.conn property. Similar adaptations will occur for other nodes by the update of the respective rs.conn property.

15.2 Dynamic and Static Aspect Weaving

1. Description

Object-oriented programming models emphasise the grouping of functionality into self-contained components. Although this often seems a natural way to separate out functionality, there are some cases where this isn’t suitable. In particular, some concerns are more appropriately spread across the various components of a system. Examples include logging and profiling capabilities. In these cases, a better approach is to add similar functionality throughout the programme. Security is another example of such a cross-cutting concern.

Aspect Oriented Programming (AOP) provides a way to allow these cross-cutting functionalities – referred to as aspects – to be added into a program in an effective manner. The DSAW platform

---

10 For detailed description of the design model and scenarios, please refer to paper [3].
provides a way to inject such aspects. However, unlike many other AOP platforms, DSAW utilises structural reflection to inject these aspects dynamically at run-time. This provides a potentially effective way to dynamically adapt a software service at run-time in order to adjust its security properties.

2. Detailed description of basic operation

As explained above, the DSAW platform provides AOP functionality in a dynamic way, allowing cross-cutting aspects to be injected into a program either at design- or run-time. In effect, this means that it provides an effective way to apply new security mechanisms to, or adjust the existing security mechanisms of, a software component that’s already running.

AOP works by providing a number of join points onto which additional functionality can be latched. Join points can cover a variety of structural elements of a program, such as entry or exit from a method, the reading of a field, triggering of an exception, and so on. Aspects (pieces of code/functionality) are attached to these join points and become part of the application. The result is that this injected code is executed whenever a join point is reached. Join points can be specified generically, for example so that a piece of code is injected across all method calls. This allows the effective introduction of cross-cutting functionality, since the code can be injected at multiple points simultaneously across an entire piece of software. However, join points can also be defined more specifically. For example, they may be chosen to apply just to a particular method call or field access with a given name or signature.

The result is that AOP provides a powerful way to utilise structural reflection to introduce new or altered functionality into a piece of code.

The DSAW platform applies to .NET code and provides a particularly powerful example of AOP for a number of reasons. Most AOP platforms are applied to the source code of an application so that the aspects are compiled into the application itself (a process referred to as weaving). However, DSAW works not at the source code level, but at the Intermediate Language (IL) level. IL is the .NET equivalent of Java bytecode: it is the platform neutral code that .NET programs are compiled into and which is then executed by the .NET runtime environment. In order to do this, the IL executable is first passed through a join point injector as shown in Figure 88. This provides the hooks that allow other pieces of code to be added to or removed from.

The DSAW platform applies to .NET code and provides a particularly powerful example of AOP for a number of reasons. Most AOP platforms are applied to the source code of an application so that the aspects are compiled into the application itself (a process referred to as weaving). However, DSAW works not at the source code level, but at the Intermediate Language (IL) level. IL is the .NET equivalent of Java bytecode: it is the platform neutral code that .NET programs are compiled into and which is then executed by the .NET runtime environment. In order to do this, the IL executable is first passed through a join point injector as shown in Figure 88. This provides the hooks that allow other pieces of code to be added to or removed from.

![Figure 88. DSAW pre-deployment architecture](image)

Working at the IL level provides a number of advantages. First, it means that DSAW aspects are language independent. They can be written in any .NET language, and woven into any other .NET program, independent of the language it was written in. Consequently the service and the aspect woven into it could be written in different languages. Second, the source code of the original software service isn’t required in order to inject the aspects into it. Aspects can therefore be added in to existing

---

pieces of software even if they’re in .NET ‘binary’ form. Finally, DSAW aspects can be injected or removed at run-time. The functionality of the software can therefore be changed even while it is currently being executed. The process for dynamic adaptation of code functionality is shown in Figure 89.

![Figure 89. Dynamic Application Adaptation](image)

It is also possible to inject multiple aspects on to layer top of one another. This is important from a security point of view, since different security requirements might require different pieces of functionality to be added in to a system, some of which may overlap.

3. An example
The platform could be used in order to adjust the security properties of a service during run-time.
Suppose a service provider wants to build a system from a number of existing off-the-shelf services, but requires that the data transmitted between components is sent in encrypted form.
Even if the components don’t themselves have the capability to transmit encrypted data, this functionality could be added using aspects. This could be achieved by injecting encryption code at all points in the service where network socket write calls are made, and decryption code at all points where network socket read calls are made.
DSAW would allow this functionality to be added or removed even while the services are running, thereby allowing the security properties to be adapted at run-time (e.g. as a result of updated policies or changing environmental threats).

16 Identity management service

16.1 ED DIAMETER AAA (ELSAG)

1. Description
ED Diameter AAA is a Java-based implementation of the DIAMETER protocol in which the AAA rules are defined by a rule language and evaluated by a rule engine (JBoss Rules). The software framework is mainly composed by a AAA Client sw module and a AAA Server sw module, acting respectively as the authenticator and the Authentication Server. These two peers accomplish their communication through the DIAMETER protocol, which is a recent protocol used for AAA purposes which overcomes RADIUS defects.

2. Detailed description of basic operation
The AAA is a software system for the conditional access: the task of AAA is to check the credentials of the user or of the user terminal (authentication), to verify what services and applications the UT (User Terminal) is allowed to use (authorization) and to keep track of the used resources and services

12 ibid.
The authorization process can have also a more general meaning, because it can also have the aim of checking the client permissions for accessing physical and transport resources and applications. Authorization process can also be applied to the case of a network signaling layer element that attempts to access physical resources while establishing session between two UTs.

The AAA framework is composed by merging two basic elements:

- **The AAA Client**: this element acts as interfacing element between UTs and the Authentication Server in the scenario and it is called Authenticator. It receives access requests by the connected user terminals and forwards them to central server the request for AAA tasks completion.

- **The AAA Server**: it acts as Authentication Server and its aim is to verify credential of the received access requests and in some scenarios verifies the chance of accessing physical resources.

These two elements can be organized in a peer to peer composition so that they can communicate to provide the access network they belong to with AAA functionalities.

Below the picture shows a typical AAA scenario:

![Diagram of AAA framework](image)

**Figure 90. AAA framework overview**

In the picture it is possible to identify the main elements which are: the clients connected to Authenticator (also called supplicants), sending authentication and authorization requests; the Authenticator itself, providing clients on one side and DIAMETER protocol on the other side with an AAA interface; the Authentication Server which is the element that accomplishes AAA tasks.

### 3. An example

In the following figure we show as example the functional architecture of our AAA Authenticator which we defined for M3-CAST national project, coordinated by Elsag Datamat.
For the sake of precision, in the specific implementation for that project, the AAA Client was actually composed by a couple of distinct modules, one towards the supplicant user devices and the Service Provider (named as Service Provided Signalling Module) and one towards the remote AAA Server (named as AAA Proxy Authenticator because it operates as a proxy towards the remote AAA Server).

From the figure it is not difficult to recognize the main functional elements of the AAA Client and how it may be easily interfaced with supplicant user handsets.

The AAA Proxy Authenticator is divided into two main elements, logically divided according to their function. Authentication and Authorization are managed by the left part of the module, while Accounting is dealt with by the right part of the module.

The Service Provided Signalling Module (the acronym SP is used here for the sake of conciseness) is the architectural element in charge to manage, on the Service Provider’s side, the AAA procedures towards the users. In order to give this support it communicates with AAA Proxy Authenticator.

On the left side of the AAA Proxy Authenticator module there are the elements which co-operate in order to provide the system with functionality needed to process incoming requests for authentication and authorization. Every single element implements the functionalities which are relevant for specific authentication protocols. It is a group of elements, not a single element, because the AAA subsystem in M3-CAST must provide the SP with necessary support to authenticate the user in many different ways, according to the access technology used by the client. In fact, M3-CAST gives users convergence on different communication protocols in order to access the provided services. It is up to the AAA Proxy Authenticator, and in particular to the element which is named as AAA request and Auth method decoding in the figure, the tasks of identifying the user’s access method and of routing the single user request to the relevant element which will be in charge of processing it.

After the specific processing of single user requests, the next step is done by the element named as Authentication and Authorization Diameter to App interface. This element interfaces the group of specific elements which are related to the different auth methodologies to the AAA Server according
to the Diameter protocol. The operations of this element consist on adapting information received by application modules according to Diameter needs. This element can receive by the application modules a wide variety of information, as user credentials, username and/or password, or specific identifiers of interest of the single auth protocol. All the information is converted and adapted in order to be managed by Diameter protocol, which is in charge of the physical transportation to the AAA Server. Whenever the AAA Server gives an answer to a user request, this module is in charge also to reformat it according to the need of the specific auth method, on the basis of the initial user request.

On the right side of the AAA Proxy Authenticator module there are the elements which co-operate in order to provide the system with functionality needed to process incoming requests for accounting.

The accounting politics adopted in M3-CAST are simple, and are in charge of the Accounting Sessions Manager. The only interesting actions are starting and stopping the accounting of a service provision after a user request.

Again is the SP Signalling Module which communicates to the AAA Proxy Authenticator the need to start counting the amount of used resources. This preliminary start request is associated to a timestamp, fixing the moment when accounting starts. The stopping request is analogous, but it stops the accounting of the amount of used resources. In M3-CAST implementation of accounting, it was sufficient to count simply the elapsed time among starting and ending the use of a video service, but in general more information could be considered as interesting for accounting and possibly for future billing (e.g. the resolution of download videos).

17 Usability and acceptance of the platform

17.1 User-centred Design and Interaction Design (PLUS, DBL)

1. Description

From a Human-Computer Interaction (HCI) and Usability viewpoint, an understanding of users and their interaction with the interface of a system is central for the creation of usable systems. In contrast to other design approaches, User-centered design (UCD) starts with investigating the needs, wants, and limitations of users, on which all further steps of the design and development process are based. Thus, when deploying a UCD approach in system development, user acceptance and actual usage of the system will be at a higher level. At large, UCD defines a general process for including human-centered activities throughout a development life-cycle by focusing on users through the planning, design and development of a product.

The UCD approach is a complement to other software development approaches rather than a replacement for them. System development following such a design approach is characterized by an investigation of user and task requirements and an active involvement of end users in the system development lifecycle. In this lifecycle, user requirements elicitation, design activities and user requirements evaluation tasks take place in iterations.

Related to the UCD approach are e.g. the Usability Engineering Lifecycle (Nielsen 1993, Mayhew 1999), Contextual Design techniques (Beyer and Holtzblatt 1998) and Participatory Design methods (Schuler and Namioka 1993). Additionally, Flechais, Mascolo and Sasse (2007) modelled the secure software development process “Appropriate and Effective Guidance for Information Security” (AEGIS), which provides guidance in the development of secure and usable systems. This is seen as part of the field of HCI in Security (HCISec).

2. Detailed description of basic operation

A central goal is the integration of HCI and Usability engineering techniques into the Aniketos platform and thus in the service engineering lifecycle. In the user/human-centred design approach, several activities (as defined by the ISO 9241-210:2010 standard for “Human-centred Design”, see Fig.1) are provided:
In this model, once the need to use a human centered design process has been identified, four activities form the main cycle of work:

- **Understand and specify the context of use**: Identify the people who will use the product, what they will use it for, and under what conditions they will use it.
- **Specify the user and organizational requirements**: Identify any business requirements or user goals that must be met for the product to be successful.
- **Produce design solutions**: This part of the process may be done in stages, building from a rough concept to a complete design.
- **Evaluate design against requirements**: The most important part of this process is that evaluation - ideally through usability testing with actual users - is as integral as quality testing is to good software development.

Deep Blue mainly uses the following techniques for the iterative requirements collection and evaluation phases:

**REQUIREMENTS METHODOLOGIES**
- Scenarios (use of) - for requirements gathering and evaluation
- Scenarios (use of) - for system testing and evaluation
- Questionnaires and Interviews
- Activity and Task Analysis

**ANALYSIS METHODOLOGIES**
- Qualitative and Quantitative Analysis of System Performances (or Key Performance Indicators)
- Ethnographic Observations
- Semi-structured Interviews
- Expert judgment
- Questionnaires (use and development of tailored questionnaires)

**CRIA – Critical Interaction Analysis Methodology**

PLUS has applied and adapted a range of UCD methods such as contextual inquiry, cultural probing, ethnographic methods, experience sampling methods, field observations, focus groups, personas and scenarios, physiological measurements, user and task analysis, user interviews and surveys, usability evaluations (expert- as well as user-involved methods), and usage logging and analysis.
3. An example

Many examples and best practices can be found in literature:


DBL and PLUS papers on the subject:


18 Case Studies

18.1 Italtel VoIP Suite (ITALTEL)

1. Description
The Italtel’s VoIP Suite is a software framework which enables the delivery of VoIP services and communication on top of SIP/IP core networks.

It is basically composed of a SIP Server, which supports IETF Session Initiation Protocol (SIP) for session management, and a Presence Server for storage, management and distribution of presence information, according to the Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) specifications by OMA (http://www.openmobilealliance.org).

2. Detailed description of basic operation
The SIP Server performs end-to-end message exchange between User Agents (UA) and other SIP proxies. The engine uses SIP/SDP signaling protocols for supporting session establishment and negotiation of media attributes. During registration, the location service creates bindings for a particular domain that associates an address-of-record URI with one or more contact addresses into a database, either internal or external (e.g. MySQL). The authentication is based on HTTP Digest mechanism, which can be locally managed; otherwise, user credentials can also be stored in an external database and retrieved by the authenticator using ordinary authentication protocols (e.g. Diameter).

Presence is an attribute related to user information (e.g. user status and availability, communication preferences, devices, applications supported, localization, ...). Acting as an enabler that allows a set of users to be informed about the availability and means of communication of the other users, the presence service can be exploited to enhance existing services as well as to create new added-value communication services.

The Presence Server supports presence subscriptions, notifications and publications of presence information. Presence authorisation is based on rules stored in the Presence XML Document Management (XDM) Server. The Resource List Server (RLS) is the functional entity that accepts and manages subscriptions to presence lists, which enables a watcher to subscribe to the presence information of multiple presentities using a single subscription transaction. Presence lists are stored in the RLS XDM Server.

A client could address both servers by using the SIP protocol and related extensions. Furthermore, the XML Configuration Access Protocol (XCAP) could be used by clients in order to manipulate the content of the presence lists and the presence authorization policies.
The XML document formats are compliant to IETF PIDF and RPID specifications. The presence data model includes the concepts of: person (e.g. willingness, activity, mood, …); service (e.g. communication models like PoC, IM, SMS, …); and device (terminals adopted) thus supporting a multi-device, multi-service environment.

Session initiation and teardown are obtained by using INVITE and BYE messages, respectively. Before initiating a session, the clients should register at the application level by issuing a REGISTER method to the SIP server. In addition, an user (presentity) may decide to PUBLISH his/her information status thus storing it into the Presence server.

Another user (watcher), that may be interested about this kind of information, could issue a SUBSCRIBE message in order to be notified of the status of presence of its peer(s): in this way, notifications messages (NOTIFY) will be received whenever a change in the presence status occur.

3. An example

This section provides a brief overview of the deployment of the Italtel VoIP Suite in a Eureka/ITEA research project, named €-Confidential. It was focused on the study, development and demonstration of a trustworthy execution environment for the delivery of Telecommunications and IT services. Actually, a key issue to face in the delivery of “mashed-up” telecommunication services is consisting in multiple and secure credentials management within different platforms by avoiding multiple annoying authentications. With the aim to overcome these limitations, Italtel’s suite demonstrated the benefits of identity federation between web services and telecommunication services for Next Generation Networks (NGN).

Three Service Providers (SP) have been developed for this scenario.

The VoIP SP service was included in order to demonstrate end-user’s seamless access between VoIP service and other regular service without re-authentication. The VoIP Suite has been used for this kind of service.

The Contact Book SP is basically a phone number repository, but can be extended in order to include some useful information about the subscriber, like profile, preferences, devices and location. It is a unique common database for the storage and handling of all subscriber information. The Contact Book server is accessible via HTTP protocol in order to search for phone numbers.

The Identity Provider (IdP) has been implemented according to Liberty Alliance Framework based on Sun Java Access Manager 7. The chosen standards are Liberty ID-FF 2.0 (SAML 2.0) for the browser based identity federation and Single Sign-On and Liberty ID-WSF 2.0 for the web services based authentication.
The user has capability of both SIP and HTTP clients.
The VoIP SP, the Contact Book SP and the clients have been customized in order to be compatible with the Liberty Alliance standards.

In the following a brief description of the experiment. The pre-requisite is an user that has already executed identity federation of both services, i.e. the VoIP SP and the Contact Book SP, with the Identity Provider in the same Circle-of-Trust (CoT). The user logs to the Contact Book SP by using the federated login, authenticates to the IdP and retrieves the information desired (phone number). Then, the VoIP service is used to make a call, but no further authentication will be required since the user has executed the federation between the VoIP SP and the IdP at a previous time.

18.2 WindChester (WIND)

1. Description

WindChester is a security service product developed and managed by Wind.

WindChester is an option offered as part of the Wind connection services NetRide or WindConnect. This is part of the offer which comprises supply/management of network security HW equipment and dedicated SW.

Wind by including it in its business service portfolio for managed security becomes the ideal partner for connectivity and security at the same time.

WindChester is delivered through a security management/control center called Security Operation Center (SOC). The SOC is connected over the Internet to the various security devices and apparatus located at the customer premises.

The Security Operation Center (SOC) is the collection of technical resources, standards, procedures and technological platforms (HW and SW) dedicated to the management and monitoring of the security aspects which are offered as services by Wind.

The SOC manages and controls the whole security flow of the customers by means of the management, control and analysis of the various customer apparatus in order to detect areas or systems that are more vulnerable to security threats.
2. Detailed description of basic operation

The platforms used by Wind to compose the WindChester service have been selected based on extensive market and product research and have been thoroughly tested at Wind laboratories. The elements that form the service are briefly described below.

**Firewalling**
- **CheckPoint**: software for firewalling and cryptography
- **Cisco ASA (Adaptive Security Appliance)**: integrated HW and SW security
- **Fortinet**: is an innovative technology in the security field. It is of the type Unified Threat Management (UTM). It is an appliance which can offer multiple security functions on a single platform. Thanks to an innovative signature recognition engine directly implemented in ASIC it is possible to manage different functionalities like Intrusion Detection System (IDS), antivirus, antispamming and web filtering employing the same HW and thus obtaining high performances.
- **StoneSoft**: with StoneGate Firewall it is possible to protect the network by creating a protective perimeter and avoiding attacks thanks to the use of Virtual Private Networks (VPN)

**Antivirus and Antispam**
- **TrendMicro**: SW especially oriented towards solutions implemented at the Internet gateway level (HTTP, SMTP and FTP traffic control)
- **Fortinet – Fortimail (Antivirus-Antispam)**: mail management that can be implemented at the Internet Gateway level (SMTP traffic control)

**Content Filtering**
- **WebSense**: SW for the control of the Internet “browsing content”. It allows the management of the user Internet browsing following specific policies defined by the customer enterprise
- **Fortinet (UTM)**: can be implemented at the Internet gateway level (HTTP traffic control)

**Intrusion Detection System (IDS)**
- **HP TippingPoint**: all the NIPS (Network-based Intrusion Prevention System) apparatus allow the configuration in high reliability mode by activating a configuration called Transparent High Availability (THA)
- **Fortinet (UTM)**: UTM that can be implemented at the Internet gateway level (HTTP, SMTP and FTP traffic control)
D1.1 Consolidation of existing work

Event Log Management (ELM)

- **RSA enVision**: events and log collection, log repository, management of data inventory and history, display in various forms of the collected data, correlation and alerting.
- **Fortinet FortiAnalyzer**: it is used for the centralized log collection, analysis tool and data storage. FortiAnalyzer receives log data from Fortigate devices or from the syslog of external servers.
- **EQNetwork**: this platform supports the real-time event gathering thus allowing the immediate vulnerability detection and the management of informatics disaster.
- **Insightix**: using a single instrument it is possible to execute a complete detection of all devices connected to the network and if required manage and control the access to the network.

3. An example

The diagram below illustrates the interconnections for the Management Services Personal Firewall Architecture (example).

![Diagram](image)

**Figure 96. Interconnections for the Management service**

With this architecture every connected computer in the customer premises LAN is configured with a personal Firewall which is a SW that allows to the manager of the security aspects of the customer enterprise or to the Wind SOC to configure the policies needed for the protected data interchange with the external LAN.

The diagram below illustrates this configuration:
For the personal Firewall, in case the systems are not Windows-based, different modules are required:
Linux: IPtables
UNIX(HP UX, SUN, IBM AIX): NetFilter

**Firewalling Service**
The firewalling system on which the service itself is based support all the protocols specified in the TCP/IP standard and are implemented using one or more heterogeneous hardware/software devices (a single router, a combination of routers, a single host system or more hosts that run a specific software, hardware equipment specifically designed for firewalling or a combination of all of them). The selection and implementation of the type of devices and the type of firewall depends on the system administrator)

*WindChester* can guarantee the following basic characteristics of the firewalling service:

**IP traffic filtering**: can protect an IP network or single workstations against unauthorized access by means of address, port and protocol blocking.

**Auditing and logging**: allows the analysis of the traffic passing through the firewall.

**Inspection Module**: executes the IP datagram inspection and filtering based on the implemented rules. The stateful inspection methodology is implemented thus avoiding the use of firewalling devices of the type Packet Filtering Stateless.

**Management module**: it is the functional component that allows the configuration and the monitoring of the firewall system.

On customer request *WindChester* can assure the following additional characteristics:

**Authentication management and access control**: it enables the regulation of the use of some services (FTP, Telnet, HTTP, HTTPs) that are allowed through the Firewall only after appropriate authentication.

**Port Address Translation (PAT) Management**: in order to increase the protection level, it enables the hiding of the effective receiving ports of a server system that is protected by a firewall by using fictitious ports.
URL Filtering Management: it enables the Web browsing through the firewall only to some specific workstations, it controls the browsing statistics and can block the access to some specific Internet/Intranet sites.

The Services of Network Firewall and Personal Firewall are provided by the Wind SOC by means of the Firewall Management platform that is connected through a dedicated link to the various security devices located at the customer premises.

Firewall Management (FWM) implementation process
The Services offered by Wind go through the following phases:

- Analysis
- Implementation
- Management

Analysis
- link verification and bandwidth specification between the equipment dedicated to the security services at the customer premises and the FWM platform that is located at Wind SOC
- identification of events to monitor based on the requirements expressed by the manager of the security aspects of the customer enterprise
- definition of the warning threshold levels based on the importance of the events generated by the security equipment installed at the customer premises
- verification of the events, of the correlation rules and of the threshold levels based on the policy established by the manager for the security aspects of the customer enterprise

Implementation
- definition and agreement on the security policies with the manager for the security aspects of the customer enterprise
- installation and configuration of the Firewall or Personal Firewall apparatus at the customer premises
- connection of the apparatus, to the FWM platform running at the SOC in Wind, through a dedicated link
- verification of the operating functionalities by means of a series of tests carried out by the SOC in Wind in order to check the correct operation of the security system and the agreement to the security policies as defined by the system administrator

Management
Monitoring of the apparatus dedicated to the security enforcement following criteria as:
- continuous verification of the functionalities of the apparatus
- proactive service: the FWM platform while it is continuously monitoring the Firewall or the Personal Firewall apparatus that are installed at the customer premises transmits to the SOC personnel also an alarm which triggers a successive intervention
- management of the alarms generated by the devices installed at the customer premises in order to trigger appropriate counter actions in case of attacks
- management of the security enforcement rules which are continuously updated following the evolution of the customer network

In addition a history log of the infrastructure variation is generated in order to keep trace of the evolution.
18.3 Poleod (DAEM)

1. Description

Poleod was developed at first to replace the need of a manually maintained register book for the Department of Urban Planning of the Municipality of Athens. Since then it has evolved into an application capable of tracking the progress of the citizens' requests towards the department as well as providing various information concerning Urban planning such as building permits.

2. Detailed description of basic operation

Poleod was developed as a two-tier application, with an Oracle Database used as data storage and a Windows application combining business and presentation layers.

The Windows application was build using the Borland Delphi 7 (now owned by Embarcadero Technologies). Its connection to the server is facilitated by the use of Oracle's client software. In its current instance it operates only within the organization’s intranet. The Database is used only for storage, with no business logic at all.

One of the two main usages of this application is the tracking of citizens' requests (see Figure 98 where the application is in Greek and the real personal data has been intentionally blurred). In order to reach their resolution they travel through many of the sub-departments of Urban Planning and every step is recorded in the database. Also, all information regarding building permits is recorded for future reference.

![Figure 98. Request Tracking Screen](image)

Finally, the application has many reports fine tuned using a variety of filters (see Figure 99, also in Greek and with intentionally blurred real personal data in the report).
3. An example
When a citizen needs to interact with the Department of Urban Planning he/she submits a request. This request is recorded in Poleod and this information can be used for future reference.

18.4 OPS (Integrated Information System) of Municipality of Athens (DAEM)

1. Description
OPS was created in order to integrate various back-office applications used by the administration of the Municipality of Athens and to create new services not only for the citizens of this municipality but also for business that operate within its limits. These services are provided through a web interface and include an information portal as well as the issuance of various certificates and the on-line payment of debts.

The portal provides the end user with information about both online and offline services provided by the Municipality as well as real-time maps with places and events taking place. The rest of the services are accessible through the same portal but require the use of both server and client side certificates in order to ensure encryption and authenticity.

2. Detailed description of basic operation
Ops is deployed in a heterogeneous environment with a variety of servers to host its components. However there is a common storage environment for all these components, namely an Oracle Cluster Database. A large part of every component business logic is implemented in that database.

There are others servers of the Oracle family of products such as:
- Oracle Portal, which hosts static content and simple applications (portlets).
- Oracle Application Server, which hosts more advanced applications and web services.
- Oracle Internet Directory, Certificate Authority and Single Sign On, which store user information and provide authentication and other security services.
- ArcGIS with Microsoft IIS, which provides dynamic maps.

These servers are part of various zones within the network of the municipality and each zone is separated from the others with firewalls so that even if one zone is compromised the rest will remain intact.

OPS consists of many parts the more important of which are presented below (Figure 100):
- Content Management, updateable through the Oracle Portal Interface, as well as special in-house applications.
- Register, an integral tool required by law for the operation of the Municipality. Every incoming and outgoing document passes through that system.
- Document Management, in which are stored the documents of the Register as well as other ones.

![Figure 99. A report of building permits](image)
• Web Portal, from where every service is served to the public.
• OPS Management, in which every aspect of the OPS configuration is managed.

OPS is an active platform which supports the administration of the Municipality of Athens. Many of its parts existed and operated as stand alone applications for many years. As such the “multi-application” OPS is gathering a great quantity and variety of data available in its database.

Figure 100. Major Parts of OPS

Figure 101. The main page of the Web Portal

3. An example
The services provided from OPS to the public fall into many categories. One of these is Government to Citizen Services. Some of these services require the establishment of a trust relationship between the OPS servers and the end-users browser. Once this relationship has been established, the end-user can request for a variety of services such as the issuance of certificates from the organization. The issuance can be either completed entirely through the internet or partly, according to the corresponding laws.

Another option available to the authenticated end-user is the on-line payment of bills.