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.

This deliverable presents the first Aniketos architecture, including the collectively created scenarios describing our visions of the research and technology development, along with the derived project requirements that guide the project. The deliverable is used as a common project reference across all the technical work packages, and contains a glossary that can be used as a reading companion to the other deliverables.
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 (SINTEF)**
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: Dhouha Ayed
dhouha.ayed@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

**Selex Elsag S.P.A. (ELSAG)**
16154 Genova, Italy
www.selexelsag.com

Contact: Paolo Pucci
Paolo.Pucci@selexelsag.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
Telecommunication Software and Systems Group (TSSG)
Waterford, Ireland
www.tssg.org
Contact: Miguel Ponce de Leon
miguelpdl@tssg.org

Universita Degli Studi di Trento (UNITN)
38100 Trento, Italy
www.unitn.it
Contact: Paolo Giorgini
paolo.giorgini@unitn.it

Athens Technology Center SA (ATC)
15233 Athens, Greece
www.atc.gr
Contact: Vasilis Tountopoulos
v.tountopoulos@atc.gr

SAP AG (SAP)
69190 Walldorf, Germany
www.sap.com/research
Contact: Achim Brucker
achim.brucker@sap.com

ITALTEL S.P.A. (ITALTEL)
20019 Settimo Milanese, Italy
www.italtel.it
Contact: Maurizio Pignolo
maurizio.pignolo@italtel.it

Paris Lodron Universität Salzburg (PLUS)
5020 Salzburg, Austria
www.uni-salzburg.at
Contact: Manfred Tscheligi
manfred.tscheligi@sbg.ac.at

Deep Blue SRL (DBL)
00193 Roma, Italy
www.dbblue.it
Contact: Valentino Meduri
valentino.meduri@dbblue.it

Wind Telecomunicazioni S.P.A. (WIND)
00148 Roma, Italy
www.wind.it
Contact: Rita Spada
MariaRita.Spada@mail.wind.it

Dimos Athinaion Epicheirisi Michanografisis (DAEM)
10438 Athens, Greece
www.daem.gr
Contact: Ira Giannakoudaki
i.giannakoudaki@daem.gr
# Table of contents

Aniketos consortium............................................................................................................................. iii
Table of contents ........................................................................................................................................ v
List of figures ........................................................................................................................................ vii
List of tables ........................................................................................................................................ viii
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 ....................................................................................... 5
### 1.5 Contributors ............................................................................................................................. 5

## 2 Context viewpoint ........................................................................................................................... 7
### 2.1 The domain of Aniketos .......................................................................................................... 7
### 2.2 Aniketos scenarios ................................................................................................................... 13
### 2.3 The environment systems ......................................................................................................... 17
#### 2.3.1 Service composition framework ...................................................................................... 17
#### 2.3.2 Service runtime environment ........................................................................................... 18
#### 2.3.3 Service registry ................................................................................................................ 18
#### 2.3.4 Integrated environments .................................................................................................. 18
### 2.4 Business to system mapping .................................................................................................. 19
#### 2.4.1 Aniketos stakeholders ...................................................................................................... 19
#### 2.4.2 Work process overview: Design-time ............................................................................. 26
#### 2.4.3 Work process overview: Runtime ................................................................................... 31

## 3 Aniketos requirements viewpoint .................................................................................................. 40

## 4 Component viewpoint ................................................................................................................... 50
### 4.1 System information model ....................................................................................................... 50
### 4.2 System decomposition model .................................................................................................. 55
### 4.3 Interface specification model ................................................................................................... 61
### 4.4 System collaboration model .................................................................................................... 67

## 5 Conclusion and further work .......................................................................................................... 75

References ............................................................................................................................................... 77

### Appendix A. Aniketos glossary ........................................................................................................ 79
### Appendix B. Aniketos scenarios ...................................................................................................... 93
List of figures

Figure 1: Goal: establish and maintain security and trustworthiness in composite services .................. 3
Figure 2: A domain model showing central concepts and their relationships ...................................... 8
Figure 3: Overview of various stakeholders for Aniketos ................................................................. 20
Figure 4: Typical processes related to design-time service composition ......................................... 26
Figure 5: Aniketos related to the specify service process ................................................................. 27
Figure 6: Aniketos related to the discovery and selection of service components ............................ 28
Figure 7: Aniketos related to service validation and contract establishment ....................................... 29
Figure 8: Aniketos related to service assembly and deployment ...................................................... 30
Figure 9: General processes related to runtime reaction to changes and monitoring ....................... 32
Figure 10: Aniketos related to runtime service provision ............................................................ 33
Figure 11: Aniketos related to runtime service validation ............................................................ 34
Figure 12: Aniketos related to runtime recomposition .................................................................... 35
Figure 13: Aniketos related to runtime reconfiguration .................................................................... 36
Figure 14: Aniketos related to runtime monitoring ........................................................................ 37
Figure 15: Aniketos related to service end user processes ............................................................. 38
Figure 16: Enriching the Aniketos platform at runtime ................................................................. 39
Figure 17: Except from the requirements database .......................................................................... 41
Figure 18: Information model for Aniketos and its environment .................................................... 51
Figure 19: Messages related to the Aniketos platform .................................................................... 52
Figure 20: Involved entities in a contract ....................................................................................... 53
Figure 21: Components in the Aniketos platform and in the environment ....................................... 56
Figure 22: Overview of Aniketos platform components with interfaces between platform components and interfaces towards the environment ................................................................. 61
Figure 23: Interfaces for Socio-technical security modelling tool and model transformation module 62
Figure 24: Interfaces for the Contract negotiation module ............................................................ 63
Figure 25: Interfaces for the Trustworthiness prediction module .................................................... 63
Figure 26: Interfaces for the Security property determination module ........................................ 63
Figure 27: Interfaces for the Security verification module ................................................................ 64
Figure 28: Interfaces for the Security policy monitoring module .................................................. 64
Figure 29: Interfaces for the Service threat monitoring module ...................................................... 64
Figure 30: Interfaces for the Monitor trustworthiness module ...................................................... 64
Figure 31: Interfaces for the Notification module ........................................................................ 65
Figure 32: Interfaces for the Threat repository module ................................................................... 65
Figure 33: Interfaces for the Aniketos marketplace ....................................................................... 65
Figure 34: Interfaces for the Threat response recommendation module ........................................ 66
Figure 35: Interfaces for the Secure composition planner module ................................................ 66
Figure 36: Interfaces for the Training material module .................................................................... 66
Figure 37: Interfaces for the Community support module ............................................................ 66
Figure 38: Aniketos design time service specification process (cf. Figure 5) .................................... 67
Figure 39: Aniketos design time service discovery and selection process (cf. Figure 6) ................. 67
Figure 40: Aniketos design time service validation and contract establishment (cf. Figure 7) ........ 68
Figure 41: Aniketos design time service assembly and deployment process (cf. Figure 8) ............... 69
Figure 42: Aniketos runtime service provision process (cf. Figure 10) ........................................... 69
Figure 43: Aniketos runtime service validation process (cf. Figure 11) ........................................... 70
Figure 44: Aniketos runtime recomposition (cf. Figure 12) ............................................................ 70
Figure 45: Aniketos runtime reconfiguration (cf. Figure 13) ......................................................... 71
Figure 46: Aniketos runtime monitoring (cf. Figure 14) .................................................................. 72
Figure 47: Service end user processes (cf. Figure 15) .................................................................... 73
Figure 48: Processes for enriching the Aniketos platform at runtime (cf. Figure 16) ...................... 74
Figure 49: Scenario handling process in WP1 ................................................................. 93

**List of tables**

Table 1. Domain model elements ......................................................................................... 7
Table 2. Explanation of general concepts in the composite service domain ......................... 9
Table 3. Explanation of specialised Aniketos concepts ......................................................... 11
Table 4. Scenario overview .................................................................................................... 13
Table 5. Analysis diagram model elements ......................................................................... 19
Table 6. Project requirements .................................................................................................. 42
Table 7. Description of items in the information model ......................................................... 53
Table 8. Descriptions of components in the Aniketos platform .............................................. 57
Table 9. Descriptions of environment components ............................................................... 60
Aniketos is a diverse project with several work packages working in parallel in order to achieve common objectives. In order to start this work with a mutual sense of direction, we needed a high-level blueprint of the overall technology development within the project. The role of this document is to be a common point of reference throughout the project.

The structure if this document is based on best practices for architecture descriptions, showing complementing views with different abstraction levels. The scenarios and requirements contained within these views have been collectively defined and revised through several iterations in this first project phase.

The first viewpoint shows the context of the Aniketos platform, meaning role and interaction in its environment. The environment includes both stakeholders and other systems. This includes a domain model, scenario descriptions, an environment description and business processes. Based on this, the next viewpoint presents the current state of the project requirements. This is followed by a component viewpoint that describes the information, subsystems and interfaces within the Aniketos platform itself. The document has two appendices; the first one a snapshot of the project glossary, and the second one an elaborated version of the scenario descriptions.

We consider the architecture to be a live document that will evolve and be aligned with the ongoing work in the project. A revised version of it will be delivered as D1.5 in month 33 of the project.
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.

Figure 1: Goal: establish and maintain security and trustworthiness in composite services

Aniketos is about establishing and maintaining trustworthiness and secure behaviour in a constantly changing service environment. The project aligns existing and develop 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 deliverable describes the initial blueprint of the technology development within Aniketos. It is based on best practices for architecture descriptions, showing complementing views with different
abstraction levels. The scenarios and requirements contained within these views have been guiding us and contributed to a common understanding on the goals and vision of Aniketos.

An initial draft of the architecture was developed within the very first months of the project, and has continuously undergone updates and extensions based on the comments and needs from the various work packages. Our intention is that this architecture is not written in stone, but will continue to evolve alongside the research and technology development of this project. This development will be documented in D1.5 – Final Aniketos architecture and requirements specification which is due in M33 of the project.

1.3 Structure of this document

The structure of this document is based on an architecture description framework named Arcade[1], which again is based on ANSI/IEEE 1471-2000: Recommended Practice for Architecture Description of Software-Intensive System.

ANSI/IEEE 1471-2000 has the following definition of architecture and architectural description:

- **Architecture:** The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.
- **Architectural Description (AD):** A collection of products to document an architecture.

The sections in this document are called viewpoints. Viewpoints are used to create a view. The view consists of one or more models that describe and present different aspects related to structure and behaviour for our system.

The five different Arcade viewpoints are explained in the following excerpt:

- **Context Viewpoint:** The purpose with the context view is to describe all aspects of the Target System’s environment, which is of importance to be able to document all the interfaces between the Target System and its environment, and what the Target System is intended to do in its environment.
- **Requirements Viewpoint:** The purpose of the requirement view is to document all specific requirements related to the Target System.
- **Component Viewpoint:** The purpose of the component view is to identify and document specific physical or logical components. Component descriptions should be purely functional, described by their data, interfaces and functionality. Note that existing or predefined hardware- or software-units can be treated as components and included as components in the component view.
- **Distribution Viewpoint:** The purpose of the distribution view is to describe the logical distribution of software and hardware components. The distribution view shows if some components cannot be separated and if any must be separated.
- **Realisation Viewpoint:** The purpose of the realisation view is to describe any constraints on how the target system’s components should be implemented and deployed into its environment.

This deliverable consists of the three first viewpoints of Arcade. The two latter are more technology dependent and therefore a part of the more detailed design within WP2-5. This document also contains a short conclusion and two appendices; a snapshot of the project glossary and elaborated scenario descriptions.
1.4 Relationships with other deliverables

This initial architecture is the common reference point for the other technical work packages, defining the high-level view and starting point for the more detailed planning and design. D1.1 – Consolidation of existing work has influenced a lot of the architectural decisions since one of our goals is to make use of existing technology as much as possible.

D1.2 gives an abstract view of the Aniketos platform design. The platform is specified in a more fine-grained manner in deliverables from WP2, WP3, WP4 and WP5. Certain inconsistencies are to be expected, due to different release dates of the deliverables. Resolving such inconsistencies is a priority in the project work.

1.5 Contributors

SINTEF has had the major effort in the development of this document, but all project partners have contributed to it. We wanted an including process where we gathered Aniketos scenarios and requirements widely among the partners, and the different viewpoints have been subject to discussions and improvements from a great number of people within the project. PLUS led the work with an investigative stakeholder study and Tecnalia has had a special focus on the environment systems.
2 Context viewpoint

The purpose of the context viewpoint is to describe what the Aniketos platform is intended to do in its environment and how the interaction will be. The environment includes both stakeholders and other systems. We do this by initially explaining the key concepts from the environment and within Aniketos in a domain model (section 2.1). Section 2.2 follows with an overview of scenarios telling different tales on how we envision Aniketos will improve service composition, but also what can go wrong if we are not careful. Section 2.3 explains the environment we would typically find surrounding the Aniketos platform, and section 2.4 builds on this by mapping business processes to the environment and the Aniketos platform.

2.1 The domain of Aniketos

The domain model of Aniketos presents the key concepts and their relations. Its purpose is to give an understanding of the scope of the project domain among the project participants as well as outsiders to the project. In Figure 2, a UML class diagram is used as a graphical representation of the domain model. The concepts found in the blue boxes belong to the general domain of composite services, while the concepts in the green boxes are more related to the Aniketos objectives of adding trustworthiness and security to this domain. The notation we have used for the domain model is summarized in Table 1. In order to make the figure more readable only the key relations have been included in the figure.

<table>
<thead>
<tr>
<th>Table 1. Domain model elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class/Concept</td>
</tr>
<tr>
<td>Aggregate</td>
</tr>
<tr>
<td>Compose</td>
</tr>
</tbody>
</table>

ANIKETOS
Figure 2: A domain model showing central concepts and their relationships
Table 2 gives a brief description of the concepts that are most central in the project, but also general for the composite service domain. For most of these concepts we have therefore selected established definitions.

Table 3 gives a description of the concepts that need a more specialised explanation in the context of this project. For extended definitions and more terms, consult the Aniketos Glossary in Appendix A.

Table 2. Explanation of general concepts in the composite service domain

<table>
<thead>
<tr>
<th>Concept name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service (component)</td>
<td>A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description [2]. The Aniketos platform is limited to software services, also referred to as service components in this document. The services are provided through public interfaces which are software readable. Nevertheless, the social context in which the services operate is of high interest in this project, so the human activities and interaction are things we take into account when working with trustworthiness and security issues.</td>
</tr>
<tr>
<td>Composite service</td>
<td>A service visible to a service consumer (or agent) via a single interface and described via a single service description that is the aggregation or composition of one or more other services. These other services can be atomic services, other composite services, or a combination of both [2]. A synonymous term is Federated service. Composite services consisting of services offered by several providers are the main focus of Aniketos. We anticipate a shift towards these kinds of services for the Future Internet and we are working with challenges that belongs to composite services.</td>
</tr>
<tr>
<td>Service developer</td>
<td>A service developer is a person able to create, modify and/or assemble services so that they can be offered by a service provider. Related terms are service composer, service designer and service creator.</td>
</tr>
<tr>
<td>Service provider</td>
<td>A service provider is a participant that offers a service that enables some capability to be used by other participants [2]. In Aniketos, a service provider is responsible for and offers services to service consumers at runtime. The term service host is often used about the same concept, though a service host is usually only responsible for the service infrastructure (network, processing power, memory) and not the service itself.</td>
</tr>
<tr>
<td>Service consumer</td>
<td>A service consumer is a participant that interacts with a service in order to realize the real world effect produced by a capability to address a consumer need [2]. A service consumer may mediate the service to another consumer; thereby have the role of provider as well.</td>
</tr>
<tr>
<td>Service specification</td>
<td>A service specification is an abstract representation of a service, typically defining the goals, functionality, behaviour, requirements and policies for the service. A service specification is usually the result of a service design. Languages commonly used for abstract specifications are given in [3], ranging from OWL-S, WS-BPEL, formal languages [4], 1st order logic and graph modelling languages. Aniketos contributes to service specifications by enabling information about socio-technical security requirements to be described. Examples are trust relationships and required security properties for service compositions.</td>
</tr>
<tr>
<td>Concept name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Service descriptor</strong></td>
<td>A service description (aka. service descriptor) is an artifact, usually document-based, that defines or references the information needed to use, deploy, manage and otherwise control a service. This includes not only the information and behavior models associated with a service to define the service interface but also includes information needed to decide whether the service is appropriate for the current needs of the service consumer. Thus, the service description will also include information such as service reachability, service functionality, and the policies and contracts associated with a service [2]. In Aniketos a service descriptor works like an agreement template [5], offering one or many options that will be manifested in the agreed service contract.</td>
</tr>
<tr>
<td><strong>Service contract</strong></td>
<td>A service contract is a composition of functional metadata and a set of policies, such as security constraints, access restrictions for user groups, transport and service level agreements, charging, etc [6]. This is an agreement between entities offering (service provider) and requiring a service (service consumer), regarding the purpose and consequences of the interaction [7]. Service level agreements (SLAs) are one commonly used category of contracts [2].</td>
</tr>
<tr>
<td><strong>Template</strong></td>
<td>A template can be a part of a service specification, and used as a blue print for the composite service in order to discover service components and assemble them. Templates describe the outline of activities needed to solve a problem, and are parameterized with respect to some variables so the generic template can be configured and customized for a specific instance based on the users’ current requirements and preferences [8]. Templates have the advantage of being reused and extendible [3].</td>
</tr>
<tr>
<td><strong>Discovery</strong></td>
<td>The act of locating a machine-processable description of a network-related resource that may have been previously unknown and that meets certain functional criteria. It involves matching a set of functional and other criteria with a set of resource descriptions. The goal is to find an appropriate service-related resource [9]. Aniketos mainly focuses on discovery based on security and trustworthiness.</td>
</tr>
<tr>
<td><strong>Assembly</strong></td>
<td>Assembly is the process of creating a runnable composite service instance based on an abstract service specification. As a part of this process, binding is used to associate protocol or data format information with an abstract entity like a message, operation, or portType [10]. In order to create an assembly you need to have performed discovery of service components in order to populate the service template and get a service composition plan.</td>
</tr>
<tr>
<td><strong>Adaptation</strong></td>
<td>Adaptation is needed when there is a mismatch between the quality of the services provided by the application and the user needs, or between the application needs and the context. Adaptation may lead to a degradation or enhancement of the quality of the provided services (QoS). For instance, the functional richness of an application may be modified, or the response time of a service may be reduced [11]. We consider adaptation to be a concept related to self-management and dynamic behaviour, in contrast to evolution where humans are more directly involved in the loop and there are new releases at regular intervals. Examples of adaptation are reconfiguration of how the services are used when a user moves into a hostile environment, leading to a change in the overall functionality of the composite service, or recomposition (see below).</td>
</tr>
<tr>
<td><strong>Recomposition</strong></td>
<td>A recomposition is a specialisation of adaptation where the services in a composite service are rearranged and/or replaced. After a recomposition the composite service should still fulfil the service specification. A recomposition can only occur for composite services.</td>
</tr>
</tbody>
</table>
### Table 3. Explanation of specialised Aniketos concepts

<table>
<thead>
<tr>
<th>Concept name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Environment</strong></td>
<td>The environment is a wide term referring to the surroundings of the service. This may include people, devices, software and conditions that influence or interact with the service. This term should not be confused by the definition used in relation to the architectural description of the Aniketos platform (see section 2.3).</td>
</tr>
</tbody>
</table>
| **Security policy** | *Policies prescribe the conditions and constraints for interacting with a service [2].*  
A security policy is a set of rules (or principles) that direct how a system (or an organization) provides security services to protect sensitive and critical system resources [12].  
In Aniketos, the relevant security policies need to be represented as a part of service specifications. This implies developing a way of defining and representing security policies that fit the purpose of this project. |
| **Security requirement** | A security requirement is a detailed requirement that implements an overriding security policy [13], and should express what is to happen in a given situation, as opposed to what is not ever to happen in any situation [14].  
In Aniketos, security requirements should be expressed as a part of the service specification. This implies developing a way of defining and representing security requirements that fit the purpose of this project. |
| **Security contract** | A security contract is a part of or addition to a service contract.  
Security contracts are vital in Aniketos as a means to identify what is required by a service consumer and what is offered by a service provider with respect to security properties and trustworthiness. These contract requirements can be used for discovery and filtering of relevant services before the assembly. The terms of an operational security contract must be monitored in order to ensure that they are kept over time. |
| **Trust relationship** | Trust is a concept that lies at the intersection of several domains, including sociology, psychology, law, economics, ethics, and computer science [15], hence it requires a socio-technical approach. Following the definitions from [16], trust is a quantifiable relation between two agents (humans or software systems), and we may differ between functional trust, which is the belief in an entity’s ability (and willingness) to carry out or support a specific function on which the relying party depends, and referral trust, which is the belief in an entity’s ability to recommend another entity with respect to functional trust. Another distinction is direct trust (where the trustor trusts the trustee directly without intermediaries) and indirect trust (based on opinions from third parties).  
In Aniketos, a trust relationship needs to be established between a service consumer (i.e. the trustor) and a service provider (i.e. the trustee). This can be based on direct and/or indirect trust. The object of the trust (i.e. the trustum) is formalised in a service (see term above) security contract (see above), which specifies the security policy (see above) at stake.  
The strength of the relationship should be based on the trustworthiness (see below). |
| **Trustworthiness**  | Security decision with respect to extended investigations to determine and confirm qualifications, and suitability to perform specific tasks and responsibilities (as cited in [17]).  
In Aniketos, trustworthiness is something that can be computed, measured or cognitively estimated in order to evaluate to what degree a service should be trusted. Trustworthiness is associated to metrics (see below), measuring one or a set of properties. |
<table>
<thead>
<tr>
<th>Concept name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Metric</strong></td>
<td>A metric provides a measurable quantification for a given property. In relation to trustworthiness this can be a numerical value associated to e.g. a reputation level, the strength of an evidence, or the economical value of a promise/guarantee.</td>
</tr>
<tr>
<td><strong>Validation</strong></td>
<td>Validation is a process intended to establish the soundness or correctness of a construct [12]. In the case of Aniketos, validation is the process of ensuring that the instantiation of the composite service has the right goals and requirements according to the service specification. Validation may be used pre-deployment by a service developer or during runtime adaptation/recomposition.</td>
</tr>
<tr>
<td><strong>Verification</strong></td>
<td>Verification is derived from the Latin word for “true” [12], and is the process of checking that the goals and requirements are fulfilled.</td>
</tr>
<tr>
<td><strong>Security property</strong></td>
<td>A security property is a characterization, attribute or specification of a service that realise a security requirement or policy. E.g. service A offers encrypted communication using a 1024-bit RSA key, only stores data locally for 24 hours or it only allows one instance of an identity to be logged in concurrently.</td>
</tr>
<tr>
<td><strong>Vulnerability</strong></td>
<td>A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy [12]. In Aniketos, vulnerabilities that arise due to compositions are of particular interest, such as elevation of privileges or information leakage due to poor compositions. If a vulnerability is detected within a service, this would typically reduce the trustworthiness of the service until it has been repaired or mitigated.</td>
</tr>
<tr>
<td><strong>Monitoring</strong></td>
<td>Monitoring in Aniketos is the process of checking that security contracts are fulfilled over time, particularly if there has been a change. This process involves verification. Also, monitoring is used to detect vulnerabilities and to discover attacks on a service, e.g. by making use of intrusion detection systems or dynamic testing tools found in the environment (see Table 2). Monitoring may register a change (see below).</td>
</tr>
<tr>
<td><strong>Threat</strong></td>
<td>A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm [12]. A threat is realised by exploiting vulnerabilities in a service. In Aniketos we are specifically considering threats that could harm compositions and ways to mitigate these. The presence of a threat to a system could trigger an alert (see below). The expected loss associated to a threat is called risk [12].</td>
</tr>
<tr>
<td><strong>Countermeasure</strong></td>
<td>An action, device, procedure, or technique that meets or opposes (i.e., counters) a threat, a vulnerability, or an attack by eliminating or preventing it, by minimizing the harm it can cause, or by discovering and reporting it so that corrective action can be taken [12]. In Aniketos we want the security properties to be realisations of countermeasures and thus mitigate vulnerabilities. A countermeasure can for instance be a template or security pattern for service composition.</td>
</tr>
<tr>
<td><strong>Alert</strong></td>
<td>In Aniketos, an alert is a kind of notification that could result in a service adaptation/recomposition. An alert can be sent after a security incident has been detected (reactively) or in order to make a service protect itself of something that potentially might happen (proactively).</td>
</tr>
<tr>
<td><strong>Change</strong></td>
<td>A change in Aniketos is a wide term, e.g. referring to an update of the security properties of a system, when the environment becomes more hostile or new threats suddenly becomes more probable, or a drop in the trustworthiness in one of the services being part of a composition. A change could either strengthen or weaken the trustworthiness of a service, and would typically send out an alert to inform the affected entities.</td>
</tr>
</tbody>
</table>
2.2 Aniketos scenarios

As part of task 1.3 in WP1 a set of practical use case scenarios for the Aniketos project have been developed, and as described in the DoW:

“This task will define how Aniketos is envisioned to improve secure service composition by describing practical use case scenarios of the Aniketos platform. Based on these scenarios, a set of detailed methodical and functional requirements will be derived that the other technical WPs must consider as guidance and success criteria.”

The scenario development has been essential for a good start in the project since they define the project context and function as a mine for requirements. We have been using collaborative brainstorming involving all project partners in order to create the scenarios. Appendix B gives an overview of the scenario handling process and contains the full scenario descriptions. Table 4 gives an overview of these scenarios with a very short summary so that the interested reader can seek further details in Appendix B. The scenarios with black text on white background represent the normal situations, while the ones with white text on black represent exceptions and threats, meaning things we do not want to happen.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>Service selection at design-time</td>
<td>Service discovery with the Aniketos marketplace.</td>
</tr>
<tr>
<td>1B</td>
<td>Manipulation of trust properties</td>
<td>Misuse of trust mechanism.</td>
</tr>
<tr>
<td>2A</td>
<td>Keeping the service end user informed</td>
<td>Letting the service end user determine privacy requirements at runtime.</td>
</tr>
<tr>
<td>2B</td>
<td>End user gets annoyed by confirmation requests</td>
<td>Too many end user confirmations will make the user ignore warnings.</td>
</tr>
<tr>
<td>3A</td>
<td>Secure behaviour monitor of a composite service</td>
<td>Detection of errors in a service composition and provision of warning.</td>
</tr>
<tr>
<td>4A</td>
<td>Specification of a Aniketos service</td>
<td>Service announcement using the Aniketos marketplace.</td>
</tr>
<tr>
<td>5A</td>
<td>Trust handover at runtime</td>
<td>Transitive trust between service users for a specific service.</td>
</tr>
<tr>
<td>6A</td>
<td>End user accesses services supplied by different service providers with their own identity-based security policies</td>
<td>Less need for multiple authentications when using composite services.</td>
</tr>
<tr>
<td>7A</td>
<td>Trustworthy services composition with the minimum disclosure of identity related information</td>
<td>Privacy management within a composite service.</td>
</tr>
<tr>
<td>8A</td>
<td>Federated Identity in Next Generation Networks (NGN): telecommunication services</td>
<td>Identity management and contracts between service providers.</td>
</tr>
<tr>
<td>8B</td>
<td>Leveraging the NGN operators’ role in IdP</td>
<td>Loss of useful customer data for service providers.</td>
</tr>
<tr>
<td>9A</td>
<td>Federated Identity in Next Generation Networks (NGN): discovery of IdP services</td>
<td>Sharing of identity data within a composite service.</td>
</tr>
<tr>
<td>ID</td>
<td>Name</td>
<td>Summary</td>
</tr>
<tr>
<td>----</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>10A</td>
<td>Design time verification of security properties</td>
<td>Verifying security for a composite service, end user feedback.</td>
</tr>
<tr>
<td>10B</td>
<td>Human factor failure: unable to understand formal proof</td>
<td>Verification feedback makes no sense to end user.</td>
</tr>
<tr>
<td>11A</td>
<td>End user makes (some) decisions about services used</td>
<td>End user service composition.</td>
</tr>
<tr>
<td>11B</td>
<td>Information and accountability lost</td>
<td>Lack of traceability for both service end user and service provider.</td>
</tr>
<tr>
<td>12A</td>
<td>Novice Aniketos platform users</td>
<td>Introduction to service composition with Aniketos support.</td>
</tr>
<tr>
<td>12B</td>
<td>Novice Aniketos users in trouble</td>
<td>Aniketos seems too complicated for a novice service developer.</td>
</tr>
<tr>
<td>13A</td>
<td>Security guidelines followed</td>
<td>Security policies are used to discover appropriate service components.</td>
</tr>
<tr>
<td>13B</td>
<td>Security guidelines compromised</td>
<td>Service developer is unable to match security policy with security properties used to describe service components.</td>
</tr>
<tr>
<td>14A</td>
<td>Payment provider scenario (component developer POV)</td>
<td>Service components have dynamic security rating according to their current vulnerabilities.</td>
</tr>
<tr>
<td>14B</td>
<td>Insufficient automated security evaluation</td>
<td>Misuse of security rating in order to give a false impression of provided service component.</td>
</tr>
<tr>
<td>15A</td>
<td>Runtime service recomposition (end-user POV)</td>
<td>Recomposition takes place when a critical vulnerability is found.</td>
</tr>
<tr>
<td>15B</td>
<td>Runtime recomposition without synchronisation</td>
<td>Failure of data migration during service recomposition.</td>
</tr>
<tr>
<td>15C</td>
<td>Runtime recomposition with synchronisation</td>
<td>Calendar data is stolen by replacing a legit service component with a malicious one.</td>
</tr>
<tr>
<td>16A</td>
<td>Stock exchange service</td>
<td>Runtime service component replacement due to discovered vulnerability.</td>
</tr>
<tr>
<td>16B</td>
<td>Stock exchange service – missing preference property</td>
<td>Unsatisfactory property in replacement component.</td>
</tr>
<tr>
<td>17A</td>
<td>Trustworthy Web scenario (service provider POV)</td>
<td>Trustworthiness estimations of service components obtained from Aniketos.</td>
</tr>
<tr>
<td>18A</td>
<td>Trustworthy Web scenario (end-user POV)</td>
<td>End user defines trustworthiness threshold for composite service.</td>
</tr>
<tr>
<td>19A</td>
<td>Futuristic home automation scenario (end-user POV)</td>
<td>End user buys a service component for his smart house based on security and trust properties.</td>
</tr>
<tr>
<td>20A</td>
<td>Web shop payment provider (end-user POV)</td>
<td>End user is assured about the security of a specific component in composite services.</td>
</tr>
<tr>
<td>ID</td>
<td>Name</td>
<td>Summary</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>20B</td>
<td>Web shops – perception-of-trust issues</td>
<td>False sense of security due to a malicious composite service using a trusted service component.</td>
</tr>
<tr>
<td>21A</td>
<td>Navigation (end-user POV)</td>
<td>Runtime end user composition based on trustworthiness.</td>
</tr>
<tr>
<td>22A</td>
<td>Service replacement at runtime</td>
<td>A service component becomes unavailable and is replaced by another.</td>
</tr>
<tr>
<td>23A</td>
<td>Landing plan</td>
<td>Runtime discovery and binding of service components due to extraordinary circumstances.</td>
</tr>
<tr>
<td>23B</td>
<td>Landing plan arrives too late</td>
<td>Runtime recomposition takes too much time, composite service not ready in time.</td>
</tr>
<tr>
<td>24A</td>
<td>Bank loan</td>
<td>Privacy requirements used for service discovery.</td>
</tr>
<tr>
<td>25A</td>
<td>Multi-user service for ambient assistant living</td>
<td>End user composition and tailoring of composite service for next-of-kin.</td>
</tr>
<tr>
<td>25B</td>
<td>Compromise of privacy</td>
<td>Information leakage due to unrestricted access control.</td>
</tr>
<tr>
<td>26A</td>
<td>Auction service</td>
<td>Two way security contract negotiations with trust requirements.</td>
</tr>
<tr>
<td>27A</td>
<td>Environment monitoring sensor network</td>
<td>Runtime change in security requirements leads to reconfiguration and avoidance of less trusted network nodes.</td>
</tr>
<tr>
<td>28A</td>
<td>Architecting and integrating a secure system</td>
<td>Aniketos tooling support used in combination with an existing architecture framework for modelling and specifying a system of system and associated contracts.</td>
</tr>
<tr>
<td>29A</td>
<td>Designing and implementing a secure sub-system</td>
<td>The creation of a critical sub-system that conforms to the overall specifications and contracts.</td>
</tr>
<tr>
<td>30A</td>
<td>Transparent contextual service composition</td>
<td>Client-side service (re)composition for a roaming employee.</td>
</tr>
<tr>
<td>30B</td>
<td>Dan doesn’t want to disclose his identity</td>
<td>Different identification requirements for roaming service users.</td>
</tr>
<tr>
<td>31A</td>
<td>Renting a movie</td>
<td>An end-user accesses services supplied by different service providers and one of these has not the required trust level.</td>
</tr>
<tr>
<td>32A</td>
<td>Dynamic service analysis and adaptation</td>
<td>Runtime security analysis reveals potential information leakage.</td>
</tr>
<tr>
<td>32B</td>
<td>Adaptation impacts functionality</td>
<td>Adaptation can potentially lead to denial of access to critical data.</td>
</tr>
<tr>
<td>33A</td>
<td>Composing services across insecure networks</td>
<td>Analysis of insecure message exchange within a composite service.</td>
</tr>
<tr>
<td>ID</td>
<td>Name</td>
<td>Summary</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>34A</td>
<td>Policy Reconciliation</td>
<td>Automatic service adaptation according to security policies.</td>
</tr>
<tr>
<td>34B</td>
<td>Policy negotiation</td>
<td>Reduction of security properties during negotiation can open up to an attack.</td>
</tr>
<tr>
<td>35A</td>
<td>Annotating images</td>
<td>Elevation of required data privileges due to combination of data from several services.</td>
</tr>
<tr>
<td>35B</td>
<td>Access level inflation</td>
<td>Unwanted increase of required data privileges.</td>
</tr>
<tr>
<td>36A</td>
<td>Making Aniketos available to stakeholders</td>
<td>Business opportunities based on the Aniketos marketplace and up-to-date trustworthiness estimates.</td>
</tr>
<tr>
<td>37A</td>
<td>Trust-Driven Composition of Services</td>
<td>The need of calculating trustworthiness for a composite service.</td>
</tr>
<tr>
<td>38A</td>
<td>Productive usage of IDE integrated tools and plug-ins</td>
<td>Explains the need for proper documentation and support for the tooling.</td>
</tr>
<tr>
<td>39A</td>
<td>Modelling and Reasoning about secure and trustworthy services</td>
<td>Using goal modelling as a mean to define commitments and perform reasoning related to trust and security.</td>
</tr>
<tr>
<td>40A</td>
<td>Different levels of authentication</td>
<td>End user requirements on authentication are matched with services.</td>
</tr>
<tr>
<td>40B</td>
<td>Providing Unnecessary Information</td>
<td>Example of an unethical service provider that harvests credit card information.</td>
</tr>
<tr>
<td>41A</td>
<td>Handling multi-tenant systems</td>
<td>A composite service aggregates data from several sources.</td>
</tr>
<tr>
<td>42A</td>
<td>Cargo Dispatching</td>
<td>Service choreography enables the results of one service to influence another.</td>
</tr>
<tr>
<td>42B</td>
<td>Malicious routing service provider</td>
<td>A malicious service component can have a negative effect on the overall service.</td>
</tr>
<tr>
<td>43A</td>
<td>E-Mobility Management</td>
<td>A composite service example involving several service providers.</td>
</tr>
<tr>
<td>44A</td>
<td>Payback Card</td>
<td>A composite service must provide information back to an associated service for a service component.</td>
</tr>
<tr>
<td>44B</td>
<td>Malicious service provider</td>
<td>An example malicious composite service that makes use of legit service components.</td>
</tr>
<tr>
<td>45A</td>
<td>Service Provider reacts to changes in the infrastructure requiring (semi-automatic) changes in the security configuration of an offered service</td>
<td>Balancing security and functionality when replacing a service component in a composite service.</td>
</tr>
<tr>
<td>46A</td>
<td>Billing for usage of a composite service</td>
<td>Gives an example on how different service billing systems can be combined for a composite service.</td>
</tr>
<tr>
<td>ID</td>
<td>Name</td>
<td>Summary</td>
</tr>
<tr>
<td>----</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>47A</td>
<td>Emergency Service Coordination Exercise</td>
<td>Evaluating the runtime performance of a static composite service defined at design-time.</td>
</tr>
<tr>
<td>48A</td>
<td>(Re)Composing services for time-critical financial trading</td>
<td>Gives an example of balancing trustworthiness with other properties, such as response time and price.</td>
</tr>
<tr>
<td>48B</td>
<td>Disaffected employee sabotages the policy infrastructure</td>
<td>Sabotage of a service component leads to a dynamic recomposition and notifications to the influenced parties.</td>
</tr>
<tr>
<td>49A</td>
<td>Unified communications (such as group chat) within virtual teams</td>
<td>Explains the need and mechanisms for federated-level policies for complex services.</td>
</tr>
<tr>
<td>49B</td>
<td>DDOS attack occurs on one of the federation partners</td>
<td>A service component is attacked and switches to stub functionality that can be provided to the consuming composite service.</td>
</tr>
<tr>
<td>50A</td>
<td>Reputation Based Trustworthiness</td>
<td>Recomposition based on change in the reputation of one of the service components.</td>
</tr>
<tr>
<td>50B</td>
<td>Trustworthiness Management Threats</td>
<td>Reputation mechanisms can be poisoned, and this need to be handled by filtering out malicious reports.</td>
</tr>
<tr>
<td>51A</td>
<td>Aniketos Training and tutorials</td>
<td>Gives example on how to provide training for the Aniketos platform.</td>
</tr>
</tbody>
</table>

2.3 The environment systems

This section explains what we consider to be the environment systems of the Aniketos platform, i.e., the systems that we typically will find surrounding and working together with it. The environment systems include the external design time tools and runtime components that the Aniketos platform needs to interface with, and thus influence the operation of the platform.

The environment systems can be grouped into three main components: A service composition framework, a service runtime environment and a service registry.

In the following we explain the tool stereotypes and tool examples for each of the environment components. Some of these tool examples are background brought into the project by the partners and other tools are external to the project in the sense the consortium has no direct control of them. All of them are candidates for interaction with the Aniketos platform. For more information on the examples, please refer to D1.1.

2.3.1 Service composition framework

The service composition framework lets developers create the composite service at design time. The tool stereotype typically includes the following mechanisms:

- **Service specification/planning mechanism**: Allows specifying how the composite service should be made up by underlying roles played by sub-services.
- **Service validation mechanism**: Checking that the composite service does the right thing according to the specifications.
- **Service discovery mechanism**: Finds service components able to play roles in the specification.

The service composition tool benefits from the interaction with the Aniketos platform because the developer is able to enrich the specification of the service with trustworthiness and security properties.
information (including threats) and create contracts based on this. Additionally, the service discovery
and validation of the specification can also be done in relation to these security and trustworthiness
requirements.

2.3.2 Service runtime environment
The minimum functionality of the service runtime environment is to execute the services, although the
tool stereotype includes all the following mechanisms (the examples are more elaborated in D1.1):

- **Service execution mechanism**: Runs the composite service. Examples: MUSIC studio and
  middleware by SINTEF (Aniketos partner), Apache Axis2, Apache CXF, Apache Synapse,
  Mule ESB, GlassFish, Swordfish, WSO2 Web Service Framework, .NET Framework, etc.

- **Service adaptation mechanism**: Adapts the composite service without changing the
  components. Examples: MUSIC studio and middleware by SINTEF (Aniketos partner) and
  DSAW platform.

- **Service recomposition mechanism**: Replaces one or more service components and can also
  include adaptation. Examples: MUSIC studio and middleware by SINTEF (Aniketos partner).

- **Service discovery mechanism**: Finds service component candidates during recomposition.

- **Service monitoring**: Monitors the inner workings of the composite service or individual
  services. Examples: Property and Policy Tools by LJMU (Aniketos partner), Platform for
  Run-time reconfigurability of security by ATOS (Aniketos partner).

- **Security monitoring**: Systems that are able to detect and possibly avoid malicious attacks to
  the environment which the composite services are running in. Examples: Intrusion detection
  systems, anti-virus systems, firewalls and runtime security testing tools.

- **Context sensor**: Discovers events or conditions that may influence the composite service or
  individual service components. Examples: MUSIC studio and middleware by SINTEF
  (Aniketos partner), Platform for Run-time reconfigurability of security by ATOS (Aniketos
  partner) and The Contextual Service Adaptation Framework.

- **Identity management**: A centralised or distributed mechanism that enables unique
  identification of services and providers. Examples: ED DIAMETER AAA by ELSAG
  (Aniketos partner).

Note that the Identity management module in D1.1 is not considered part of the Service runtime
environment but a separate module. This is because D1.1 was delivered prior to the final version of the
architecture.

2.3.3 Service registry
The service registry provides the service descriptions or specifications published by the service
providers. Basically it is usually a database that stores the duplets with the service identifier and the
URL where the executable service resides.

Examples of available technologies for service registry are: Composition Repository by ATC
(Aniketos partner), UDDI and Oracle Service Registry within Oracle SOA Suite. See D1.1 for more
information and references.

2.3.4 Integrated environments
As integrated environment we denote a service infrastructure that allows developers building,
deploying and managing services, and for this integrate service composition, registration and runtime
management in a single tool or tool suite, i.e. they are a merge between the above represented Service
composition framework, Service registry and Service runtime environment.

Examples of integrated environments are: Oracle SOA Suite, WSO2 SOA Platform and SAP
Enterprise Services Architecture. See D1.1 for more information and references.
2.4 Business to system mapping

In our Business to system Mapping models we present stakeholders, high-level work processes and the main information elements. We initially present a hierarchy of stakeholders that have been extracted from the scenarios (section 2.2) using ordinary UML actors and specialisations. We have used this “library” of stakeholders when creating the following work processes. In order to represent these work processes we have used a notation called Analysis diagram\(^1\), which is a simplified UML Activity diagram used to capture high level business processes and early models of system behaviour and elements. It is less formal than some other notations for business processes, but provides a good means of capturing the essential business characteristics. This notation is based on the Eriksson-Penker Business Modeling Profile [18] and we have summarized the model elements in Table 5.

Table 5. Analysis diagram model elements

<table>
<thead>
<tr>
<th>Actor/Stakeholder</th>
<th>Association</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Supply/invoke dependency</td>
</tr>
<tr>
<td>Process</td>
<td>Outcome dependency</td>
</tr>
<tr>
<td>Send</td>
<td>Decision</td>
</tr>
<tr>
<td>Receive</td>
<td>Note/comment</td>
</tr>
<tr>
<td>Information</td>
<td>Information item</td>
</tr>
</tbody>
</table>

2.4.1 Aniketos stakeholders

The Aniketos community consists of stakeholders who in one way or another will be influenced by the results of this project. Figure 3 presents a stakeholder breakdown (the arrow relationship between stakeholders is always inheritance/specialisation) along with a brief comment explaining the typical characteristics. These stakeholders are not mutually exclusive, for example a service provider might also be a service owner.

\(^1\) http://www.sparxsystems.com/enterprise_architect_user_guide/8.0/modeling_languages/analysisdiagram.html
Figure 3: Overview of various stakeholders for Aniketos

- **Aniketos community member**: A high level stakeholder that represents users able to benefit from Aniketos results. This stakeholder represents various kinds of Aniketos platform end users.

- **Aniketos authority**: Maintains the software and services part of the Aniketos platform, such as the marketplace and threat repository.

- **Aniketos platform contributor**: A user able to enrich the Aniketos platform with meaningful content, e.g., provide demonstration material, example services, guidelines or information in the threat repository.

- **Service consumer**: Invokes Aniketos compliant services. See explanation in the domain model.

- **Service end user**: End user of a service developed/composed with Aniketos platform.

- **Service provider**: A mediator both consumes and provides services, thus inherits from both service consumer and service provider. A service mediator often enriches the service with some value added feature.

- **Service mediator**: Provides a service to service consumers at runtime. See the domain model for further explanation.

- **Service owner**: A person or organisation that is behind the capabilities offered by the service. Would typically be liable responsible and make revenue from the service.

- **Service developer**: A high level stakeholder involved in the creation of a service.

- **Service designer**: Designs/specifies services using aided by the Aniketos platform.

- **Service implementor**: Someone implementing a service (from scratch) that can be used with the Aniketos platform, e.g., possible to discover and use in a composition.

- **(Re)composition agent**: Uses the Aniketos platform to compose a new service from existing and compliant services.

- **Third party**: Software which recomposes a composite service at runtime triggered by changes, e.g., in service requirements, execution environment, consumed services, or due to a change in the threat picture.

- **Third party able to act as a certifier of trust for stakeholders involved in a service invocation**.
Stakeholder study
This overview of various stakeholders for Aniketos was the starting point for an exploratory study on the characteristics, work practices and needs of these future users of the Aniketos platform. The study took place between February and May 2011 and was conducted as a part of a larger user-centred design process, which is focusing on investigating user requirements for Aniketos platform services and tools as a first step. The goal of this study was to provide exemplars of Aniketos stakeholders in order to inform the Aniketos platform design.

We recruited five people who are involved, to varying degrees, in security engineering for web services, and who have been considered as possible Aniketos stakeholders to participate in an exploratory, semi-structured interview. The interviews addressed characteristics of the interviewees’ daily work: typical tasks, work conditions, tools used, techniques and methods, work rules and policies, further education habits, skills, and their personal viewpoint about security. The recorded interview data were transcribed, summarized, and content analysed according to the following questions:

- Which Aniketos stakeholder group does the interviewee belong to?
- Which tasks does the interviewee have to fulfil?
- Which strategies does the interviewee make use of to fulfil his (security engineering) tasks?
- What relation does the interviewee have to security engineering/web services?
- What opinion/motivation/attitude does the interviewee have towards web service and (security) engineering issues in general?
- What abilities does the interviewee consider to be important/necessary for his profession?
- How does the interviewee maintain his level of knowledge (trainings, tutorials etc.)?

Based on the results of this analysis, we provide recommendations about what to consider when developing Aniketos platform services and tools for future platform users (stakeholders). These recommendations should be seen as initial findings, which need to be further investigated in follow-up studies (increasing the number of participants, with a broader range of characteristics).

In the following, each interviewee is briefly described by providing information about his profession, professional background and working experience.

**Interviewee #1** is a senior developer in an IT company with a focus on the public sector, who spends most of his working time on building applications and web services. He is one member in a team of two to three people, with one person – usually, but not always him – delegating the tasks to the other two. Interviewee #1 has been working as software developer for 6 years now, with a two years break in between, after which he changed the company. “I have not been involved in security engineering tasks or issues regarding the development or composition of web services so far; this is something completely new for me.”

**Interviewee #2** is a freelance software developer working full-time since 2007, who mainly implements applications and partly web service clients for his clients, who are mostly agencies. He also does consulting, but to a lesser extent than developing. Interviewee #2 attended a university of technology, where he also took a course on web services one and a half year ago. He says that he can’t tell much about the composition of web services and also doubts that “many people can tell about this.” He assumes web service composition to be a vision of the future, but he doesn’t think that many people are currently using this, “because the automated web service composition isn’t that advanced yet.”

**Interviewee #3** is a freelance senior consultant in a software development company with a focus on IT security. In #3’s opinion “senior consultant is the most general role in this domain.” and it fits very well what he is doing, because, “‘senior developer’ is too technical and ‘project manager’ wouldn’t...
specify it [the role] accurately.” His tasks cover everything from customer contact, project management, project controlling, process enhancement, technical architecture and system modelling to programming and testing. Interviewee #3 has been working within the IT domain for around 10 years and has been a professional software developer for 6 to 7 years. He once worked in the automobile industry, which is an industry with high security standards. “We combine or aggregate web services of different providers, and offer this as a web service. But composition, as it is taught, with four different levels, is not really part of our projects. In our software projects we have different levels, which interact with each other, but we don’t do web services and SOAs in the classical sense.”

Interviewee #4 is a freelance software architect working full-time for an IT company, which produces software solutions for the automotive industry. Once he started studying computer science but never graduated. Interviewee #4 has been working as freelance software architect for 10 years, changing his employers approximately every two years. He has been working for different banking institutions, an entertainment corporation in the gambling sector and a global powerhouse in electronics and electrical engineering. Interviewee #4 has done web service implementation and has been providing security for web services for several years. “Implementing a web service always means thinking about security, because if you are implementing a web service badly, I mean really badly, someone could erase the whole database behind it.”

Interviewee #5 is a software developer, working for an IT company which provides software solutions in the automotive sector. There he is mainly engaged in the fields of interface technology (web services) and Java architecture, especially doing a lot of coordination, integration, documentation and knowledge transfer work regarding software products and frameworks his colleagues have to use after he has installed and configured them. Interviewee #5 is the team leader of a small technical team, consisting of 4 team members. He also has to think about ways to provide security for the corporation he works for as well as their customers, when they have to work with external services. “You have different groups of people concerned with the different kinds of security, and regarding the security of applications and services I’m one of them.” Interviewee #5 has worked full-time since 1990. He studied (something similar to) mechatronics at a Swedish university.

The following part describes our recommendations for what to consider when developing Aniketos platform services and tools for future Aniketos platform users (stakeholders):

**High importance of teamwork and collaboration processes**

For all of the interviewees, teamwork and collaboration (internally, within companies, and externally, between people of different organisations) play an important role in daily work practices. For instance within a software development company with a strong security focus, it’s custom that several people are serially involved in security validations of initial concepts of systems during design time. Interviewee #3 reports that one colleague has a look at the document thoroughly for one day, and then another colleague has a focused look at the original document and the comments from the previous validation.

Interviewee #3 participates in meetings with his colleagues to think about how to perform security tests of software systems. Each team member examines a specific part and then they meet and discuss this in the team. These conjecture and refutation processes are considered as important by them, especially when they lack experience on specific issues. In interviewee #4’s company the handling of a task, especially a new task, is discussed in team to identify a good way to work on it. Also regarding time management the strategy is to calculate the amount of time needed for the task completion altogether. The collectively determined amount of time is a reference point the individual software architect can adjust to.

Additionally, according to interviewee #4, a security expert would come in “handy”. Due to the fact that no one in the company, interviewee #4 is working for, is a declared security expert, they are often confronted with unexpected challenges regarding security issues. Therefore collaborating with
someone with a sound knowledge on this field would be helpful. Moreover, collaboration in testing a web service client takes place when the software engineer needs to work with a contact person from the provider of the web service server to do the testing together and to obtain the final usage approval. Consequently, Aniketos platform services and tools, which are used to perform similar tasks, should enable an easy handover of work between co-workers and help them to better relate to each other’s tasks within a work process. Future Aniketos platform users could benefit from features, which support collaboration and communication between developers, consumers, providers and security experts.

Security modelling task: creative thinking and retrieval of personal knowledge about risks and threats
When interviewee #3 does security checks of diagrams made by an external system architect, he experiences this task as a playful, creative one: “In as much as I print out the diagrams, which are there [in the concept], or trace them again, leave parts of the diagrams out, play games with the diagrams about what could happen, what they [who made the diagrams] have not thought about.” Moreover, interviewee #3 explains that in his department, doing security checks (within consulting tasks) is based on the experience which the consultant has. There are check-lists available, “which our technical security people have made, because we implicitly have check-lists e.g. from Microsoft, a security catalogue, how it should be done correctly. We, in our case, have the experience, which is added, and which forms our checklists. In terms of that we are not that formal as our technical security colleagues.” Relying on own experiences is also important for interviewee #5. He and some of his colleagues, who have also experience in the field of web service development and implementation, have elaborated some kind of “Best practice”-guidelines regarding security issues, interface definitions etc.

In order to address these user needs, Aniketos platform tools and services, more specifically modelling tools, should help users of these tools in their creative thinking task. Additionally, because of the reliance on personal experience and knowledge, the modelling tools could help their users by providing access to a person-specific history of ways of problem solving in security modelling.

Clear definition and explanation of risk levels for risk assessment
One difficulty in assessing risk levels is related to clearly communicating and understanding the types of risks and their consequences. When interviewee #3 is doing security checks, he finds risk factors, and makes judgements about the level of the risk. They provide their customers with a risk list and a judgement of the level of each risk (3-4 risk levels), but they don’t add a financial value to these risks, because they “aren’t entitled to state a value”. “Then it’s about explaining these risks in a way that the severity level of a risk becomes clear to the consulting customer”, which is difficult to do, according to interviewee #3. He thinks it’s easier for the technical process colleagues, who say, “Ok, this is a one million Euro risk” to explain risk levels. Whereas, “we have to be creative in describing the risk, to make it explicit to the customer what we want to state. And not just red, yellow and green, or the like.”

One can conclude from this that attention has to be paid to how Aniketos platform services and tools define and explain risk levels. This may apply to threat alerts sent by an Aniketos platform service, which should provide meaningful risk level descriptions for the receiver of the alert.

Pervasive, on-demand exchange of knowledge
Interviewees have stressed the importance of continuous learning within the area of software (security) engineering. To be well suited for this profession, one should be eager for knowledge and of course interested in the domain. Interviewee #4 says: “A software engineer, who isn’t learning all his life, is lost. He simply has to continue learning.” Continuous learning is mostly accomplished by learning informally, rather than taking formal courses. All of the interviewees describe their knowledge acquisition as strongly related to a pervasive knowledge transfer via different ways. One way to
informally learn is to use company-internal knowledge resources, by talking to colleagues with specialist knowledge. For instance, interviewee #3 is working in a company, which is closely connected to a research centre (regarding work domain and physical location). He really appreciates that when he is unsure about tricky technical and security details, he doesn’t need to contact someone external, but can just ask a colleague.

Aside from this, further personal education is done via informal learning embedded pervasively in daily life activities. “In order to be able to detect security problems, you also have to watch the news in the morning for half an hour, get to know which IT problems are currently there.” Interviewee #3 thinks this is a more important way of getting further educated than attending trainings and going to conferences.

Moreover, knowledge acquisition is usually an on-demand search for very specific, problem-oriented information. For instance, interviewee #5 attended official trainings in the past, but meanwhile he seeks knowledge himself on a more demand-oriented basis (e.g. looking up information in books, newspapers, special interest magazines and especially the Internet), because “in the majority of cases trainings are a bit inefficient.” If he is doing “training” on his own, he can adapt to what he needs for his current work more accurately. The same applies to interviewee #4, who thinks that he can find the information provided in formal courses easily somewhere else.

Thus, due to the characteristics described beforehand, the user interfaces of Aniketos platform services and tools should support on-demand, easy to conduct information search within the context in which the knowledge deficit of a user occurs. Moreover, the provided information is up-to-date and efficiently retrieved and processed, e.g. by offering bite-size chunks of information.

**Learning security issues by observation**

Interviewee #3 thinks that in IT you learn most by observing how things work (and knowing cause and effect). He experienced this in the mentoring programmes for the junior developers. When the junior developers are made aware of something, then they immediately know what the problem is or what they should conclude from this. The same applies to security issues: “Within security, it is similar. When they (his colleagues in security research) invent new attacks and you can watch how the attacks work, then you probably won’t directly apply this knowledge, but you still keep it at the back of your mind and you know that you can or should take care about these issues (within your own work)”. Consequently, Aniketos platform services and tools related to knowledge transfer should explain and ideally demonstrate, by example, security issues as clearly and intuitively as possible.

**Webinars and trainings**

Some of the interviewees have participated in webinars/trainings (#1, 3, and 5) and most of them regard them as practical and useful, if they provide the following benefits:

- It is easy to register for the webinar and thus, there is no time-consuming registration processes.
- They provide access to information of diverse niche topics, which is otherwise not accessible.
- A sufficient amount of dates, when the webinar takes place, is offered and there is no fixed schedule, e.g. to be required to attend 40 meetings in the next 12 weeks. This would be important for interviewee #2, who is a freelancer and who has flexible working hours.
- There are downloads available after the event, thus you can watch the event afterwards. You can decide on your own, when to watch them.
- Samplings (“getting a taste”) of the trainings serve as decision guidance and enable you to decide quickly if the content fits your needs or not. Although interviewee #5 thinks it would be difficult for the providers to offer samples/extracts of their trainings in the forefront, he thinks that would facilitate the decision making process, if a training is worth attending it or not.

These requirements also apply to knowledge transfer via the Aniketos platform.
Critical view on automated testing tools

One problem, which Aniketos platform services and tools might have to face, is related to acceptance of automated security testing tools. For instance, interviewee #3 personally doesn’t think as much of tools, “which can (nearly fully) automatically check software projects and which finds a catalogue of 800,000 to 1 Million security flaws”. For the infrastructure tests he is performing, he uses smaller, semi-automated tools to check e.g. if specific infrastructure components meet their requirements. These tools have an output like “we recommend you to also enable this and that, because this is often a problem”. “It’s not like this that you push a button and at the end a report comes out about what problem it could be. In reality, for the documents we receive to check, this a creative task.”

However, interviewee #3 thinks this class of static checker tools is helpful in some way (“the more help one gets the better”), but the reasons for his critical view on them are:

a) He thinks these tools don’t scale because “in reality, these tools just go through checklists” and “I always have the impression this only works for the most simple applications, and not for a real project.”

b) He thinks that these tools are not as competent as developers in familiarizing with projects and in understanding how components relate to each other: “It’s difficult enough for a developer to familiarize with diverse projects, so I don’t think that a software product can even do this in a similar manner, and if it is able, then to find the security problems in the projects.”

c) High costs of these tools, but limited usefulness.

d) The tools (or the marketing departments) suggest a false sense of security, which the tools don’t have. They should not claim “This is a secure product, because it was checked with this tool”.

In order to enhance future trust in and adoption of Aniketos platform services and tools, it’s recommended to care for scalability issues and make realistic, credible claims about the capacity of the services and tools.

Trust and standardization issues limit optimism about automated web service composition

Aside from a critical view on automated testing tools, people are taking a critical look at automated web service composition itself. Interviewee #2 says that he can’t tell much about automated composition of web services, but he believes that including appropriate services during run-time isn’t that advanced yet. He thinks that this is a vision of the future, it should become this way, but he doesn’t think that many people are currently using this, because some web service standards are still problematic. According to his knowledge, especially for SOAP services there are a bunch of sub-standards, of which some are not so perfectly implemented. Moreover, the impression of interviewee #2 not being able to judge the trustworthiness of replaced web services during run-time hinders him to think about realistic application scenarios of automated web service composition in security critical domains. This viewpoint underlines the importance of the Aniketos project itself to provide methods, tools and services for judging the trustworthiness of (composite) services.
2.4.2 Work process overview: Design-time

Aniketos does not seek to create a whole new process of creating composite services; a lot of work has been done in this field already that must be exploited. Figure 4 shows typical work processes related to design-time service composition. There are other variations of this figure, with more/fewer process boxes (e.g. testing has been omitted since we focus on validation at design-time) and where the order might be a little bit different (e.g. contracts can be established after service assemble), but we think this one is a fairly generic version to which we can relate to. Note that loops have been omitted (e.g. if validation fails it will be necessary to go back one or several steps).

![Diagram of typical processes related to design-time service composition](image)

*Figure 4:* Typical processes related to design-time service composition

Only the service developer stakeholder has been included here, but as seen in Figure 4, this one can be composed of several others with more specific roles. Additionally, the service owner and service end user would typically be involved in giving input to the service specification process.

The following figures in this section show how Aniketos processes complement the generic ones. We are using the same colour scheme as previously to differ between the environment and Aniketos.
Figure 5 extends the Specify service process (see service specification in the domain model of section 2.1). Aided by a Socio-technical Security Modelling tool a service developer will model and analyse trust and security requirements for the service, and this outcome will supply the overall service specification. As a part of this process, the Aniketos platform services can be used to lookup and add information about threats and countermeasures relevant for the service.

![Diagram showing the Specify service process related to Aniketos]

**Figure 5: Aniketos related to the specify service process**
Figure 6 shows that a service specification is used in order to perform service discovery and selection. Aniketos extends this initial process by allowing discovery/filtering to take trustworthiness and security properties (which can include required countermeasures) into consideration. The service provider must enable his service to be available for such discovery using the Aniketos marketplace, while the actual values are things that the Aniketos platform services will provide.

Based on the available components, Aniketos platform service may also support suggest one or several secure compositions.

Figure 6: Aniketos related to the discovery and selection of service components
After potential service candidates have been found, we need to validate that the composed service does what it is supposed to do. Figure 7 contains this general process, along with *Establish contracts* (which in many cases is not performed or possible today). The supporting process *negotiate contract* is optional (hence the *unbound* association between the two processes), meaning that there services that both have and have not negotiable terms (in the first case, this is something usually involving the *service provider, service owner* and *service consumer*). Aniketos supports the complementary processes of analysing that the composition fulfils the trust and security specification. If this is the case, these notions should also be part of the service contracts, and monitoring should be set up to ensure that these terms are kept over time and after a change.

![Figure 7: Aniketos related to service validation and contract establishment](image)

Figure 7: Aniketos related to service validation and contract establishment
Figure 8 is the final figure for the design-time phase of composition, where there is a handover from service developer to service provider. Aniketos is not directly involved in the actual design-time assembly/binding, but as the service is made “live”, we need to register what should trigger alerts/notifications with the Aniketos platform, as well as make the service available for discovery through the Aniketos marketplace.

![Figure 8: Aniketos related to service assembly and deployment](image-url)
2.4.3 Work process overview: Runtime

In Figure 9 we have moved to the runtime domain. The Provide service process is running continuously in the environment, and will at some point in time receive an alert from the Aniketos platform. This will indicate to the service provider that a service validation would be a wise thing to do. The service validation can have three outcomes:

- The service is OK and the alert was nothing to care about, go back to regular provision.
- The service is not OK, try and adapt with a reconfiguration (meaning keep the same service components but with some modifications).
- The service is not OK, try to recompose (replace service components).

In the two latter cases the service provider would normally do a new validation since there has been a change.

In the lowermost part of the figure we want to show that monitoring is also something that is continuously done in the environment by the service provider. If something out of the ordinary is detected, an alert can be sent to the Aniketos platform, which would route this message to the relevant receivers. For instance, if a service provider detects an intrusion he would have to notify consuming composite services if this is a contract requirement. A more detailed break-down of monitoring is given in Figure 14.
Figure 9: General processes related to runtime reaction to changes and monitoring
Figure 10 shows that as long as the service provider is providing a service he should have it announced in the Aniketos marketplace. This is necessary for the Aniketos platform in order to send him alerts, and also make it discoverable based on the service’s trust and security properties.
In Figure 11 we can see that the Aniketos platform supports the validate service process. Using the service composition plan and composition contracts we can check if the predicted trustworthiness and security properties (including implemented countermeasures) still hold against the commitments in the security contract(s).

Figure 11: Aniketos related to runtime service validation
The breakdown of the runtime recomposition in Figure 12 is similar to Figure 6, Figure 7 and Figure 8 for design-time. The Aniketos platform does not really care if the environment is invoking it from at design-time or runtime. One difference is that we have omitted Validate service performed by the environment here, since it already knows that something was wrong with the original composition, so the validation is done after the assembly instead. Depending on the recomposition strategy of the environment, this could be done at an earlier stage as well.
During a service adaption shown in Figure 13, Aniketos provides support by analysing the new configuration. This is simpler than recomposition since we do not have to perform discovery and assembly. We might have to update the contracts though, and this could influence the monitoring points and alert thresholds that the Aniketos platform should know of. As in Figure 7 we have an optional contract negotiation, but at runtime it would be more feasible to have an agent-based approach to this.

Figure 13: Aniketos related to runtime reconfiguration
During monitoring (Figure 14) a lot is left to the environment, but the Aniketos platform can be used to check for changes that would violate the security contract. If this should be done as a poll by the environment or if Aniketos should actively push changes is a design decision we are not able to determine here. Alerts can be sent to the Aniketos platform in order to notify affected parties or as an early warning.

Figure 14: Aniketos related to runtime monitoring
From the service end user’s point of view she would normally invoke a service using her service client platform, which could be for instance a Web-browser. Using the Aniketos platform it would be possible for her to check the security policy and trustworthiness of the services she is invoking and receive notification about changes affecting these properties.

Figure 15: Aniketos related to service end user processes
Figure 16 shows a number of processes related to how the Aniketos platform can be enriched:

- Service providers can announce their services in the Aniketos marketplace.
- An Aniketos authority is responsible for maintaining and improving software and services that make the Aniketos platform.
- Aniketos platform contributors are able to add reference material, such as developer guidelines and demonstrations, as well as threat information that can be used for service specifications.
- The general stakeholder Aniketos community member is for instance needed for the trustworthiness mechanism to work.

![Figure 16: Enriching the Aniketos platform at runtime](image-url)
3 Aniketos requirements viewpoint

In order to guide the project by validating that we are building the right things and to verify that we are building them right we have created a set of high-level requirements for the project. These originate from the scenarios described in section 2.2 (elaborated in Appendix B) and have undergone a refinement process involving WP leaders and the technical manager. The requirements are stored in a dedicated repository with the following attributes:

- **ReqID:** A unique identifier for each requirement constructed from the originating scenario and a counter, (e.g. S27A-3 denotes a requirement from scenario 27A, number three)
- **Title:** A short title of the requirement (also unique)
- **Description:** A description of the requirement (in order to fully understand the requirement it should be read in the context of the scenario).
- **Type:** The requirements types are based on FURPS [19]:
  - Functionality - Feature set, Capabilities, Generality, Security
  - Usability - Human factors, Aesthetics, Consistency, Documentation
  - Reliability - Frequency/severity of failure, Recoverability, Predictability, Accuracy, Mean time to failure
  - Performance - Speed, Efficiency, Resource consumption, Throughput, Response time
  - Supportability - Testability, Extensibility, Adaptability, Maintainability, Compatibility, Configurability, Serviceability, Installability, Localizability, Portability
- **Source:** Originating scenario, for instance Scenario 27A.
- **Target WP:** The requirements are project wide, but have all been assigned to one work package, making the work package leader responsible for analysing and providing feedback to the technical manager. Several of the requirements span over more than one work package, in these cases the comment field has been used.
- **Target phase:** Indicates in which phase of the project we are able to verify that the requirement has been (partly) fulfilled. The possible values are {1,2,3,4, post-Aniketos}.
- **Author:** Indicates who (which partner) formulated the initial requirement as a part of the scenario.
- **Related to:** Other requirements that (partly) cover the same things.
- **Verification method:** Indicates which method will or has been used to verify the fulfilment of the requirement. Options:
  - Inspection/review – A solution is textually described in a deliverable (e.g. an algorithm) that a person is able to evaluate.
  - Runtime testing – We are able to verify the fulfilment through testing of running software.
  - Formal proof – A formal method of verification is applied.
  - User feedback – End-users provide their evaluation/opinion on the fulfilment of the requirement.
- **Priority:** Indicates how important the requirement is in order to achieve the objectives of the project:
  - 1: Must have
  - 2: Should have
  - 3: Nice to have
- **How addressed:** An explanation on how the requirements have been fulfilled (free text).
• **Target:** The document, module or similar that meets the requirement.

• **Process status:** Identifies where we are in the Aniketos requirements handling process (described in a separate document). Possible values:
  o Identified by author
  o Filtered by technical manager
  o Analysed by WP leader
  o Endorsed

• **Fulfilment status:** Indicates whether the requirement is achieved or not in the current project phase. It is also possible to set the value as:
  o Rejected – The requirement is considered to be out-of-scope or not suitable for the project.
  o Obsolete – The requirement is not considered relevant any more.
  o Redundant – This requirement is already covered by another one.
  o Deferred post-Aniketos – We were not able to achieve this requirement in this project, but still think it is something valuable to do.

• **Changelog/comments:** Written explanations to changes and minor comments (e.g. relationship to other work packages).

Figure 17 shows a small screenshot (version 9, 30th of June 2011) from the actual requirements repository. We are continuously making updates to this in order to improve the requirements and to track their progress, and follow an iterative approach where reprioritizing and new additions reflect the research and technology development, as well as updated needs from the industry.

![Figure 17: Except from the requirements database](image)

In the initial phase of the project we elicited 150 requirements based on the scenario descriptions. The refinement process rejected 16 of these, mostly because they were a bit out of scope for the project or they were rather requirements to the environment. Additionally 11 requirements have been marked as redundant. Table 6 shows a snapshot of the project requirement where we have included the ReqID, Title, Description, Type, Source, Target phase and Priority attributes.
### Table 6. Project requirements.

<table>
<thead>
<tr>
<th>ReqID</th>
<th>Title</th>
<th>Description</th>
<th>Type</th>
<th>Source</th>
<th>Target phase</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>S27A-3</td>
<td>Balancing requirements</td>
<td>The platform must support the balancing of security requirements against other requirements, directed by policies defined by system owners/operators.</td>
<td>Functionality</td>
<td>Scenario 27A</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S28A-1</td>
<td>Compatible with environmental systems</td>
<td>To cover the complete system development cycle, Aniketos approaches and products should be compatible with at least one system of systems (SoS) architecture framework (AF), and one system/software modelling tools environment. AFs and modelling tools environments definitively not supported should be clearly identified, with the reason why.</td>
<td>Functionality</td>
<td>Scenario 28A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S28A-3</td>
<td>Elicitation of security needs</td>
<td>Aniketos shall support the elicitation of security needs.</td>
<td>Functionality</td>
<td>Scenario 28A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S29A-2</td>
<td>Model-based risk assessment</td>
<td>Aniketos shall support a model-based security risk assessment as a specific viewpoint in parallel to the mainstream system/software engineering.</td>
<td>Functionality</td>
<td>Scenario 29A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S30B-3</td>
<td>Aniketos modelling language</td>
<td>Aniketos should provide a modelling language that incorporates business level concepts (actors have goals and interact via contracts to achieve them).</td>
<td>Functionality</td>
<td>Scenario 31A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S30B-5</td>
<td>Service consumer policy</td>
<td>Aniketos has to allow service consumers to express their policy about service composition.</td>
<td>Functionality</td>
<td>Scenario 31A</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>S39A-1</td>
<td>Model elements</td>
<td>ANIKETOS should provide socio-technical modelling support for actors, roles, goals, context and social dependencies related to security, trust and risk properties.</td>
<td>Functionality</td>
<td>Scenario 39A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S39A-2</td>
<td>Model-based verification</td>
<td>Reasoning techniques should allow for verifying security, trust, and risk properties against models.</td>
<td>Functionality</td>
<td>Scenario 39A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S39A-3</td>
<td>Support SOA description</td>
<td>The Aniketos modelling support should provide input to a high-level architectural description of the service-oriented application.</td>
<td>Functionality</td>
<td>Scenario 39A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S39A-4</td>
<td>Roles for commitment</td>
<td>A composite service should be specified at the level of roles in terms of commitments. Commitments are input to define security contracts.</td>
<td>Functionality</td>
<td>Scenario 39A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S39A-5</td>
<td>Reasoning about security properties and trustworthiness</td>
<td>Reasoning techniques should assess security properties and the trustworthiness of the composite service architecture when it is populated with real service components.</td>
<td>Functionality</td>
<td>Scenario 39A</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S41A-1</td>
<td>Specify information flow</td>
<td>The Aniketos modelling support should be able to specify security requirements of information flow, so that information does not end up in the wrong places or can be aggregated in a way that compromises privacy.</td>
<td>Functionality</td>
<td>Scenario 41A</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S42A-1</td>
<td>Information restrictions</td>
<td>Aniketos should provide mechanisms to define restrictive usage of information or data.</td>
<td>Functionality</td>
<td>Scenario 42A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S44B-1</td>
<td>Separation of information</td>
<td>It should be possible to specify requirements to separation of information flow using the Aniketos platform.</td>
<td>Functionality</td>
<td>Scenario 44B</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>S45A-1</td>
<td>Verification and validation module</td>
<td>The design-time part of the Aniketos platform should provide a verification and validation module.</td>
<td>Functionality</td>
<td>Scenario 45A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S01B-1</td>
<td>Manipulation of trust properties</td>
<td>The Aniketos platform should be able to detect or avoid manipulation of trust properties (e.g. trustworthiness based on heuristics).</td>
<td>Functionality</td>
<td>Scenario 1B</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>ReqID</td>
<td>Title</td>
<td>Description</td>
<td>Type</td>
<td>Source</td>
<td>Target phase</td>
<td>Priority</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------</td>
<td>---------------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>S05A-1</td>
<td>Temporal trust</td>
<td>The Aniketos platform should provide a mechanism that allows end users to establish trust relationships between them for specific uses and/or time.</td>
<td>Functionality</td>
<td>Scenario 5A</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S06A-4</td>
<td>Circle of trust</td>
<td>The Aniketos platform will provide support to create circle of trust (trust transitivity) among several Services Providers.</td>
<td>Functionality</td>
<td>Scenario 6A and 7A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S07A-1</td>
<td>Trustworthy composition</td>
<td>Aniketos platform should enable trustworthy composition of services; meaning that the composer can select the most trustworthy among a set of different alternatives.</td>
<td>Functionality</td>
<td>Scenario 7A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S14A-2</td>
<td>Evaluate trustworthiness</td>
<td>The Aniketos platform should constantly evaluate all service components for trustworthiness.</td>
<td>Functionality</td>
<td>Scenario 14A</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S17A-1</td>
<td>Check trust credentials</td>
<td>The Aniketos platform should provide information about services providing trust credentials according to specified requests.</td>
<td>Functionality</td>
<td>Scenario 17A</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>S22-4</td>
<td>Platform allows rechecking of trust</td>
<td>The Aniketos platform should provide a mechanism that allows services to recheck their trust level.</td>
<td>Functionality</td>
<td>Scenario 22A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S23A-1</td>
<td>Trust based service discovery</td>
<td>The Aniketos platform should allow specifications about trustworthiness of information to be used for service discovery.</td>
<td>Functionality</td>
<td>Scenario 23A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S24A-1</td>
<td>Validation of security contracts</td>
<td>The Aniketos platform must support validation of security contracts to regulate and protect the use of data between services.</td>
<td>Functionality</td>
<td>Scenario 24A</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>S24A-2</td>
<td>Contracts through service chains</td>
<td>Security contracts must be able to operate through service chains.</td>
<td>Functionality</td>
<td>Scenario 24A</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>S26A-1</td>
<td>Predict trust levels</td>
<td>The Aniketos platform should be able to predict trust level of service components and the composed service.</td>
<td>Functionality</td>
<td>Scenario 26A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S26A-2</td>
<td>Negotiate contracts</td>
<td>The Aniketos platform should be able to negotiate contract, primarily through matching contract templates, with regard to security and trust requirements between service provider and consumer.</td>
<td>Functionality</td>
<td>Scenario 26A</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S28A-5</td>
<td>Contract representation</td>
<td>The Aniketos service trust contract shall be agnostic of whether the service will be rendered by people, equipment (hardware / software) or procedures.</td>
<td>Functionality</td>
<td>Scenario 28A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S28A-6</td>
<td>Contract states</td>
<td>The Aniketos service trust contract shall comprehend an attribute specifying its approval state. Note: possible states might be draft, proposed, approved, invalidated (e.g. due to design-time changes), failed (e.g. due to runtime non-compliance).</td>
<td>Functionality</td>
<td>Scenario 28A</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S28A-7</td>
<td>Model-based contracting</td>
<td>The Aniketos service trust contract shall be sufficiently explicit to support model-based contracting.</td>
<td>Functionality</td>
<td>Scenario 28A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S28A-8</td>
<td>Design-time contract testing</td>
<td>The Aniketos service trust contract shall be testable at design-time, meaning verifying both their format and content.</td>
<td>Functionality</td>
<td>Scenario 28A</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S37A-1</td>
<td>Provide information about trustworthiness</td>
<td>The Aniketos platform should provide a mechanism that allows services to provide information about their trustworthiness, but this measurement must be objective.</td>
<td>Functionality</td>
<td>Scenario 37A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S37A-2</td>
<td>Monitor trust according to contract</td>
<td>The Aniketos platform should provide a mechanism that is able to manage contract in order to monitoring them for measuring trust.</td>
<td>Functionality</td>
<td>Scenario 37A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ReqID</td>
<td>Title</td>
<td>Description</td>
<td>Type</td>
<td>Source</td>
<td>Target phase</td>
<td>Priority</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>-------------------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>S37A-3</td>
<td>Trust feedback</td>
<td>The Aniketos platform should be able to manage trust feedbacks.</td>
<td>Functionality</td>
<td>Scenario 37A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S37A-4</td>
<td>Certify trustworthiness</td>
<td>The Aniketos platform should be in charge of certifying the trustworthiness of the services.</td>
<td>Functionality</td>
<td>Scenario 37A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S40A-2</td>
<td>Trust between end user and service provider</td>
<td>The Aniketos platform should provide means for establishing a trust relationship between the end user and the service provider(s).</td>
<td>Functionality</td>
<td>Scenario 40A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S50A-1</td>
<td>Reputation as a trust property</td>
<td>Aniketos platform should include reputation status in determining trustworthiness level.</td>
<td>Functionality</td>
<td>Scenario 50A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S50A-2</td>
<td>Trust changes</td>
<td>Aniketos platform should facilitate composition, recomposition and adaptation of composed services dynamically due to trust changes.</td>
<td>Functionality</td>
<td>Scenario 50A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S02A-1</td>
<td>Unwanted recomposition</td>
<td>The Aniketos platform should detect when a recomposition would break the security requirements requested by an end user (consumer).</td>
<td>Functionality</td>
<td>Scenario 2A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S02A-2</td>
<td>Runtime requirements change</td>
<td>It should be possible to change the security requirements to a composite service at runtime.</td>
<td>Functionality</td>
<td>Scenario 2A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S10A-1</td>
<td>Proving requirement fulfilment</td>
<td>Aniketos platform should provide a way of proving requirement fulfilment within a composition.</td>
<td>Functionality</td>
<td>Scenario 10A</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>S10B-1</td>
<td>Explanation of proofs</td>
<td>There should be information available to the developers stating what results from a theorem prover means.</td>
<td>Functionality</td>
<td>Scenario 10B</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>S13A-1</td>
<td>Form of security metric</td>
<td>The data of the security-related metrics of the basic services needs to be in a form, which makes it easy for the service developer to check if security guidelines are met.</td>
<td>Usability</td>
<td>Scenario 13A</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S26A-3</td>
<td>Detect contract violations</td>
<td>The Aniketos platform should be able to detect contract violation at runtime.</td>
<td>Functionality</td>
<td>Scenario 26A</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>S28A-4</td>
<td>Mapping between security needs and contracts</td>
<td>The mapping between security needs and security contracts should be &quot;provable&quot;.</td>
<td>Functionality</td>
<td>Scenario 28A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S28A-9</td>
<td>Runtime contract monitoring</td>
<td>The Aniketos service security contract shall be &quot;monitorable&quot; at run-time.</td>
<td>Functionality</td>
<td>Scenario 28A</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>S32A-2</td>
<td>Suggest security solutions</td>
<td>The Aniketos platform should be able to suggest and apply solutions to security problems, automatically and dynamically update the service composition to match.</td>
<td>Functionality</td>
<td>Scenario 32A</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>S33A-1</td>
<td>Remove security elements</td>
<td>The Aniketos platform should provide a way to analyse service compositions and remove unnecessary security elements for the sake of efficiency/usability.</td>
<td>Performance</td>
<td>Scenario 33A</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>S34A-1</td>
<td>Multiple contract analysis</td>
<td>The Aniketos platform should support the analysis of multiple service contracts simultaneously.</td>
<td>Functionality</td>
<td>Scenario 33A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S34A-2</td>
<td>Policy reconciliation</td>
<td>It should be possible to perform policy reconciliation in order to establish a unified policy that applies to composed services.</td>
<td>Functionality</td>
<td>Scenario 33A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S34B-1</td>
<td>Same level of overall security</td>
<td>Policy reconciliation or negotiation should guarantee the same level of overall security.</td>
<td>Functionality</td>
<td>Scenario 34B</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>ReqID</td>
<td>Title</td>
<td>Description</td>
<td>Type</td>
<td>Source</td>
<td>Target phase</td>
<td>Priority</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
<td>-------------------------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>S45A-2</td>
<td>Service descriptions in the marketplace</td>
<td>The Aniketos marketplace should use a service description language that provides information for allowing a security analysis of the offered services and compositions thereof.</td>
<td>Functionality</td>
<td>Scenario 45A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S45A-3</td>
<td>Fixing after verification/validation</td>
<td>The verification and validation module of the Aniketos platform should be able to provide instructions for fixing (simple) security problems.</td>
<td>Functionality</td>
<td>Scenario 45A</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>S47A-1</td>
<td>Performance impact</td>
<td>The Aniketos platform should be aware, at composition design time, of the performance impact of different service composition security policies.</td>
<td>Performance</td>
<td>Scenario 47A</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>S47A-2</td>
<td>Mitigate performance costs</td>
<td>The Aniketos platform should be able to mitigate the performance costs of service compositions, either by removing redundant security elements or by providing feedback to service providers that their policy infrastructure needs to be improved.</td>
<td>Functionality</td>
<td>Scenario 47A</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>S47A-3</td>
<td>Notification about unfulfilled access criteria</td>
<td>In case a composition cannot be found that meets access control performance criteria, the Aniketos platform should raise this issue with the service composer.</td>
<td>Functionality</td>
<td>Scenario 47A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S48A-1</td>
<td>Alternative compositions for service users</td>
<td>The Aniketos platform should provide alternative service recompositions, annotated with sufficient security and trustworthiness information for the service users to choose between them.</td>
<td>Functionality</td>
<td>Scenario 48A</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>S48A-2</td>
<td>Non-functional characteristics along with security context data</td>
<td>The Aniketos platform should provide hooks to enable composed service consumers to measure service non-functional characteristics like performance, together with security context data, upon which to make informed business decisions.</td>
<td>Functionality</td>
<td>Scenario 48A</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>S48B-1</td>
<td>Security performance</td>
<td>The Aniketos platform should keep track of composed service security performance. Overall service time is the sum of setup and usage times. The setup time defines the security performance.</td>
<td>Performance</td>
<td>Scenario 48B, Scenario 49B</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>S48B-2</td>
<td>Notify about lack of service quality</td>
<td>The Aniketos platform alerts service providers if their service quality falls below a threshold, such that an alternative service, with better Trust/performance is chosen instead.</td>
<td>Functionality</td>
<td>Scenario 48B</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S48B-3</td>
<td>Share security performance</td>
<td>Given that setup time information is available from the environment, the Aniketos platform can share service composition performance with trusted service partners, if serious composed service security performance problems arise.</td>
<td>Functionality</td>
<td>Scenario 48B</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>S01A-2</td>
<td>Notify about trust change</td>
<td>A composite service should be able to subscribe to notifications if some of the services they consume change trustworthiness.</td>
<td>Functionality</td>
<td>Scenario 1A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S03A-1</td>
<td>Notify about vulnerabilities</td>
<td>The Aniketos platform should support publishing detected vulnerabilities to the relevant parties.</td>
<td>Functionality</td>
<td>Scenario 3A</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>S03A-2</td>
<td>Vulnerability binding</td>
<td>Aniketos platform should allow a vulnerability to be tied to a specific service component or a composition.</td>
<td>Functionality</td>
<td>Scenario 3A</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>S11B-1</td>
<td>Traceable messages</td>
<td>Messages and notifications should be traceable.</td>
<td>Functionality</td>
<td>Scenario 11B</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S14A-3</td>
<td>Notify about change in security level</td>
<td>The Aniketos platform should notify all service developers (along with the composite service providers) if the security level of a service changes.</td>
<td>Functionality</td>
<td>Scenario 14A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ReqID</td>
<td>Title</td>
<td>Description</td>
<td>Type</td>
<td>Source</td>
<td>Target phase</td>
<td>Priority</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>-------------------------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>S19A-1</td>
<td>Threat binding to installed components</td>
<td>The Aniketos platform should inform trace threat to affected component and where they are installed in order to provide updates.</td>
<td>Functionality</td>
<td>Scenario 19A</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S25A-1</td>
<td>Inform when security or trust has been reduced</td>
<td>The Aniketos platform should inform service provider when composition has reduced security functionality or trustworthiness.</td>
<td>Functionality</td>
<td>Scenario 25A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S27A-1</td>
<td>Detect changes in threat level</td>
<td>The platform must support the detection of changes in threat level and security requirements at runtime.</td>
<td>Functionality</td>
<td>Scenario 27A</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>S32A-3</td>
<td>Alert when no security match can be found</td>
<td>In case a composition can’t be amended to match a security policy, action needs to be taken (e.g. by alerting an administrator).</td>
<td>Functionality</td>
<td>Scenario 32A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S50B-1</td>
<td>Distributed manipulation of trust</td>
<td>Aniketos platform management of trustworthiness should be resilient against attacks from a significant proportion of users who have malicious intent through providing dishonest ratings.</td>
<td>Functionality</td>
<td>Scenario 50B</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S01A-1</td>
<td>Publish trust properties</td>
<td>The Aniketos platform should provide a mechanism that allows services to provide information about their trust properties.</td>
<td>Functionality</td>
<td>Scenario 1A and 20A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S01A-3</td>
<td>Trustworthiness ranking</td>
<td>The Aniketos platform should be able to rank services based on their trustworthiness.</td>
<td>Functionality</td>
<td>Scenario 1A</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>S02B-1</td>
<td>Guidelines to composition with end user involvement</td>
<td>The Aniketos platform should provide information on how to create composite services that will not annoy end users with exasperating confirmation requests related to trust and security.</td>
<td>Usability</td>
<td>Scenario 2B</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>S04A-1</td>
<td>Developer center</td>
<td>The Aniketos platform should have a developer center containing information about requirements and methods of creating Aniketos compliant services. It should also include a kind of forum + download area (to download Aniketos modules/tools).</td>
<td>Usability</td>
<td>Scenario 4A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S04A-2</td>
<td>Upload specifications</td>
<td>The Aniketos platform should provide a mechanism allowing organizations to upload security and trust specific service specifications. It should be impossible to upload a security and trust specification that does not match the corresponding (functional) service description.</td>
<td>Functionality</td>
<td>Scenario 4A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S04A-3</td>
<td>Usage conditions</td>
<td>The Aniketos platform should provide a mechanism to describe usage conditions of each service.</td>
<td>Functionality</td>
<td>Scenario 4A</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S05A-2</td>
<td>Trust awareness</td>
<td>Services can be made aware of trust relationships between involved entities. A trust relationship is defined in the domain model.</td>
<td>Functionality</td>
<td>Scenario 5A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S11B-1</td>
<td>End user choosing</td>
<td>The end user should be able to influence the choice of services within a composite service (environment requirement).</td>
<td>Functionality</td>
<td>Scenario 11A</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S11B-2</td>
<td>Event logging</td>
<td>Events should be logged, so that e.g. lost messages can be found.</td>
<td>Functionality</td>
<td>Scenario 11B</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S12A-1</td>
<td>Aniketos overview</td>
<td>From the Aniketos platform user interface the service developer should be provided with an overview about the Aniketos platform features already at the entry point.</td>
<td>Usability</td>
<td>Scenario 12A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ReqID</td>
<td>Title</td>
<td>Description</td>
<td>Type</td>
<td>Source</td>
<td>Target phase</td>
<td>Priority</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>------------</td>
<td>-------------------------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>S12A-2</td>
<td>Aniketos infrastructure</td>
<td>The Aniketos platform should provide information about its infrastructure and supportive services for design-time and run-time to the service developer in a structured, easy, accessible and efficient way.</td>
<td>Usability</td>
<td>Scenario 12A</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>S12A-3</td>
<td>Aniketos supportive services</td>
<td>The Aniketos platform should provide additional, more detailed, and easy-to-find information about its supportive services for design-time and run-time to the service developer.</td>
<td>Usability</td>
<td>Scenario 12A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S14A-4</td>
<td>Rank services</td>
<td>The Aniketos platform should be able to rank services based on an objective security rating.</td>
<td>Functionality</td>
<td>Scenario 14A</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>S18A-1</td>
<td>End user trust threshold</td>
<td>The Aniketos platform should inform an end user if a composite service is below the required trustworthiness.</td>
<td>Functionality</td>
<td>Scenario 18A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S20A-1</td>
<td>Interface for checking trustworthiness</td>
<td>Aniketos should offer an interface through which end users can easily check the trustworthiness of any component.</td>
<td>Functionality</td>
<td>Scenario 20A</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S20A-2</td>
<td>Interface for informing about used components</td>
<td>Aniketos should offer an interface through which end users can stay updated about the components used by a certain service.</td>
<td>Functionality</td>
<td>Scenario 20A</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>S22A-5</td>
<td>Marketplace allows service providers to renegotiate trust</td>
<td>The Aniketos Marketplace should provide a mechanism that allows Service Provider to renegotiate or update the trustworthiness of the services.</td>
<td>Functionality</td>
<td>Scenario 22A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S23B-1</td>
<td>Predicted computation needs</td>
<td>It must be possible to make predictions on the complexity and computation needs for features/functionality offered by the Aniketos platform.</td>
<td>Performance</td>
<td>Scenario 23B</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S23B-2</td>
<td>Required response time for selecting algorithm</td>
<td>The Aniketos platform should be able to choose service discovery and evaluation algorithm based on maximum required response time.</td>
<td>Performance</td>
<td>Scenario 23B</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>S28A-10</td>
<td>Aniketos independent of development methodology</td>
<td>The Aniketos methodology shall be agnostic of the processes &amp; framework the company uses for its mainstream engineering.</td>
<td>Usability</td>
<td>Scenario 28A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S28A-2</td>
<td>Methodology for using Aniketos</td>
<td>The Aniketos methodology shall guide the system-developer's organisation to build its security risk management plan, especially in terms of when and how to apply the different Aniketos risk management methods and tools, and how to monitor the results.</td>
<td>Usability</td>
<td>Scenario 28A</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>S30B-2</td>
<td>Aniketos infrastructure</td>
<td>Aniketos should provide an infrastructure to compute and store trust values.</td>
<td>Functionality</td>
<td>Scenario 31A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S30B-4</td>
<td>Business level interaction</td>
<td>Aniketos has to support service composition that reflects and is compliant with business level interaction.</td>
<td>Supportability</td>
<td>Scenario 31A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S31A-5</td>
<td>Dynamic management of circle of trust</td>
<td>The Aniketos platform should allow dynamically management of the status of the circle of trust.</td>
<td>Functionality</td>
<td>Scenario 31A</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>S32A-1</td>
<td>Analyse federated services</td>
<td>The Aniketos platform should provide a way to analyse federated services at run-time and discover security properties, even when those services belong to different organisations.</td>
<td>Functionality</td>
<td>Scenario 32A</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>ReqID</td>
<td>Title</td>
<td>Description</td>
<td>Type</td>
<td>Source</td>
<td>Target phase</td>
<td>Priority</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------------</td>
<td>------</td>
<td>--------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>S36A-1</td>
<td>Pay for trust determination</td>
<td>Determination of trust levels should be associated with online payment services.</td>
<td>Functionality</td>
<td>Scenario 36A</td>
<td>post-Aniketos</td>
<td>3</td>
</tr>
<tr>
<td>S36A-2</td>
<td>Register for Aniketos support</td>
<td>The Aniketos platform should enable for registration and role-based access to design and runtime support (information, methods, tools and functionality).</td>
<td>Functionality</td>
<td>Scenario 36A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S36A-4</td>
<td>Advertisement on the platform</td>
<td>The Aniketos platform should provide an advertising service related to trust and security properties for service components.</td>
<td>Functionality</td>
<td>Scenario 36A</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>S38A-1</td>
<td>Support service developers</td>
<td>The Aniketos design-time support modules need to come along with sufficient documentation, training, and support for the service developers.</td>
<td>Usability</td>
<td>Scenario 38A</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S07A-4</td>
<td>Minimum identity data</td>
<td>Service Providers should be allowed to manage the minimum amount of identity related information they need in order to provide their services;</td>
<td>Functionality</td>
<td>Scenario 7A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S25A-1</td>
<td>Service composition based on security properties and trust level of environment</td>
<td>The Aniketos platform should provide a mechanism that allows services to be composed according to their security properties and the trustworthiness of the environment.</td>
<td>Functionality</td>
<td>Scenario 25A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S25A-2</td>
<td>Inform end user during composition</td>
<td>A composite service (using the Aniketos platform) should be able to decide when to inform end user and when not to in cases of changes in service composition.</td>
<td>Functionality</td>
<td>Scenario 25A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S26A-4</td>
<td>Runtime component replacement</td>
<td>The Aniketos platform should be able to change service component (dynamic service composition) based on template/goal according to the security and trust situation at runtime.</td>
<td>Functionality</td>
<td>Scenario 26A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S27A-2</td>
<td>Adaptation</td>
<td>The platform must support adaptation of services and service compositions due to changes in security requirements.</td>
<td>Functionality</td>
<td>Scenario 27A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S31A-3</td>
<td>Single sign on</td>
<td>End users should be able to access several services offered by different service-providers without authenticating more than once.</td>
<td>Usability</td>
<td>Scenario 31A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S31A-4</td>
<td>Circle of trust</td>
<td>The Aniketos platform should provide support to create a circle of trust among several Services Providers.</td>
<td>Functionality</td>
<td>Scenario 31A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S43A-1</td>
<td>Discovery based on trustworthiness and security requirements</td>
<td>Aniketos platform should be able to select available services according to required trustworthiness and security criteria.</td>
<td>Functionality</td>
<td>Scenario 43A</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S38A-2</td>
<td>Guarantee IDEs</td>
<td>The longevity of Aniketos design-time support modules for IDEs has to be guaranteed.</td>
<td>Supportability</td>
<td>Scenario 38A</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>S51A-1</td>
<td>API documentation</td>
<td>Generation of API documentation for Aniketos components should be supported by a documentation generator tool (such as Doxygen or Javadoc). The code comments must be appropriately formatted for these tools to work; a common and general awareness of this is necessary for all WP1-5 contributors.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>ReqID</td>
<td>Title</td>
<td>Description</td>
<td>Type</td>
<td>Source</td>
<td>Target phase</td>
<td>Priority</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------------</td>
<td>------</td>
<td>--------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>S51A-2</td>
<td>Exercise understandability</td>
<td>Practical exercises in training material should be easily comprehensible. Exercises should be well commented, and supported with supplemental documentation, targeting those who develop and maintain exercises, those who present them, and those who accomplish them.</td>
<td>Usability</td>
<td>Scenario 51A</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S51A-3</td>
<td>Training related to case studies</td>
<td>Practical exercises in training material should all refer to the same scenarios and themes, such as the Aniketos case studies.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S51A-4</td>
<td>Virtual machines for training</td>
<td>Practical exercises in training material should be developed in a pre-set virtual machine environment.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>S51A-5</td>
<td>Developers guide</td>
<td>A Developer's Guide (consisting of detailed API documentation and code examples for each Aniketos component) should be available for Aniketos developers and kept up to date.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>S51A-6</td>
<td>Aniketos User's Guide</td>
<td>A User's Guide should be available for end-users. This manual should cover the usage of Aniketos features from an end-user perspective and kept up to date.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>S51A-7</td>
<td>Distance learning material for individual use</td>
<td>Aniketos distance learning materials (e.g. webinars) should be available for developers and end-users with various levels of expertise. Such material should be accompanied by an information sheet as well that contains information such as their purpose, content, methods, and required time investment.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S51A-8</td>
<td>Classroom training material coverage</td>
<td>Each Aniketos component should be covered by the classroom training curriculum.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S51A-9</td>
<td>Complementary materials for classroom trainings</td>
<td>Classroom trainings should be supported by complementary materials for the training creators, the presenters and the attendees.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S51A-10</td>
<td>Feedback from training participants</td>
<td>Participants in Aniketos trainings (both classroom training and distance learning) should be able to provide feedback.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>S51A-11</td>
<td>Classroom training quality</td>
<td>Each classroom training must score above 75% on the assessment form that is handed out at the end of the training.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>S51A-12</td>
<td>Adherence to educational practices</td>
<td>All provided learning material should take educational traditions and practices into account.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>S51A-13</td>
<td>Training subject consistency</td>
<td>Practical exercises in training material should all refer to the same scenarios and themes.</td>
<td>Supportability</td>
<td>Scenario 51A</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
4 Component viewpoint

The purpose of the component view is to describe the system in terms of its subsystems and information objects, and to document how subsystem interaction and information processing is carried out in order to provide the desired behavioural effect.

4.1 System information model

The system information model describes the central information elements in the system. We’ve divided the system information model into three parts, one focusing on specifications, one on messages and the final on entities involved in a contract.
Figure 18: Information model for Aniketos and its environment

Figure 18 shows the information items used by Aniketos platform. Here, the focus is the information related to specifications and contracts. The same colour scheme as that for the component model is applied for Aniketos specific information items and items related to the environment. Some particular information items are explained in Table 7.
In Figure 19, the focus lies on information messages that travel within Aniketos (between components) and between the environment and the Aniketos platform.
Figure 20 shows involved entities in a contract. A contract needs to specify which service it is defined for, what are the roles of the contract entities (consumer, provider, overseer), and what type of entity these are (TTP, person, organisation).

Table 7. Description of items in the information model

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security specification model</td>
<td>Represents the security specification model specifying the social technical security requirements. It contains information about security properties, trust relationships and threat. The model is created using the modelling tool and then transformed to different representations, e.g. Aniketos compliant specification, contract template, service composition plan, security descriptor. The model is only contained in the modelling tool, but the representations are available and used in different life cycle stages and cross different platform components. The most important / complete one (i.e. reflecting the model) is the Aniketos compliant specification.</td>
</tr>
<tr>
<td>Aniketos compliant specification</td>
<td>This is a specialized service specification extended with security and trust characteristics. It contains information about service composition plan, security requirements and is the basis for deploying a service.</td>
</tr>
<tr>
<td>Security property</td>
<td>Provides information about a security property, such as its type and description.</td>
</tr>
<tr>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Security descriptor</td>
<td>Provides information about security properties for a registered service.</td>
</tr>
<tr>
<td>Trust relationship</td>
<td>Describes a trust relationship with information about the trustee, trustor, or a trusted third party.</td>
</tr>
<tr>
<td>Threat</td>
<td>Identifies a threat with a threat repository id.</td>
</tr>
<tr>
<td>Threat response recommendation / countermeasures</td>
<td>Describes the countermeasures related to a threat.</td>
</tr>
<tr>
<td>Contract template</td>
<td>A template used to create security contract at design time or at runtime.</td>
</tr>
<tr>
<td>Security contract</td>
<td>Represents a security contract with information such as the contract compliance status. A contract contains a set of terms (see below).</td>
</tr>
<tr>
<td>Terms</td>
<td>Defines the terms of a security contract, including trust relationship and rules for trust relationship and security properties.</td>
</tr>
<tr>
<td>Involved party</td>
<td>Describes the involved parties for a security contract.</td>
</tr>
<tr>
<td>Security property rule</td>
<td>Describes rules for security properties.</td>
</tr>
<tr>
<td>Trust relationship rule</td>
<td>Describes rules for trust relationship. It specifies the underlying trust model and consists of trustworthiness metric based on the trust model.</td>
</tr>
<tr>
<td>Trustworthiness</td>
<td>Is represented in a metric of a specific trust model.</td>
</tr>
<tr>
<td>Service descriptor</td>
<td>A description of a service includes information about service type, interface description, function description and security descriptor. It is offered by a service provider and used to match against a service query during service discovery</td>
</tr>
<tr>
<td>Registration</td>
<td>A message for registering a service for alert. It specifies the types and threshold of the alerts as well as the notifier.</td>
</tr>
<tr>
<td>Contract negotiation request</td>
<td>A request message for contract negotiation. It provides Aniketos compliant specification as the basis for negotiation.</td>
</tr>
<tr>
<td>Contract negotiation result</td>
<td>A response message for contract negotiation. If negotiation is successful, a security contract is generated; otherwise, negotiation failure is returned.</td>
</tr>
<tr>
<td>Discovery request</td>
<td>A request message for service discovery based on security requirements. Aniketos compliant specification is used as input.</td>
</tr>
<tr>
<td>Discovery response</td>
<td>A response message for service discovery containing a set of services matching the request.</td>
</tr>
<tr>
<td>Security verification request</td>
<td>A request message for security verification with service composition plan as input.</td>
</tr>
<tr>
<td>Security verification result</td>
<td>A response message for security verification indicating the result as OK or not_OK.</td>
</tr>
<tr>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Trustworthiness prediction request</td>
<td>A request message for prediction of trustworthiness based on the trust relationship specified.</td>
</tr>
<tr>
<td>Trustworthiness prediction response</td>
<td>A response message providing the trustworthiness predicted for a trust relationship.</td>
</tr>
<tr>
<td>Selection support request</td>
<td>A request message for suggesting secure composition based on a composition template.</td>
</tr>
<tr>
<td>Selection support response</td>
<td>A response message for secure composition suggestion with a set of populated composition plan.</td>
</tr>
<tr>
<td>Marketplace announcement</td>
<td>A message for registering a service in the Aniketos marketplace.</td>
</tr>
<tr>
<td>Notification</td>
<td>A message for alerts indicating service change, context change, trust level change, threat level change, security property change or contract violation.</td>
</tr>
</tbody>
</table>

### 4.2 System decomposition model

The purpose of the system decomposition model is to describe how the system is divided into different subsystems/components, and how these are related to form a coherent whole. We also identify the main points of integrations with other systems.
Figure 21 shows a graphical representation of the decomposition model using a UML component diagram. We have described the logical components found in the service tier of both the environment (blue area) and the Aniketos platform itself (green area). Notice that the Aniketos platform has not been modelled as black box with a single interface, but rather a collection of loosely coupled components that can be accessed directly by the environment or internally within the platform. Some relationships have been left out in order to preserve the readability of the figure. Descriptions of these components can be found in Table 8 and Table 9.
### Table 8. Descriptions of components in the Aniketos platform

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Related WP/tasks</th>
</tr>
</thead>
</table>
| Socio-technical security modelling tool | This is a graphical modelling tool integrated with the Aniketos platform. It supports a socio-technical security modelling language, which supports the analysis and identification of socio-technical security requirements of the services created with the aid of Aniketos.  
**Related work process:** See *Model and analyse trust and security requirements for the service* in Figure 5. | WP1 (task 1.2).                   |
| Model transformation module        | This module transforms the socio-technical security specification models into representations for other modules, such as contract requirements and bindings between the service and relevant threats.  
Gives input to WP2, 3 and 4, developed as a part of WP5 |                                  |
| Trustworthiness prediction module  | This module predicts the current trustworthiness of a service based on the trust relationship and trust model(s) in use. It should also be able to estimate the overall trustworthiness of a composite service based on the suggested composite assembly.  
**Related work processes:** See *Discovery based on trust and security requirements* in Figure 6 and Figure 12. Also, *Analyse secure composition of service* in Figure 11. | WP2 (task 2.1 and 2.2).           |
| Monitor trustworthiness module     | This module enables runtime monitoring of trustworthiness based on the mechanisms and metrics defined according to the trust model. If changes in trustworthiness defy the required levels in a contract, then the affected parties are notified using the notification module. This module could invoke the trustworthiness prediction module. | WP2 (task 2.1)                     |
| Contract negotiation module        | This module enables the matching of provided and required contract offerings. The transformed socio-technical security model defines what is required of trust relationships and security properties. The provided security properties for the composition/service are found by using the *security property determination module*. The provided trustworthiness is found by using the *trustworthiness prediction module*.  
**Related work processes:** See *Discovery based on trust and security requirements* in Figure 6 and Figure 12. Also, *Figure 7 if real negotiations take place.* | WP2 and WP3                       |
| Security property determination module | This module allows the determination of the security properties of individual services (task 3.1) as well as the overall properties for the federated composed services (task 3.2). It is used both at design time and at runtime.  
**Related work process:** See *Discovery based on trust and security requirements* in Figure 6 and Figure 12. | WP3 (task 3.1, 3.2 and 3.4)        |
| Security verification module       | This module verifies the service compliance to security contracts both at design time and at runtime. It uses inputs from the trustworthiness prediction module as well as the security property determination module.  
**Related work process:** See *Analyze secure composition of service* in Figure 7 and Figure 11. | Mainly developed in WP2 (task 2.3, 2.4, 2.5), but also contain work from WP3 (task 3.2). |
<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
<th>Related work process</th>
<th>Task(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security composition planner module</td>
<td>This module creates one or several suggestions on how a composition can be assembled based on the service specification and available service components in the Marketplace. It makes use of the Trustworthiness prediction module and Security property determination module to accomplish this.</td>
<td><strong>Related work process:</strong> See <em>Suggest secure composition</em> in Figure 6 and Figure 12.</td>
<td>WP3 (task 3.2)</td>
</tr>
<tr>
<td>Security policy monitoring module</td>
<td>This module enables the monitoring of security policies and properties, as well as the monitoring for security contract fulfilment. It can find the latest descriptors of the security properties in the Marketplace. If the security properties are no longer compliant with the contract requirements, notifications are sent to the affected parties.</td>
<td><strong>Related work process:</strong> See <em>Analyse secure composition of service</em> in Figure 7 and Figure 11.</td>
<td>WP3 (task 3.2 and task 3.4)</td>
</tr>
<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 are uploaded to the threat repository module to be used by interested services. At design time, services created with the Aniketos platform will consult the recommendations to include the threat response capability in order to make services threat aware.</td>
<td><strong>Related work processes:</strong> See <em>Model and analyse trust and security requirements for the service</em> in Figure 5, as well as <em>suggest secure composition</em> in Figure 6 and Figure 12.</td>
<td>WP4 (task 4.2)</td>
</tr>
<tr>
<td>Service threat monitoring module</td>
<td>This module stores the relationship between services and relevant threats in order to notify corresponding components when there is a change of threat level. It can be connected to a security monitoring module in the environment in order to e.g. detect an ongoing intrusion attempt. It also updates the threat information in the threat repository at runtime.</td>
<td><strong>Related work process:</strong> Alerts are received during <em>provide service</em>, see Figure 10.</td>
<td>WP4 (task 4.3)</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, violation of contracts or threats.</td>
<td><strong>Related work process:</strong> Alerts are received during <em>provide service</em>, see Figure 10.</td>
<td>WP4 (task 4.4 and 4.5)</td>
</tr>
<tr>
<td>Community support module</td>
<td>This module is a content repository giving support to service developers, service composers and end users with material such as patterns and guidelines for establishing trust among end users.</td>
<td><strong>Related work process:</strong> See <em>Add reference material</em> in Figure 16.</td>
<td>WP5 (task 5.3) and WP2 (task 2.2)</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, subscribe to</td>
<td></td>
<td>WP4</td>
</tr>
</tbody>
</table>
relevant threat types and incorporate corresponding threat response capabilities according to the threat response recommendations. Runtime response mechanisms might also be stored here if such patterns exists.

**Related work process:** See *add threat information* in Figure 16.

| **Marketplace** | This module includes a set of services supporting Aniketos marketplace. The marketplace complements existing service registry technology, such as UDDI, with specific information on trust and security properties. It acts as a service broker for service consumer giving specific requirements on trustworthiness and security properties. Service providers must be able to upload their offered specifications as service descriptors so that their services are made available for discovery.

**Related work processes:** See *Discovery based on trust and security requirements* in Figure 6 and Figure 12. Also, see announce service in Aniketos marketplace in Figure 8 and Figure 16. | WP5 |

| **Training material module** | 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.

**Related work process:** See *add reference material* in Figure 16. | WP8 |
### Table 9. Descriptions of environment components

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service composition framework</td>
<td>Framework 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 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 compliments 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>Security 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 mechanism</td>
<td>A specific mechanism that we will depend on in order to allocate trust to identifiable entities. Will run in the environment, but accessed by the Aniketos platform.</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>
4.3 Interface specification model

The interface specification models show which interfaces are exposed for each of the platform components in section 4.2. The information that goes in and out of the interfaces is from the system information model in section 4.1.

Figure 22 gives an overview of the interface specification based on the components from section 4.2. Arrows between Aniketos components represent information flow and point from a message sender towards a message receiver. The squares are ports for information flow and the provided ports are shown with the provided interfaces. In addition, ports exposed to the environment are shown together with their provided (circles) or required (half circles) interfaces. These ports are delegated to the corresponding ports of the Aniketos components with arrows.

![Figure 22: Overview of Aniketos platform components with interfaces between platform components and interfaces towards the environment.](image)

**Note:** There may be a link between the community support module and the trustworthiness prediction module if the trustworthiness information can be provided at runtime by Aniketos authority or a third party. This depends on the trust model to be developed in WP2.

Below we show the interface diagrams for each platform component. Each figure shows the component with its provided (shown as circles) and required (shown as half circles) interfaces, as well as the operations for the provided interfaces.
The Socio-technical security modelling tool offers IModelling interface. This interface has two methods:

- `exportSecuritySpecificationModel()` exports a security model from the modelling tool to be used by other modules (e.g., the Model transformation module);
- `modelTrustAndSecurityRequirements()` creates the security specification model.

This module uses two interfaces provided by other modules. It uses IThreatRepository interface to look up the threat repository for relevant threats and their countermeasures (cf. Figure 38). It can for instance use ITransformation to transform security models to Aniketos compliant specifications, and generate contract templates and security descriptors. This can also be done directly by the service composition framework.

The Model transformation module offers ITransformation interface. This interface offers the following methods:

- `transformModel()` gets the security specification model (e.g., using the IModelling interface provided by the security modelling tool) as input and transform it to Aniketos compliant specification.
- `generateContractTemplate()` generates contract template from the Aniketos compliant specification.
- `generateSecurityDescriptor()` generates security descriptor from the Aniketos compliant specification.
- `bindServiceThreat()` registers a service with the Notification module for alerts.

In this design, we assume that Aniketos compliant specification is the primary information item used to pass information about the security model between Aniketos modules. It is however possible for other modules to use the security model directly via the `exportSecuritySpecificationModel()` method of the IModelling interface. For example, the security modelling tool can directly invoke the model transformation module to transform security models to other formats, while other modules can obtain the security model using IModelling interface or use Aniketos compliant specification obtained using the ITransformation interface.
The contract negotiation module provides IContractNegotiation interface for contract negotiation and update and uses the ITrustworthinessPrediction and the ISecurityProperty interfaces during contract negotiation. In addition to the operations provided by the IContractNegotiation interface, the contract negotiation module is also responsible for determining the monitoring points and alerts according to the security contract created.
The security policy monitoring module does not provide interfaces of its own, but uses interfaces provided by other components to perform monitoring of contracts and security policies. For example, it uses IAlert interface to send notifications about contract violations and security property changes.
The notification module offers IAlert interface for other components to register for alerts and to send alerts, such as changes of trust level, threat level and security properties. It requires also the INotification interface so that it can forward the alerts to relevant entities, such as the end user or the service runtime environment.

**Figure 31: Interfaces for the Notification module**

**Figure 32: Interfaces for the Threat repository module**

**Figure 33: Interfaces for the Aniketos marketplace**
**Figure 34:** Interfaces for the Threat response recommendation module

**Figure 35:** Interfaces for the Secure composition planner module

**Figure 36:** Interfaces for the Training material module

**Figure 37:** Interfaces for the Community support module
4.4 System collaboration model

The system collaboration model shows the main interaction between components using messages and information items defined in section 4.1 and interfaces defined in section 4.3. We show only the most basic and high level interactions and there may be other possible interaction sequences for a process.

Figure 38: Aniketos design time service specification process (cf. Figure 5)

The socio-technical security modelling tool allows the modelling of trust and security requirements and represents them in the security specification model. The modelling tool can look up the threat repository for relevant threats and their countermeasures, and incorporate such countermeasures in the security specification model. Such security specification model can be transformed to Aniketos compliant specification using the model transformation module.

Figure 39: Aniketos design time service discovery and selection process (cf. Figure 6)

The Aniketos marketplace matches services against security requirements specified in the DiscoveryRequest based on predicted service security properties and trustworthiness obtained from Aniketos platform services.
The Aniketos platform can support service selection by suggesting secure composition using the functionality provided by the secure composition planner module.

![Diagram of Aniketos service validation and contract establishment](image)

**Figure 40**: Aniketos design time service validation and contract establishment (cf. Figure 7)

Aniketos can validate if a populated composition plan is secure with regard to the security and trust requirements. The verification result can be OK or not_OK. If the verification result is OK and if the services have negotiable terms, a contract negotiation process is initiated.

- If the negotiation is successful, a security contract is created. Aniketos platform then determines the monitoring points and alert thresholds according to the contracted terms for security and trust requirements, and sets up the corresponding monitoring mechanisms.
- If the security contract creation is failed, Aniketos platform notifies the composition framework about the negotiation failure.
- In case the verification is not OK, or the negotiation is failed, Aniketos platform may suggest that the composition framework recompose and then validate again or it may suggest secure new composition directly for the composition framework.
Service is assembled using the service discovery and selection process as shown in Figure 39. After a service is deployed, it is registered with Aniketos notification module for alerts, i.e., the Aniketos platform will notify the deployed service about relevant threats and other changes.

Security descriptor used for publishing a service with Aniketos marketplace can be generated right after deployment (as shown in this figure) or as part of the announcing service process (cf. Figure 42).

After a service is provided (i.e. deployed), the runtime environment starts a loop of monitoring, adaptation, validation, reconfiguration and recomposition as shown in Figure 9. The Aniketos platform will notify the runtime environment about changes related to trust and security properties as well as threat situation.

A service will be announced in Aniketos marketplace after the service is deployed. When its security properties are changed, the service needs to be re-announced in the marketplace with the updated properties (e.g. in Figure 45).
The starting point of the runtime service validation process is the populated composition plan which is obtained using a process similar to the one specified in Figure 39 or in Figure 44. The security verification module analyses the composition and checks if the trustworthiness and security properties of the composition conform to the security contract, which is defined as part of the composition contract. If the verification result is not OK, relevant decisions for adaptation can be made, e.g., a service recomposition or reconfiguration may be needed.

During the runtime recomposition, Aniketos platform supports the service runtime environment for discovery of service component candidates based on trust and security requirements. If desired,
Aniketos platform can also suggest secure composition based on the service specification, the trust and security requirements and the threat information. Aniketos helps to establish the security contract for the composition. If the security contract is created successfully, monitoring points and alert thresholds are determined according to the contract to allow for runtime monitoring and alert. If the security contract cannot be created, the runtime environment needs to decide on appropriate actions, possibly with the help of Aniketos (e.g., suggesting new secure composition, not shown in the figure). Finally, the service is assembled according to the populated composition plan and the composition contract and deployed at runtime.

**Figure 45: Aniketos runtime reconfiguration (cf. Figure 13)**

When the service runtime environment adapts a composite service by reconfiguring a composition, Aniketos platform can support it with the analysis of the reconfigured composition and check if the reconfiguration conforms to the already established security contract. If the verification result is OK and if the reconfiguration requires negotiation of new contract (e.g., some new terms), a contract negotiation process is performed. Such negotiation is a process depending on the negotiation protocol. It can be several rounds until either a contract is established or failed (cf. Figure 44).

The reconfiguration may only lead to update of some terms of the original security contract. In this case, the contract negotiation module will be invoked to update the contract and set up the modified monitoring points and alerts.

Finally, the adapted service needs to re-announce itself on the Aniketos marketplace if its security properties are changed after reconfiguration.
The main monitoring modules in Aniketos platform are the service threat monitoring module, the security policy monitoring module and the monitor trustworthiness module. Figure 46 gives one possible design showing high level interactions, where the Notification module receives all the notifications and decides how to handle them and which components to forward the notifications. Assume that the monitors have knowledge about what to monitor, e.g., each selected service has its own associated monitor (or a set of rules of a monitor).

- The service threat monitoring module receives threat related events from the environment and other Aniketos components, e.g., the trust level change from the Monitor trustworthiness module, collects the log and diagnoses the threat situation. It notifies changes of threat level.
- The security policy monitoring module monitors security properties by checking the security properties determined with the security properties determination module against the security

---

**Figure 46: Aniketos runtime monitoring (cf. Figure 14)**
policies and security properties registered in the marketplace and notifies changes of security properties if any. It also checks the security properties against the security contract and notifies contract violation. A contract violation affects the trust level of a service; therefore, it will be forwarded to the trustworthiness prediction module.

- The monitor trustworthiness module monitors the trust level and checks it against the security contract. It notifies changes of trust level and contract violation.
- When the notification module receives a change notification, it can invoke the security verification module to verify if the composition is secure under such change. If the verification result is not OK, Aniketos can provide different recommendations. For example, the Aniketos platform may simply ask the runtime environment to recompose or reconfigure and then analyze if the recomposition is secure, or Aniketos may suggest secure recomposition for the runtime environment.
- The notification mechanism can be based on publish/subscribe paradigm. In such case, the notification module can forward notifications to relevant subscribers according to the subscription criteria such as the alert/notification types and thresholds (cf. the determineMonAndAlert() method provided by the contract negotiation module in Figure 24). The subscribers can be relevant services or end-users in the environment, or other Aniketos platform components. For example, change notifications can be sent to the service threat monitoring module for threat analysis.

There can be alternative designs for monitoring. For example, each monitoring component can invoke the security verification module to analyze secure composition when there is a change in trust level, threat level or security properties and then send notifications to the notification module depending on the verification result.

![Figure 47: Service end user processes (cf. Figure 15)](image-url)
For service end user processes we assume that an end user service contract is created so that the user security policies and requirements are available in the Aniketos platform and can be used for monitoring. Therefore, when a service end user invokes a service from the service client platform, the service is registered with Aniketos platform for monitoring (cf. Figure 47) and the user will receive alerts when the service security properties or trust level change.

**Figure 48:** Processes for enriching the Aniketos platform at runtime (cf. Figure 16)
5 Conclusion and further work

The three initial viewpoints of the Arcade architecture description framework have been very useful to define and communicate the initial blueprint of the Aniketos platform, and we have intentionally left out specific technology choices from this document in this phase of the project. The architecture will continue to evolve throughout the project and we need to make sure that it is consistent with design and implementation work in the other technical work packages.

The final version of the architecture (D1.5) will also include the distribution and realisation viewpoint, and even though this is not due before M33, we will already at this point start making initial drafts of that. The architecture is a *live* document that will evolve and be aligned with ongoing work in the project.

By having an early draft of the architecture ready in the initial project months, and by involving all partners in the scenario building and requirements elicitation this document has truly become project-wide. It has and will continue to be a key tool for technical management of the project. There are minor inconsistencies between D1.2 and some of the other deliverables, but we care about constantly trying to align these deviations.
References

2. OASIS, Reference Model for Service Oriented Architecture 1.0. 2009.
Appendix A. Aniketos glossary

This is the glossary of terms relevant to the ANIKETOS project. The terms and their definitions appear in alphabetic order. For better understanding, some of the terms contain short examples. The references are to the Glossary references on page 91.

Access control

A general term for regulating how resources in a system may be used. Its most common use in a software system is to implement Authorisation [1].

Actor

An actor is an active entity that is external to the system and interacts with the system. An actor can be a human being or another system [2].

Adaptability

Adaptability is the ability to detect and react to changes in the state of a system including its context, such as resource availability and service requirements [3].

Alert

In Aniketos, an alert is a kind of notification that could result in a service adaptation/recomposition. An alert can be sent after a security incident has been detected (reactively) or in order to make a service protect itself of something that potentially might happen (proactively).

Aniketos Platform Environment

The environment in Aniketos platform is a set of systems that interact and work together with it. The environment systems include the external design time tools and runtime components that the Aniketos platform need to interface with, and thus influence the operation of the platform.

Authenticity

Authenticity is the property of being genuine and able to be verified and be trusted [4].

Authorisation

An approval that is granted to a system entity to access a system resource [4].

Authorization can also refer to the process for granting approval to a system entity to access a system resource [4].

Availability

Availability is the property of a system or a system resource being accessible, or usable or operational upon demand, by an authorized system entity, according to performance specifications for the system; i.e., a system is available if it provides services according to the system design whenever users request them [4].

Change

A change in Aniketos is a wide term, e.g. referring to an update of the security properties of a system, when the environment becomes more hostile or new threats suddenly becomes more probable, or a drop in the trustworthiness in one of the services being part of a composition. A
change could either strengthen or weaken the trustworthiness of a service, and would typically send out an alert to inform the affected entities.

**Commitment**
Social commitments are a conceptual abstraction that represents social relationships between different agents. A commitment is a promise (a contract stipulated in a socio-legal context) between a debtor and a creditor to bring about a certain state of the world. Commitments can be conditional: the state of the world will be brought about only if a certain precondition holds. More information is provided in D1.1, section 2.3.10.

**Composite service**
A service visible to a service consumer (or agent) via a single interface and described via a single service description that is the aggregation or composition of one or more other services. These other services can be atomic services, other composite services, or a combination of both [1]. A synonymous term is Federated service.

Composite services consisting of services offered by several providers are the main focus for Aniketos. We anticipate a shift towards this kind of services for the Future Internet and we are working with challenges associated to composite services.

**Confidentiality**
Confidentiality is the property of a system that certain information can only be read by authorised users [2].

Confidentiality concerns the protection of privacy of participants in their interactions. Confidentiality refers to the assurance that unauthorized entities are not able to read messages or parts of messages that are transmitted [1].

**Context**
Context is the situation within which something exists or happens, and that can help explain it [5].

When referred to services, the service context is the set of circumstances that occur when the service is being executed at run-time. Context changes might impact in a need for service adaptation or recomposition.

Information about a particular service activity (in addition to session data) is qualified with the term context. The larger or more complex a service composition, the more context information will generally need to be managed because more services and inter-service data exchanges will be involved in the corresponding service activity [6].

See also Context change.
See also Context sensor.
See also Context viewpoint.

**Context change**
A service context change is any modification in the situation in which the service works.

In Aniketos, we consider that context changes are detected by Context sensor in the Aniketos platform environment. E.g. a context change occurs when a service consumer crosses the border between two countries while using a telco service in his/her mobile phone.
Context viewpoint

Context viewpoint of the Aniketos architecture expresses what the Aniketos platform is intended to do in its environment and how the interaction is. The context viewpoint is included in deliverable D1.2.

Context sensor

Context sensor is a functionality or service in the Aniketos platform environment that at runtime can be used to detect context changes that can trigger Aniketos activities. E.g. change of user location.

Countermeasure

An action, device, procedure, or technique that meets or opposes (i.e., counters) a threat, a vulnerability, or an attack by eliminating or preventing it, by minimizing the harm it can cause, or by discovering and reporting it so that corrective action can be taken [4].

In Aniketos we want the security properties to be realisations of countermeasures and thus mitigate vulnerabilities. A countermeasure can for instance be a template or security pattern for service composition.

Dependability

Dependability is the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers [7].

Dependability is an integrating concept that encompasses the following attributes: availability: readiness for correct service; reliability: continuity of correct service; safety: absence of catastrophic consequences on the user(s) and the environment; integrity: absence of improper system state alterations; maintainability; ability to undergo repairs and modifications. When addressing security, an additional attribute has great prominence, confidentiality, i.e., the absence of unauthorized disclosure of information. [8].

Design time support

In Aniketos, 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 will also be able to choose service providers and service components by trustworthiness aspects for service composites.

Encryption

Transforming a piece of data to make it unreadable for anyone except those who know a certain piece of information ("key") to revert the encrypted information to its original form [2].

Environment

The environment is a wide term referring to the surroundings of the service. This may include people, devices, software and conditions that influence or interact with the service. This term should not be confused by the definition of Aniketos Platform Environment.

Federated identity

Federated identity approach allows sharing unique identities securely across different networks and applications. A federated identity is a single user identity that can be used to access resources across different web sites, networks or applications that are bound by the ties
of federation. With federated identity, users do not need to manage different credentials in each of the federated systems, but they can work with a single identity.

**Identity**
The collective aspect of a set of attribute values (i.e., a set of characteristics) by which a system user or other system entity is recognizable or known [4].
The set of physical and behavioral characteristics by which an individual is uniquely recognizable. [9]

**Identity management system**
Identity management system is a special-purpose system, possibly comprised of one or more systems or applications, that manages the identity verification, validation and issuance process [9].

**Integrity**
Integrity is the property of a system that certain information can only be modified by authorised users [2].
Integrity concerns the protection of information that is exchanged – either from unauthorized writing or inadvertent corruption. Integrity refers to the assurance that information that has been exchanged has not been altered [1].

**Metric**
A metric provides a measurable quantity for a given property. In relation to trustworthiness this can be a numerical value associated to e.g. a reputation level, the strength of an evidence, or the economical value of a promise/guarantee.

**Monitoring**
Monitoring in Aniketos is the process of checking that security contracts are fulfilled over time, particularly if there has been a change. This process involves verification. Also, monitoring is used to detect vulnerabilities and to discover attacks on a service, e.g. by making use of intrusion detection systems or dynamic testing tools found in the environment. Monitoring may register a change.

**Privacy**
Privacy is the right of an entity (normally a person), acting in its own behalf, to determine the degree to which it will interact with its environment, including the degree to which the entity is willing to share its personal information with others [4].
Privacy is the confidentiality of personal data. Personal data is any data that describes the properties or the behavior of a specific human being [2].

**Reliability**
Reliability is the ability of a system to perform a required function under stated conditions for a specified period of time [4].
Trust in infrastructure facilities could also be known as reliability [1].

**REST**
REST (Representational State Transfer) is an architectural style for distributed hypermedia systems. It is a formal definition of core design principles of the World Wide Web and the
HTTP/1.1 protocol and advocates a design approach for constructing distributed systems based on the notion of resources [6].

REST services
REST services are lightweight programs that are designed with an emphasis on simplicity, scalability, and usability. REST services can be further shaped by the application of service-orientation principles and therefore represent an alternative implementation medium to SOAP-based Web services and components for building services in support of SOA and service-oriented computing [6].

Runtime support
In Aniketos, runtime support consists of methodologies and tools that monitor and evaluate the trustworthiness and security violations of service components during service execution, using design time definitions and also considering contextual information such as change in operation conditions and users’ behavior. 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.

Security contract
A security contract is a part of or addition to a service contract.

Security contracts are vital in Aniketos as a mean to identify what is required by a service consumer and what is offered by a service provider with respect to security properties and trustworthiness. These contract requirements can be used for discovery and filtering of relevant services before the assembly. The terms of an operational security contract must be monitored in order to ensure that they are kept over time.

See also service customer security policy, service provider security contract, and service security agreement.

Security engineer
A security engineer is the person with assigned responsibility for conducting security engineering activities.

See security engineering.

Security Engineering
Security Engineering is an interdisciplinary approach and means to enable the realization of secure systems. It focuses on defining customer needs, security protection requirements, and required functionality early in the systems development life cycle, documenting requirements, and then proceeding with design, synthesis, and system validation while considering the complete problem [10].

Security need
Security need is a security property the service consumer wishes to be satisfied by the service provider.

Security objective (more standard way)/ security goal
A security objective is a statement of intent to counter identified threats and/or satisfy identified organization security policies and assumptions [11].
Security property
A security property is a characterization, attribute or specification of a service that realise a security requirement or policy.
E.g. service A offers encrypted communication using a 1024-bit RSA key, only stores data locally for 24 hours or it only allows one instance of an identity to be logged in concurrently.

Security policy
A security policy is a set of policy rules (or principles) that direct how a system (or an organization) provides security services to protect sensitive and critical system resources [4]. Policies prescribe the conditions and constraints for interacting with a service [1].
In Aniketos, the relevant security policies need to be represented as a part of service specifications. This implies developing a way of defining and representing security policies that would fit the purpose of this project.

Security requirement
A security requirement is a detailed requirement that implements an overriding security policy [12], and should express what is to happen in a given situation, as opposed to what is not ever to happen in any situation [13].
In Aniketos, security requirements should be expressed as a part of the service specification. This implies developing a way of defining and representing security requirements that would fit the purpose of this project.

Security risk
Security risk is an expectation of loss expressed as the probability that a particular threat will exploit a particular vulnerability with a particular harmful result [4].

Service
A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description [1].
The Aniketos platform is limited to software services, but its use and application must be seen in relation to the environment in which the services operate in, and interaction with non-software services (e.g. functionally performed by humans).
Software services are often referred to as service components. A service component provides and requires a finite set of services. These services are provided through public interfaces [14].

Service adaptation
Adaptation is needed when there is a mismatch between the quality of the services provided by the application and the user needs, or between the application needs and the context. Adaptation may lead to a degradation or enhancement of the quality of the provided services (QoS). For instance, the functional richness of an application may be modified, or the response time of a service may be reduced [3].
We consider adaptation to be a concept related to self-management and dynamic behaviour, in contrast to evolution where humans are more directly involved in the loop and there are new releases at regular intervals. Examples of adaptation are reconfiguration of how the services are used during when a user moves into a hostile environment, leading to a change in the overall functionality of the composite service, or recomposition.
Service architect
The service architect is a person able to analyse and design services so that they can be offered by a service provider. 

*It is a role concentrated on the design of services. Service architect therefore can participate in the definition of service candidates and then assume responsibility for authoring design specifications that fulfill business requirements while being compliant to design standards and service-orientation principles* [6].

Service assembly
Assembly is the process of creating a runnable composite service instance based on an abstract service specification. As a part of this process, binding is used to associate protocol or data format information with an abstract entity like a message, operation, or portType [15].

In order to create an assembly you need to have performed discovery of service components in order to populate the service template and get a service composition plan.

Service composition
See service assembly.

Service consumer
A service consumer is a participant that interacts with a service in order to realize the real world effect produced by a capability to address a consumer need [1]. A service consumer may mediate the service to another consumer; thereby have the role as a provider as well.

Service contract
A service contract is a composition of functional metadata and a set of policies, such as security constraints, access restrictions for user groups, transport and service level agreements, charging, etc [16] [6]. This is an agreement between entities offering (service provider, see below) and requiring a service (service consumer, see below), regarding the purpose and consequences of the interaction [17]. Service level agreements (SLAs) are one commonly used category of contracts [1].

Service consumer security policy
Service consumer security policy (aka. security policy) is a set of security properties the service consumer wishes to be satisfied by the service provider.

Service description
A service description is an artifact, usually document-based, that defines or references the information needed to use, deploy, manage and otherwise control a service. This includes not only the information and behavior models associated with a service to define the service interface but also includes information needed to decide whether the service is appropriate for the current needs of the service consumer. Thus, the service description will also include information such as service reachability, service functionality, and the policies and contracts associated with a service [1].

In Aniketos a service descriptor works like an agreement template [18], offering one or many options that will be manifested in the agreed service contract.
Service developer
A service developer is a person able to create, modify and/or assemble services so that they can be offered by a service provider. Related terms are service composer, service designer and service creator.

Service discovery
The act of locating a machine-processable description of a network-related resource that may have been previously unknown and that meets certain functional criteria. It involves matching a set of functional and other criteria with a set of resource descriptions. The goal is to find an appropriate service-related resource [19][9].

In Aniketos, discovery is mainly focused on discovery based on security and trustworthiness.

Service engineer
See service developer.

Service interface
A service interface is the abstract boundary that a service exposes. It defines the types of messages and the message exchange patterns that are involved in interacting with the service, together with any conditions implied by those messages [20].

Service orchestration
A service orchestration is the description of how a specific service can be realised by interacting with other services. The orchestration is under control of a single endpoint. An Orchestration may be executable [7].

An orchestration defines the sequence and conditions in which one Web service invokes other Web services in order to realize some useful function. I.e., an orchestration is the pattern of interactions that a Web service agent must follow in order to achieve its goal [20].

Service provider
A service provider is a participant that offers a service that enables some capability to be used by other participants [1]. In Aniketos, a service provider is responsible for and offers services to service consumers at runtime. The term service host is often used about the same concept, though a service host is usually only responsible for the service infrastructure (network, processing power, memory) and not the service itself.

Service provider security contract
Service provider security policy (aka. security contract) is a set of security properties the provider wishes to offer to the service consumer.

Service recomposition
A recomposition is a specialisation of adaptation where the services in a composite service are rearranged and/or replaced. After a recomposition the composite service should still fulfil the service specification. A recomposition can only occur for composite services.

Service security agreement
Service security agreement (aka. security agreement) is a mutually agreed set of security properties the customer wishes to be satisfied by the service provider and the provider wishes to offer by their engagement.
Service specification
A service specification is an abstract representation of a service, typically defining the goals, functionality, behaviour, requirements and policies for the service. A service specification is usually the result of a service design. Languages commonly used for abstract specifications are given in [14], ranging from OWL-S, WS-BPEL, formal languages [21], 1st order logic and graph modelling languages.

Aniketos contributes to service specifications by enabling information about socio-technical requirements to be described. Examples are trust relationships and required security properties for service compositions.

SVRS
SVRS is an abbreviation for the SHIELDS Security Vulnerability Repository Service, which is a repository used for sharing of security knowledge (e.g. security models and artefacts) [2].This repository is a result of the SHIELDS FP7 EU project. For more information see D1.1.

Template
Templates describe the outline of activities needed to solve a problem, and are parameterized with respect to some variables so the generic template can be configured and customized for a specific instance based on the users’ current requirements and preferences [22]. Templates have the advantage of being reused and extendible [14]. A template can be a part of a service specification, and used as a blueprint for the service in order to discover services and assemble them.

Threat
A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm [4]. A threat is realised by exploiting vulnerabilities in a service.

In Aniketos we are specifically considering threats that could harm compositions and ways to mitigate these. The presence of a threat to a system could trigger an alert. The expected loss associated to a threat is called risk [4].

Trust
Trust is a concept that lies at the intersection of several domains, including sociology, psychology, law, economics, ethics, and computer science [23], hence it requires a socio-technical approach. Following the definitions from [24], trust is a quantifiable relation between two agents (humans or software systems), and we may differ between functional trust, which is the belief in an entity’s ability (and willingness) to carry out or support a specific function on which the relying party depends, and referral trust, which is the belief in an entity’s ability to recommend another entity with respect to functional trust. Another distinction is direct trust (where the trustor trusts the trustee directly without intermediaries) and indirect trust (based on opinions from third parties).

Cognitive trust
In Aniketos we consider cognitive trust between (living, breathing) cognitive agents, following the research line of Castelfranchi and Falcone, which promote that a trust relationship can be considered as a five-part relation [25],

\[ \text{trust}(X, Y, c, \tau, g) \]
which denotes that $X$ (the trustor) trusts $Y$ (trustee) in context $c$ to do task $r$ (action $a$, leading to state of affairs, $p$) to achieve $X$’s desired goal, $g_x \subseteq p$.

Castelfranchi and Falcone [25] argue that the trustor ($X$) must be a cognitive agent with goals and intentions. Or to put it another way, to trust, according to the authors, it is necessary (though not sufficient) to be able to care about achieving a goal. A computer cannot trust, according to the authors, for any sensible notion of trust. However the trustee ($Y$) need not be a cognitive agent, so, matching intuitions, it is possible according to the theory to trust a software component.

See D2.1 for more information on social and cognitive trust.

**Non-cognitive trust**

Non-cognitive trust in Aniketos is associated to objective and measurable properties of the service (see trust metrics).

**Trust level**

See trustworthiness level

**Trustworthiness**

*Security decision with respect to extended investigations to determine and confirm qualifications, and suitability to perform specific tasks and responsibilities* [11].

In Aniketos, trustworthiness is something that can be computed, measured or cognitively estimated in order to evaluate to what degree a service should be trusted. Trustworthiness is associated to metrics, measuring one or a set of properties.

**Trustworthiness level**

The trustworthiness level of a service is a ratio including various trust metrics and scalable results of trust mechanisms.

The trustworthiness level of a service in Aniketos, is a combination of both:

- Cognitive trust of the user (service consumer or service developer) on the service that is based on reputation derived from:
  - previous experiences of the user (direct trust),
  - recommendations given by a user, or by a community of users, on the service (indirect trust) and
  - other particular and unpredictable subjective metrics/aspects that are of important for the user for evaluation of trust.

- Non-cognitive trust is associated to objective and measurable properties of the service (see trust metrics).

**Trust metrics**

See trustworthiness metrics.

**Trustworthiness metrics**

Metrics associated to the trustworthiness of the service, i.e. to the ability of the service to be trusted. Depending on the type/nature of the service some metrics might apply and some others not. See metric.

For a detailed list of trustworthiness metrics that is being considered in Aniketos see D2.1.
Trust relationship
In Aniketos, we consider trust relationships in two different contexts, service composition and service provisioning, described below.

In Aniketos, a trust relationship needs to be established between a service developer (i.e. the trustor) and each of the component services (i.e. the trustee) when creating a composite service (creation of a composition plan at design time) The object of the trust (i.e. the trustum) is formalised in a service provider security contract which includes the set of security properties the provider wishes to offer to the service consumer of the component at stake. The strength of this relationship is inspired by the predicted service property trustworthiness.

As Aniketos also supports service provisioning, a trust relationship needs to be established also in this context between a service consumer (i.e. the trustor) and a service provider (i.e. the trustee). During service provision the consumer uses (consumes) the service so its composition plan is deployed. As opposed to service composition, the object of the trust (i.e. the trustum) is formalised in the service security agreement. In this case, the strength of the trust relationship is inspired by the monitored service property trustworthiness.

Validation
Validation is a process intended to establish the soundness or correctness of a construct [4]. In the case of Aniketos, validation is the process of ensuring that the instantiation of the composite service has the right goals and requirements according to the service specification. Validation may be used pre-deployment by a service developer or during runtime adaptation/recomposition.

Verification
Verification is derived from the Latin word for “true” [4], and is the process of checking that the goals and requirements are fulfilled.

Vulnerability
A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy [4].

In Aniketos, vulnerabilities that arise due to compositions are of particular interest, such as elevation of privileges or information leakage due to poor compositions. If vulnerability is detected within a service, this would typically reduce the trustworthiness of the service until it has been repaired or mitigated.

Web service
A Web service is a body of solution logic that provides a physically decoupled technical contract consisting of a WSDL definition, an XML schema definition, and possibly a WS-Policy definition. This service contract exposes public functions (called operations) and is therefore comparable to a traditional application programming interface (API) [6].
Glossary references


Appendix B. Aniketos scenarios

Figure 49 shows an overview of the scenario handling process.

The following scenario descriptions are the results of the scenario handling process, and have been used as a basis for requirements elicitation, the architecture and as starting points for user stories in several other deliverables. There are two types of scenarios, those denoted “A” which explain the normal “happy-day” flow of actions, and those denoted “B” or “C” that show exceptions and threats.
Scenario 1A: Service selection at design-time

Description
Bert is a service developer at SoftServ and is creating a service partly consisting of own code and services provided by others. For his service he needs a mechanism to handle online customer payments in an easy and secure manner, so he uses a UDDI registry to find services capable of this feature. The registry provides him with a long list of alternatives, so Bert is a bit unsure which one to choose. Three of these services have information about their trust level through the Aniketos Marketplace, so Bert is able to match these with the required trust level of the service he is putting together. Bert picks one service based on the closest match and gets an assurance that his service will be notified if the trust level of the payment service should drop below a certain threshold.

Workflow
1. Search for service components according to specific needs
2. Select among suitable service components based on trust level
3. Integrate service component with closest match
4. Register for notification if the external service component should change trust level at runtime

Alternate workflow
2b. None of the service components have the required trust level, adjust the trust requirements of the composite service.

Stakeholders
Service developer and composer, Service provider, Aniketos marketplace

Outcome
Bert is able to put together an overall trusted service for the end user.

Scenario 1B: Manipulation of trust properties

Description
Johnny is a service provider with limited ethics. By setting up a large number of false composite services using a payment service he provides he is able to boost the trust level of this payment service. The trust levels are determined by a system based on recommendations from consuming services.

Stakeholders
Unethical service provider

Outcome
Trust levels can be manipulated

Assets
-

Countermeasures
TBD
### Scenario 2A: Keeping the service end user informed

**Description**
Alice has an application on her mobile phone that automatically uses the cheapest way of calling based on her location and available networks. However, Alice does not want to use a service that stores her whereabouts and too much information about her. Alice is making a VoIP call whilst sitting in a train with free WiFi access going from Switzerland to France, and service components used by her app exchange information about her identity, current location and available gateways to the phone network. When Alice crosses the border to France, the service needs to perform a recomposition in order to change to a national gateway for France. However, this new VoIP-to-phone network service requires more identity data compared to the one used in Switzerland. Alice gets a request on her mobile screen to give out more identity data, but Alice denies and her call is switched to a more costly 3G voice call.

**Workflow**
1. A service end user uses a composite service that requires a given set of identity data
2. The service needs to do a recomposition due to her context change.
3. A service used in the new composition requires additional identity data that Alice prefers not to publish.
4. Alice is notified about this and chooses to force a new recomposition so that a 3G network is used for her call and she does not have to publish the additional identity data.

**Alternate workflow**
4b. Alice accepts to give away the additional identity data

**Stakeholders**
Service end user, VoIP service providers

**Outcome**
Alice can make informed decision on her privacy requirements

### Scenario 2B – End user gets annoyed by confirmation requests

**Description**
Alice is using her mobile app in a rural area exploiting free WiFi access points to make a VoIP call. The ISP’s of these access points have somewhat different policies to what kind of identity data they require, so when Alice is on the move she gets a new request every 15 second about giving away identity data. Alice gets annoyed and selects “Always OK”

**Stakeholders**
Service end user, Internet Service Providers

**Outcome**
A security mechanisms meant to protect user identity data is disabled due to the number of requests

**Assets**
Identity data

**Countermeasures**
TBD
### Scenario 3A: Secure behaviour monitor of a composite service

**Description**
John is a security expert working with testing and monitoring of services for a software company. He notices that a composite service has a vulnerability that can be exploited through a well-known attack mechanism. John analyses the composition of that service to find the atomic services being used, and derives which is the service containing the error. He then alerts the provider of the service, who again gets in touch with a service developer to have this fixed. The Aniketos platform makes sure that all compositions depending on this service are notified as well in order to take preventive actions.

**Workflow**
1. End user selects a composite service
2. The service is checked to detect known types of vulnerabilities.
3. The user checks the atomic services to find the one causing the vulnerability
4. The user alerts the service provider in order to improve the implementation
5. Service consumers depending on the service are notified of the vulnerability

**Alternate workflow**
3b. The atomic services are secure by themselves, it is their combined usage that causes a vulnerability, and the composition should be modified.

**Stakeholders**
Service provider, Security expert, service developer

**Outcome**
New vulnerability occurrence is detected

### Scenario 4A: Specification of a Aniketos service

**Description**
Atomicland is a company that creates and offers small supportive services to be used by third parties. Atomicland has recently developed a new service and wants to advertise it through the Aniketos Marketplace. Guybrush, a developer of Atomicland, visits the Aniketos platform website to find requirements to and examples of services that can be announced through the marketplace.
Guybrush then uploads an Aniketos compliant specification of his service to the marketplace. This specification contains information about the security properties and ways to ensure the trustworthiness of the service at any time.
Once this process has been finished, the service can be discovered through the Aniketos marketplace

**Workflow**
1. The service provider connects to the Aniketos web interface and get the technical information
2. The service provider uploads the service specification to the Aniketos marketplace
3. The service specification is published through the Aniketos Platform

**Alternate workflow**

**Stakeholders**
Service developer, Service provider, Aniketos marketplace,

**Outcome**
A company uses Aniketos Platform to publish digital services with all the information related to the service
### Scenario 5A: Trust handover at runtime

**Description**
John wants to acquire a permit for a new building he plans to erect. However, he wants to use a civil engineer to process the permit's procedure for him. He has no such engineer at hand. So he uses a service to select one from a registry of available engineers. First, he provides his credentials to the service and then chooses to hire an engineer, for this example, let's call him Manos. He then authorizes Manos to act on his behalf for the rest of the permit's procedure.

Manos now uses the service provided from the Department of Urban Planning to issue the permit. The department is aware of the trust established between John and Manos and allows Manos to proceed in John's stead.

**Workflow**
1. John wants to acquire a permit
2. John provides his identity and uses a service to select Manos
3. A trust relationship between John and Manos is established for the purpose of the permit's issuance
4. Manos provides his identity to proceed the permit in John's stead, using the services provided

**Alternate workflow**
2b. John goes through the procedure himself, without Manos' involvement

**Stakeholders**
Service end user, service providers

**Outcome**
The permit application service is made aware of the trust relationship between John and Manos.

### Scenario 6A: End user accesses services supplied by different service providers with their own identity-based security policies

**Description**
An end user uses his notebook to access a web application in order to choose a new digital camera.

The end user provides his credentials to the web application, which is actually a composite service. His identity is managed by an Identity Management System service, which allows the user to access a group of services/applications authenticating only once. The application then invokes an online shopping search engine, which helps the end user to find offers in a quick and easy way, a service offering user and expert reviews and finally a service using many specific shopping guides to find the best price.

This part of the composite service allows the user just to find information, but not to buy the camera, so he needs to select and use other service components as well.

The end user selects a service that, based on his GPS location, helps him to find a shop in the neighborhood where he can go and buy the camera.

Based on the gathered information, he chooses a shop, but before reaching it, he wants to make certain that the product is available. So he uses VoIP service to call the shop and to check the availability of the camera he wants to buy.

**Workflow**
1. A service end user provides his identity information and uses a service component to choose a camera
2. The end user uses another service component to find a shop in the neighborhood
3. The end user uses a VoIP service to place a call

**Stakeholders**
Service end user, service providers, network providers

**Outcome**
End user can access several services without the need to authenticate more than once.
<table>
<thead>
<tr>
<th><strong>Scenario 7A: Trustworthy services composition with the minimum disclosure of identity related information</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>A Service Provider wants to offer to its end users a web-shop application, which results from the composition of different kinds of services provided by different Service Providers (SP1, SP2 and SP3). Some composing services need identity related information, while others don’t. The application uses preferences and GPS-based position information associated to the identity information allowing the end users to:</td>
</tr>
<tr>
<td>- find a list of shop-stores in the neighbourhood (Service 1 provided by SP1);</td>
</tr>
<tr>
<td>- choose to pick up the product at the shop-store or to receive it at home (Service 2 provided by SP2).</td>
</tr>
<tr>
<td>- carry out an online transactional payment (Service 3 provided by SP3);</td>
</tr>
<tr>
<td>Whenever an end user wants to access the web-shop he/she has to provide his/her credentials.</td>
</tr>
<tr>
<td>In order to give the list of the closest shop-stores the related service doesn’t need to manage identity related information, it just uses position information.</td>
</tr>
<tr>
<td>The on-line purchase service needs identity related information and additional critical data, as credit card number, that need to be securely transmitted among users, web application and online purchase service.</td>
</tr>
<tr>
<td><strong>Workflow</strong></td>
</tr>
<tr>
<td>1. The end user accesses the web-shop and provides his/her credentials;</td>
</tr>
<tr>
<td>2. The web-shop application provides SP1 with end user’s position information and obtains a list of the closest shop-stores (Service 1);</td>
</tr>
<tr>
<td>3. The web-shop application shows the user a form to choose how to get the product, and provides SP2 with information for booking the product at the shop-store or for sending it to the user’s postal address (Service 2)</td>
</tr>
<tr>
<td>4. The web-shop application asks the end user for his/her credit card number and invokes SP3 in order to carry out the online transactional payment (Service 3).</td>
</tr>
<tr>
<td><strong>Stakeholders</strong></td>
</tr>
<tr>
<td>Service developers, service providers, service end users</td>
</tr>
<tr>
<td><strong>Outcome</strong></td>
</tr>
<tr>
<td>Service developers are enabled to create applications composed by services that manage identity information and services that don’t. The end user is assured that just composing services that need identity information are allowed to manage them.</td>
</tr>
</tbody>
</table>

---

2 With ‘Identity related information’ we refer to ‘Identity attributes’ dealt with in Rec. ITU-T Y.2720 (01/2009)
Scenario 8A: Federated Identity in Next Generation Networks (NGN): telecommunication services

Description

The aim of this use case is to demonstrate Identity Management\(^3\) in NGN frameworks, with particular regard to Identity Federation\(^4\) and Single-Sign-On (SSO).

In this scenario, there are two telecom operators (NGN providers), each one is supposed to play the role of Identity Provider (IdP) in the respective domain.

Besides the two telecom operators (Home, Visited), there are two service providers (VoIP SP and ContactBook SP), each belonging to a different operator’s domain.

Pre-defined collaborative agreements have been signed between each SP and its corresponding IdP, as well as between the two telecom operators (IdPs), so that single-sign-on is possible.

During a trip in a foreign country, the user is connected to the Visited network, and searches for a phone number stored in his/her ContactBook placed in the Home domain. Then, the user calls his/her colleague with a VoIP softphone from the Visited domain.

Workflow

1. The user accesses the ContactBook service placed in his/her Home domain to retrieve the phone number of his/her colleague.
2. Since no authentication has been done so far, the user from the Visited domain is redirected to the Home domain. Then, he/she finally authenticates on the Home domain.
3. By means of the federation established between the two domains, from now on, the user will get access to all services in a seamless manner, with no further authentication(s) required.
4. The user is now authenticated on the ContactBook service and retrieves the phone number of his/her colleague.
5. Then, the user wants to call his/her colleague whose telephone number he/she finally knows by using the softphone from VoIP service provider placed in the Visited domain.
6. The user is already authenticated for the VoIP service and can make the call with his/her softphone.

Stakeholders

Users, Service Providers, NGN providers, Service Developers

Outcome

VoIP is a conversation-oriented service used as a complement to data transfer in corporate services. It has been included as proof-of-concept in order to demonstrate end-user’s access between VoIP services and other identity-based web services (such as contact book) without the need of re-authentication.

Users will have seamless access to multiple applications with single-sign-on experience in a multi-operator environment.

---

\(^3\) In a general Identity Management (IdM) architectural model, an Entity (User) seeks for a service from a Service Provider (SP) - or Relaying Party (RP) - and provides a claimed identity to that party. The RP needs to have these credentials authenticated before providing the service, so it queries the Entity for the name of the Identity Provider (IdP) for the claimed identity. The RP then queries the IdP for validation of identity that may return some attributes of that identity.

\(^4\) In Identity Federation, a “Circle of Trust” (CoT) is established among a set of Service Providers in order to allow the user to get consistent access to composite services without interruption. The user has to sign only once (SSO) in the system, and he/she will be authenticated for each application (in the trusted domain) he/she wants to use in the future without the need to provide further authentication.
### Scenario 8B: Leveraging the NGN operators’ role in IdP

**Description**

In a multiple service provider environment, it is possible for an NGN provider to be an Identity Provider (as in Scenario 1A).

However, it is also possible for an NGN provider to offer IdP services to other providers, and finally, it is possible to use third party IdP services.

**Stakeholders**

NGN operators, IdP providers

**Outcome**

- Leverage of the vast amount of information that NGN operators have about their customers (i.e. the subscribers’ database).
- Leverage of NGN operator’s service enabler functionalities, like presence, location and availability, in order to enforce policies and adopt sophisticated mechanisms in the authentication/authorization process.

### Scenario 9A: Federated Identity in Next Generation Networks (NGN): discovery of IdP services

**Description**

The aim of the use case is to demonstrate Identity Management in NGN frameworks, with particular regard to Identity Federation and Single-Sign-On (SSO).

In this scenario, an user wants to deploy two kinds of applications:

- portal-based applications, in which there is a pre-determined number of SPs that can be seamlessly accessed by users with a single authentication. This case, also known as pre-determined federation, can be easily achieved by an agreement stipulated among SPs which is statically pre-configured in IdP location.

- applications delivered by SPs outside the domain of trust of the user.
  
  In order to be accessed by the user, the authentication process requires that each SP sends to the IdP an explicit authentication request.

  The latter case requires an IdP discovery mechanism: the user should have to input his/her IdP thus having a simplified (instead of single) sign-on experience. In order to avoid this, the system provides mechanism(s) to perform this task automatically.

**Workflow**

1. User Bob wants to make a hotel reservation for organizing his next business trip.
2. Bob authenticates to the portal by issuing his credentials (e.g. username/password). From now on, he is authenticated to all applications belonging to the portal domain, e.g. in this case he needs a Yellow Page service.
3. Bob gets access to the Yellow Page service in order to find a list of hotels for his final destination. User credentials are exchanged between the Yellow Page SP and the IdP in a transparent manner to the user. Then, Bob gets a list of suitable hotels.
4. Bob is now ready to access the hotel website in order to make a reservation. Since this is a reserved area and the hotel SP is outside the domain of trust of the user, the user must be authenticated.
5. An IdP discovery process is started so that the user’s IdP can be located by the SP and then, the authentication takes place automatically.

**Stakeholders**

Service Providers, Users, Identity Providers
<table>
<thead>
<tr>
<th>Outcome</th>
<th>Users have access to all portal-based applications in their domain of trust (CoT) with SSO. Nevertheless, they can have seamless access to web applications outside this domain.</th>
</tr>
</thead>
</table>

### Scenario 10A: Design time verification of security properties

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob is developing a service composed of other services. There are various ways to combine the services, taking into consideration, e.g., various requirements for privacy and provision of adequate QoS. Bob wants to verify these properties using an automatic theorem prover.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Workflow</th>
</tr>
</thead>
</table>
| 12. The architecture is specified using a graphical notation  
13. This is automatically translated into a formal logic description  
14. Which in turn is passed to an automatic theorem prover |

<table>
<thead>
<tr>
<th>Alternate workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholders</td>
</tr>
<tr>
<td>Service developer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob receives a summary of how many properties have been automatically proved; counterexamples; and conjectures which can neither be proved nor disproved using the system.</td>
</tr>
</tbody>
</table>

### Scenario 10B: Human factor failure: unable to understand formal proof

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only 70% of the automatically generated conjectures are proved/disproved; the rest require manual proof. Formal descriptions are passed back to Bob which he doesn’t understand. Bob decides that 70% coverage is sufficient.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stakeholders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service developer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>An (in principle discoverable, from the specification) privacy issue is present in the architecture design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Countermeasures</td>
</tr>
<tr>
<td>- Services and compositions of services provide transparent descriptions of what verification methods have been used</td>
</tr>
<tr>
<td>- Service developers are educated what to do in the event that the theorem prover is unable to prove or disprove some of the conjectures.</td>
</tr>
</tbody>
</table>
### Scenario 11A: End user makes (some) decisions about services used

**Description**
Alice is an expat and wishes to buy a paper (i.e., tree-based) subscription to her favourite newspaper from her home country. The paper uses a SOA such that Alice can control which services are composed to provide the transaction, for instance she can use a credit card validation checker closest to her home country; she can choose from a range of couriers, some of which are more expensive but more flexible in where they deliver to.

**Workflow**
1. Go to newspaper’s website
2. Enter various details
3. Provides options for payment, automatically selecting an option previously used on another site. Similarly for the couriers.

**Alternate workflow**

**Stakeholders**
Service end user

**Outcome**
Alice receives the newspaper subscription.

### Scenario 11B: Information and accountability lost

**Description**
Alice tries to sign up for a subscription to the newspaper from a foreign country, making use of an SOA comprised of a series of services. For delivery reasons it is not possible to send the newspaper to that country, so one of the services cancels the order. However this is a rare event and the service does not pass the information back to other services with which it is composed. In fact one service sends an email to Alice saying that her subscription was successful. The newspaper has no record of Alice’s details.

**Stakeholders**
Service end user, service developer

**Outcome**
Alice waits for her subscription to arrive, which it never does. She calls the newspaper and speaks to Bob who has no record of her details. Bob does not report the error, trusting the system, assuming that Alice has made some mistake.

**Assets**

**Countermeasures**
- Ensure all messages are traceable (e.g., using a certificate) so users can demonstrate that they really received a message.
- Enable logging which assumed that messages may be lost due, e.g., to services not correctly handling all possible conditions.
- Provide methods to trace which services were involved and contact details, for each step along a composed service.

### Scenario 12A: Novice Aniketos platform users

**Description**
Bob, a developer working at a SME, is used to traditional software development processes, but he has recently started to follow a service-oriented architecture approach. Composing services is fairly new to him and when he uses the Aniketos platform for the first time, he needs to understand its infrastructure and supportive services for design-time and run-time. He easily gets an overview of the Aniketos platform and its main features. Additionally, the platform enables him to get a deeper, detailed understanding of creating and maintaining secure and trusted composite services. To discover relevant services is quite easy for him, since he can quickly find and apply a suitable methodology and tool for defining trustworthiness and risk-based security properties.
### D1.2: First Aniketos architecture and requirements specification

#### Workflow
1. At the entry point to the Aniketos platform, the service developer looks for information about the platform.
2. The Aniketos platform immediately provides an overview about its infrastructure and supportive services for design-time and run-time to the service developer.
3. Based on the given overview, the service developer is able to efficiently use the Aniketos platform for his tasks.

#### Alternate workflow
1. The service developer looks for additional, more detailed information about supportive services for design-time and run-time.
2. The Aniketos platform provides the service developer with detailed explanatory material (descriptions, tutorials …).

#### Stakeholders
- Service developer

#### Outcome
- The novice service developers can easily and quickly get an overview about the features of the Aniketos platform.

### Scenario 12B: Novice Aniketos users in trouble

#### Description
When using the Aniketos platform for the first time, Bob, the developer, has difficulties to get an overview about the features of platform, as the platform provides him too many options at the same time, and he doesn’t know where to get started. Although already frustrated by the usage of the Aniketos platform, he continues trying to identify areas of the platform (tools, methodologies), which are useful for him. Finally, he finds methods and tools, which might suit his needs, but is still angry about the waste of his time when getting started with the usage of the Aniketos platform.

#### Stakeholders
- Service developer

#### Outcome
- The usage of the Aniketos platform (for the first time) causes a waste of resources.

#### Assets
- TBD

### Scenario 13A: Security guidelines followed

#### Description
Phil, the composite service developer, needs to choose appropriate service components and is prompted with a range of security-related metrics per service component. His project team leader instructed him to follow the security guidelines (policies) defined by the security specialist of the project team. Phil is able to easily match these guidelines with the security properties and metrics of the services and finally selects the best fitting service component.

#### Workflow
1. The service developer has to follow the guidelines/policies, which have been defined by the security specialist.
2. The service developer needs to apply these guidelines to the available basic services for his composite service.
3. The service developer matches the guidelines with the data of the security-related metrics of the basic services and selects those, which fit to the guidelines.

#### Alternate workflow

#### Stakeholders
- Composite service developer, project team leader, security specialist

#### Outcome
- The composite service developer is able to match project-related security guidelines with security properties of the service components.
### Scenario 13B: Security guidelines compromised

| Description | Phil, the composite service developer, is overwhelmed by the range of security-related metrics per service component. He is wondering which level of security fits for his purposes. Although he has security guidelines (policies) available (defined by the security specialist of his project team), he isn’t able to match the guidelines with the security properties and metrics of the services. |
| Stakeholders | Composite service developer, project team leader, security specialist |
| Outcome | The composite service developer is not able to match project-related security guidelines with security properties of the service components. |
| Assets | |
| Countermeasures | TBD |

### Scenario 14A: Payment provider scenario (component developer POV)

| Description | Chris has developed a new payment provider service called ChrisPay. He has limited funds for marketing the service, but he has heard of the Aniketos Marketplace and thinks it is a good venue for promoting ChrisPay. After submitting his service specifications, it gets evaluated by the Aniketos platform; unfortunately several critical security vulnerabilities are discovered, so his service gets a low security rating. After receiving this feedback, Chris learns more about security vulnerabilities and fixes the problems in the code. After resubmitting it to the Marketplace, his service gets a much higher rating. Jason is developing an e-business solution and eventually finds ChrisPay as a low-cost payment provider with a good security rating; he integrates it into his own service. Several months later a new vulnerability is discovered that affects ChrisPay (among other services) and lowers its security rating again. Chris is notified of this, and he immediately fixes the problem. This causes his security rating to rise above his competitors’, and many more people end up using ChrisPay than before. |
| Stakeholders | Service developer, Service provider, Aniketos marketplace |
| Outcome | Many composite service providers use Chris’ service because of its high level of security. |

### Workflow
1. Developer submits component
2. Component is evaluated for trustworthiness
3. Component is added to the Aniketos Marketplace
4. Re-evaluate component whenever a new threat or vulnerability is discovered
5. Notify developer of service component if a new threat or vulnerability affecting the component is discovered
6. Restore security rating of the service when the new threats are addressed

### Alternate workflow
6b. Service developer doesn’t fix the threat or vulnerability, and consequently the security rating of the service drops.
### Scenario 14B: Insufficient automated security evaluation

| **Description** | Mallory is a disgruntled programmer for ACMEPay Ltd., owner of a highly-rated payment provider component; she also knows the exact algorithms used by the Aniketos runtime evaluation components. She uses this information to insert a backdoor/vulnerability that cannot be detected by the algorithms into the payment provider component. She then exploits this vulnerability (or waits for someone else to exploit it), which results in the release of all confidential information collected by the payment provider. |
| **Stakeholders** | Unethical service developer |
| **Outcome** | Trust loss in Aniketos as a whole |
| **Assets** | Sensitive data |
| **Countermeasures** | Manual auditing |

### Scenario 15A: Runtime service recomposition (end-user POV)

| **Description** | Judy uses a composite service (a mash-up service or a social networking site) that contains a calendar component. Since the trust level of the composite service is high, Judy decides to store sensitive information in the calendar module. A few weeks later, a vulnerability is found in the calendar component, resulting in a decrease of its trust level. Aniketos triggers a runtime recomposition – the service provider switches to a calendar component with an acceptable trust level. |
| **Workflow** | 1. A service end user uses a composite service 2. The service needs to perform a recomposition due to a decrease in the trust level of one of its components 3. A new component with the same functionality is chosen to replace the old (untrusted) one |
| **Alternate workflow** | 3b. If no such component is available, the specific functionality in the composite service is disabled. |
| **Stakeholders** | Service provider, End user |
| **Outcome** | Judy can keep using the service with no apparent changes from her side. |

### Scenario 15B: Runtime recomposition without synchronisation

| **Description** | The service provider of the composite service changes the calendar module according to the Aniketos recommendation, but there is no synchronisation function to copy data between the old and new calendar components. |
| **Stakeholders** | Incompatible service component |
| **Outcome** | All sensitive data stored by Judy in the calendar is lost. |
| **Assets** | Sensitive data |
| **Countermeasures** | Out of scope for Aniketos; providers must use a common interface for migrating data and services of the same type need to present their data in the same format through this interface. |
**Scenario 15C: Runtime recomposition with synchronisation**

**Description**
The service provider of the composite service changes the calendar module according to Aniketos’ recommendations. In order to save Judy’s sensitive data, the new calendar service transparently synchronizes itself with the previous calendar. Jack knows about this functionality and wants to obtain Judy’s (and other users’) sensitive data. He creates a calendar service with exactly the same functionality, and somehow achieves a high trust level for his service. If this trust level is high enough when the vulnerability in the original calendar service is found, his web service will be chosen for the composite service.

**Stakeholders**
Malicious service provider

**Outcome**
A malicious third party obtains sensitive data without the user’s consent.

**Assets**
Sensitive data

**Countermeasures**
User confirmation

---

**Scenario 16A: Stock exchange service**

**Description**
Donald is a business man who uses a stock quote service to see the current stock prices for certain important stocks. Suddenly, a new vulnerability is discovered, causing the trust level of his web service to decrease. His client chooses another stock exchange service based on the Aniketos trust levels.

**Workflow**
1. A service end user uses a composite service
2. The service needs to perform a recomposition due to a decrease in the trust level of one of its components
3. A new component with the same functionality is chosen to replace the old (untrusted) one

**Alternate workflow**
3b. If no such component is available, the stock quote service stops working.

**Stakeholders**
Service provider, End user

**Outcome**
Even after a recomposition, Donald will be getting accurate and trustworthy information.

---

**Scenario 16B – Stock exchange service – missing preference property**

**Description**
When Donald set his preferences about which stock exchange service to use, he only set the minimum trust level and the maximum price of the service. When the originally used web service was no longer usable because of the trust level drop, accidentally a free stock exchange service was at the highest trust level and Donald’s client recommended choosing this one. Unfortunately the free service has a 15-minute built-in delay for stock market data.

**Stakeholders**
End user

**Outcome**
Donald loses a lot of money because he does not notice that the received data is 15 minutes old.

**Assets**
Money

**Countermeasures**
Warning the user when a component with different functionality is selected
### Scenario 17A: Trustworthy Web scenario (service provider POV)

| **Description** | Bob runs a website that tracks the latest news in a specialized domain of interest. In addition to presenting own articles, the site aggregates news items from several sources. Furthermore, a forum, a means for blogging and several web services enriching the visitor experience are provided – viz. a weather forecast service and search engine bindings. The news entries’ contents may reach out from Bob’s site to reference video content – provided by on-line sharing sites –, pictures and other resources. Each external link carries with it an associated risk because every possible data format that ends up rendered by the end user’s browser is prone to malicious tampering. As Bob prides himself on his reputable and popular website, he strives to provide maximum browsing security to his readers. Therefore, he makes sure that all of the external services and links point to sources that can supply adequate trust credentials. By double checking each external source’s security credentials using the Aniketos framework, insecure links get rejected on the spot. This way he can assure himself that he provides a secure and reputable service. |
| **Workflow** | 1. A service provider maintains a content-driven site  
2. The provider can add new services to the site  
3. Both the provider and end users can contribute content  
4. The provider uses Aniketos to evaluate the trustworthiness of services and content  
5. Services and content whose trust level is above a certain threshold are allowed on the site |
| **Alternate workflow** | 5b. If the service’s or content’s trust level is below the threshold, it is rejected. |
| **Stakeholders** | Service provider, End user |
| **Outcome** | Visitors trust Bob’s website. |

### Scenario 18A: Trustworthy Web scenario (end-user POV)

| **Description** | Alice is an avid reader of Bob’s site. She likes to periodically check in and get up to date with the latest developments. She accesses the site from several different locations: home, work and internet cafes. Because Alice is a serious professional, she only accesses websites from work that are sure not to compromise her companies’ infrastructure, i.e. are trustworthy to a very high degree. As her surroundings become more and more casual, she becomes apt to relax her security considerations accordingly, in favour of some entertainment. |
| **Workflow** | 1. An end user defines a trustworthiness threshold when browsing, depending on context  
2. End user’s browser checks the trust level of each website  
3. If the trust level of a site is over a threshold, the user can view it |
| **Alternate workflow** | 3b. If the site’s trust level is below the threshold, it is rejected. |
| **Stakeholders** | End user (browser), Service provider |
| **Outcome** | Web users can ascertain whether a site they’re accessing is secure or not. |
### Scenario 19A: Futuristic home automation scenario (end-user POV)

#### Description
José decides to get a new intelligent home automation system. Since these systems can control the locks on the doors and services that cost considerable amount of money (e.g. the temperature of the house and the swimming-pool), security and reliability are the primary aspects for choosing the right product. He already has some devices that are able to provide various services to such systems (automatic doors, heaters, thermometer, etc.), so he checks with Aniketos the composite security measures of these services combined with each possible home automating system.

He finds a cheap service composition that seems to fit his needs, but he discovers with the help of Aniketos that one of the services (to be installed on his fridge) has a vulnerability. Finally José successfully finds the most secure solution that even automatically updates its own current security values whenever a new threat is found. Since he knows that there is no system that is protected against not-yet-discovered vulnerabilities, he buys the SmarterHome product from Smarty Inc.

Later on the system warns him that a new vulnerability was found in the firmware running on the garage door opener, and offers him more secure software that is compatible with his garage door and offers the same services.

When on the road, José monitors and controls his home remotely with his smartphone. He is perfectly aware of the sensitivity of the information sent over the network while communicating with his smarthome, but he stays confident because the Aniketos add-on on his phone continuously informs him on the trust-level of the network he is currently using.

#### Workflow
1. An end user wants to buy a composite service that fits his needs in an IoT context
2. A service provider uses the Aniketos Marketplace to offer a composite product that is deemed “secure” by Aniketos
3. The service components are installed onto each element in the end user’s house
4. When a new threat is discovered, the service provider automatically offers to update the firmware of all devices the threat is relevant to
5. The entire system can be managed remotely in a trustworthy way

#### Alternate workflow
- 

#### Stakeholders
End user, Service provider, Aniketos Marketplace

#### Outcome
José’s home automation system is secure against attackers.
### Scenario 20A: Web shop payment provider (end-user POV)

**Description**
John is a customer who likes to buy books from web shops. There are lots of online bookstores with various prices and payment methods. John knows that the confidentiality of his credit card number and verification code is extremely important – if somebody was to steal them, they could buy anything in his name.

John checks the payment provider component specifications in the Aniketos Marketplace, and finds information regarding the trustworthiness, security level and preferred ways of payment of each. He then uses this information to find optimal payment providers; he only buys from web shops that support these providers.

John is aware that web shops may switch payment providers over time. Therefore, he watches the Aniketos profiles of his favourite web shops – if one of them stops supporting his preferred payment providers, he takes his business elsewhere.

**Workflow**
1. End user wants to know more about service components
2. End user peruses Aniketos Marketplace to get information
3. End user chooses preferable service components
4. End user stays up-to-date with the component listing of several composite services via the Aniketos Marketplace
5. End user only uses the composite services that use his preferred service components

**Alternate workflow**
4b. If Aniketos does not support automated notification for end users, the user has to check the descriptor every time he wants to use the service.

**Stakeholders**
End user, Service provider, Service developer, Aniketos Marketplace

**Outcome**
John can be certain that the web shops he’s dealing with use secure payment providers.

### Scenario 20B: Web shops – perception-of-trust issues

**Description**
Gary is a customer who uses the Aniketos Marketplace to find the best payment provider for himself. The WrongWeb shop uses his preferred provider, SafePay, so he trusts the shop implicitly (false sense of security). However, when he makes a purchase, the received product is of a terrible quality and worse yet, the WrongWeb shop sells his contact information to spammers. The transaction itself goes without problems.

**Stakeholders**
Unethical service provider

**Outcome**
Gary blames Aniketos or the service developer for the service provider’s faults.

**Assets**
- 

**Countermeasures**
Strong definition of which part(s) of a certain system are trustworthy and how that translates into trust for the system as a whole.

### Scenario 21A: Navigation (end-user POV)

**Description**
Istvan is a frequent traveller and like to spend his time in the most efficient way during travels. Therefore he subscribed to a service which provides him with up-to-date navigation information at any time of the day, anywhere inside the service coverage area – which is currently the whole EU.

This service includes information about a wide range of transport facilities, including public transport (bus, train, boat routes, timings, prices, etc.), local taxi companies’ contact info and prices, and of course
traditional car navigation with up-to-date traffic control information. Beside navigation, the service is able to provide various valuable location-based pieces of information, like interesting local events, nearby restaurants, tourist attractions, etc. The service – which Istvan decided to subscribe to – integrates various service providers. A few of them are free and based on information collected by volunteers in open communities, others come from local or global commercial sources specialized to provide certain kinds of information: live local traffic info, local maps, local POI databases, etc. Istvan recently turned on the Aniketos service trust indicator in his device since there were a couple of reported cases where tricky service providers and hackers of free POI databases tried to cheat the navigation service in order to advertise certain restaurants or divert car traffic to certain areas by disseminating false traffic information.

**Workflow**

1. End user subscribes to a composite service
2. The service provider presents many alternatives for a given component
3. End user can see the ‘trust level’ of each provider to aid in the decision-making process by connecting to the Aniketos Marketplace
4. End user chooses and uses one of the trustworthy components

**Alternate workflow**

- 

**Stakeholders**

End user, Service provider, Aniketos Marketplace

**Outcome**

With the real-time Aniketos service trust detector, Istvan is able to avoid false information providers even when he is travelling to unknown places.

**Scenario 22A: Service replacement at runtime**

**Description**

Vasilis is a service end user/consumer and wants to have access to specific public administration services, which will enable him issuing a military services completed certificate. All he has to do is to identify the appropriate Military Office and ask for the relevant documentation. However, he is not aware of the exact public administration body, which could serve his request, as well as the exact set of documents, which should be delivered by him, in order for the certificate to be issued. In that respect, he visits a Governmental Portal and delegates this task to this portal.

The issuing certificate service can be provided by both the Central Military Office (CMO) and the Ministry of Defence. The Governmental Portal has established a trusted connection only with the CMO in order to make use of the relevant Web Service.

The process of issuing the certificate contains two steps:

- The end user is identified as the delegate to receive the service outcome (seamless user authentication to both the portal and the CMO)
- Cross checking the end user military status to issue the certificate

While identifying the appropriate service to meet Vasilis’ request, the Governmental Portal fails to connect to the CMO, due to service unavailability or failure. In this case, the Portal has to discover the existing service published by the Ministry of Defence, assess that the appropriate level of trust is ensured and make use of it to satisfy the user request.
**Workflow**
1. Access the Governmental Portal
2. Request for the military services completed certificate
3. Discover the most appropriate service replacement
4. Assess the trust level of the alternative service
5. Issue the certificate

**Alternate workflow**
4b. Trust Assessment fails, so adjust the trust requirements of the composite service

**Stakeholders**
Service provider, Aniketos marketplace, Service end user

**Outcome**
Vasili is able to get the military services completed certificate

---

### Scenario 23A: Landing plan

**Description**
A plane is experiencing some technical problems and needs to make an unplanned landing near its current position. The pilot and the air control authorities are in the need of a landing plan showing landing options and local information. A landing plan service is able to make use of several atomic services in order to provide this information. Input to the service is location, time and date, type of plain and equipment, and the service provides an annotated map with landing vectors, weather, meteorology and tidal conditions, as well as local air traffic information. The composite service must be put together by atomic services available for this region/country. In order to select the right atomic services for this area and achieve trustworthy information, there are certain requirements to availability, reliability, accuracy and integrity that must be met. There has already been specified a service template for situations like this.

**Workflow**
1. Analyse service template and perform discovery based on geospatial features and security/trust properties.
2. Select service components and validate service composition.
3. Perform dynamic binding and deploy composite service.
4. Execute composite service.

**Alternate workflow**
2b. The accuracy of the geospatial services is not in accordance with the requirements in the template. The end user is made aware of the degraded trustworthiness of the information from the service.

**Stakeholders**
Service end users (pilot, ground control), local service providers (local ground control, weather information service, map provider)

**Outcome**
A composite landing plan service creating an annotated map.

---

### Scenario 23B: Landing plan arrives too late

**Description**
The plane has limited fuel and can only stay in the air for 10 minutes. The process of performing service discovery, dynamic composition and execution usually takes about 15 minutes. The pilot is forced to make a risky landing before the landing plan is ready.

**Stakeholders**
End user (pilot)

**Outcome**
Functionality/features offered by the Aniketos platform require more computation time than is available

**Assets**
Passenger and property

**Countermeasures**
TBD

---

5 This scenario is inspired by the example application in "A Framework for Automatic Web Service Composition" by Anya Kim, Myong Kang, Elias Ioup, Catherine Meadows, and John Sample. Naval Research Laboratory, NRL/MR/5540—09-9191, April 30, 2009.
Scenario 24A: Bank loan

**Description**

Joe would like to loan some money in order to buy his dream boat. He uses a service called LoanCrawler that finds the best loaning conditions among a set of online banks. LoanCrawler wants to protect Joe’s privacy, but at the same time certain personal information is requested by the banks in order to perform credit checks, often performed by underlying third party credit bureau services. When LoanCrawler contacts a bank to investigate the possibilities of a loan, it receives a request of personal data about Joe. LoanCrawler will only give out the minimum set of information, so the response is likely to by somewhat different between the various banks. LoanCrawler and each bank will initially set up a contract on what this personal data can be used for, how long and secure it will be stored and to whom it might be further shared. The bank might have to use another credit check service than the default if the privacy requirements from LoanCrawler are very strict. In the end, Joe receives loan offers from three banks, while two other were not able to respond due to what they consider to be incomplete credit checks based on their available information about Joe. Two weeks later, Joe sets sails for in the Mediterranean sea.

**Workflow**

1. Joe requests bank loan alternatives from LoanCrawler.
2. LoanCrawler acts as an agent for Joe and checks with a number of bank services for their loaning conditions.
3. LoanCrawler and each bank service negotiate a contract on the use of private data.
4. The bank services use a credit check service that is in accordance with their negotiated contract.
5. The bank is able to make an offer that is sent to Joe.

**Alternate workflow**

5b. The bank is unable to run a proper credit check due to the strict privacy requirements.

**Stakeholders**

Service end user (Joe), agent service (LoanCrawler), banks, credit check bureaus

**Outcome**

Joe receives loan offers from a set of banks and knows that his personal data has been regulated and protected by contracts.

---

Scenario 25A: Multi-user service for ambient assistant living

**Description**

Ole wants to create a service to help his aging mother Berit to remember her doctor’s appointment. The service should keep track of the appointments, reminding her when its time to go, helping her find the way to the doctor’s office and let him keep track of her progress and alerts him of problems.

As Ole is not an IT expert, he gets guidelines for how to compose services according to his needs. He has a set of service requirements, e.g., support Berit’s needs with minimal user interaction and allow access from different terminals (e.g. for Berit, Ole and health centre personnel). Ole then composes the service from the following service components he finds in an online database: calendar,......
scheduler, bus info, location service, alert, messaging, route (itinerary) planning, logger, travel assistant (guider or navigator), smart home sensor services. There are typically several variants for similar services with different features and quality of services. Ole is free to select service components according to his needs.

With the composed service, Ole can connect to the service from his computer and enter the time and date for her next doctor’s appointment. The service has access to the bus schedule, and calculates what bus Berit should use and when she should leave the house to catch the bus in order to make the appointment in time. The service alerts Berit via messages displayed on a TV at home and on her mobile phone as well. The mobile phone tracks her position, and a sensor at the front door can register when she leaves the house. The mobile phone assists Berit on her way to the doctor by e.g. tracking her position and reminding her of which bus stop to get on and off. Messages are sent to Ole’s phone and email to alert him of the appointment, her mother’s leaving, and any deviation from the plans/schedules. Ole can also view the logs to check her mother’s progress and the status of the service. The health centre of Berit’s doctor could also register Berit’s appointments and keeps track of Berit’s progress when she has her appointment.

The service should be composed according to security properties, e.g., high security requirements in untrusted environment (on the street) than in trusted environment (at home). For example, there are two travel assistant services in the city: FreeGuide is a free service, but has public access to user’s whereabouts (e.g., any user can see who are moving around nearby), while SecureGuide requires a fee but does not reveal user’s whereabouts. Ole may select SecureGuide to protect Berit’s privacy.

The service can be recomposed at runtime, and Ole should be informed if the composition has significantly reduced security/trustworthiness. For example, if real time bus information is not available, a fixed bus schedule can be used for bus information. Ole (as a service composer) may be informed about this since the service composition has reduced QoS (real time vs fixed info), but Berit (the end user) does not need to be informed in this case.

**Workflow**

1. Service composer finds guidelines and howtos
2. Service specification with socio-technical requirements (e.g. minimize user interaction)
3. Search for service components according to specific needs
4. Select among suitable service components based on security properties and other QoS properties
5. Integrate service component and validate the service
6. Deploy service composition on several machines, e.g., Ole’s PC, Berit’s PC and mobile phone, health centre’s terminal
7. Recompose services at runtime to replace an unavailable service with a similar one
8. Inform composer when composition has reduced security/functionality

**Alternate workflow**

**Stakeholders**

Service developer and composer, Service provider, Aniketos marketplace, end user

**Outcome**

Ole can compose a multi-user service according to his needs. Berit can live happily mostly on her own and Ole does not need to worry about her.
### Scenario 25B: Compromise of privacy

**Description**
Berit’s calendar contains other appointments and activities besides doctor’s appointments, which is rather private and should be kept within the family. However, her whereabouts and activities (of the family as well) could be revealed to irrelevant persons as the health centre where Berit goes to has also access to Berit’s calendar and her position information.

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Outcome</th>
<th>Assets</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unethical end user</td>
<td>Privacy is compromised</td>
<td>Sensitive personal information</td>
<td>Limit the access to sensitive information, access rights</td>
</tr>
</tbody>
</table>

### Scenario 26A: Auction service

**Description**
Trust is an important issue for an online auction system, such as eBay. An online auction service offers several options of performing payment. These are service components from different payment service providers, such as PayPal, credit card companies and banks. As the payment service providers and end users may not trust each other, some payment providers do not accept end users, while end users do not accept all payment providers. The auction service should be able to select payment service according to the trust information.

An online auction system MyAuction provides several payment options: PayPal, credit card and bank transfer. Bill is a user of MyAuction and has a PayPal account. In Bill’s profile, he gives his credit card information and indicates to use his credit card from MyCreditCard company as the default payment method. As MyCreditCard is a new credit company for MyAuction, MyAuction determines the trust level of MyCreditCard and makes a contract with MyAuction specifying a set of security requirements. MyAuction creates also a contract with Bill to use MyCreditCard for payment.

Bill buys a new mobile phone via MyAuction and asks for payment. MyAuction checks the contract with Bill and MyCreditCard against the trust and security situation and makes payment.

<table>
<thead>
<tr>
<th>Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Service composer determines the trust level of the payment service and the end user</td>
</tr>
<tr>
<td>2. Create SLA with security level and trust levels</td>
</tr>
<tr>
<td>3. Check contract and trust properties at runtime</td>
</tr>
<tr>
<td>4. The contract and trust level is acceptable, so the payment is performed via MyCreditCard successfully.</td>
</tr>
</tbody>
</table>

**Alternate workflow**
4b. Detect deviation of trust level from the contract: As the online payment system of the credit card company MyCreditCard has a security flaw and is been attacked by malicious users, Bill’s default payment service becomes untrustworthy.

5. SLA chain effect: as the contract between MyAuction and MyCreditCard is violated, the contract between MyAuction and Bill is also violated. MyAuction creates a new contract by selecting PayPal according to Bill’s profile and continues with payment

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service end user, Payment and Auction service providers</td>
<td>Bill can perform auctions without worrying about the payment methods</td>
</tr>
</tbody>
</table>
Scenario 27A: Environment monitoring sensor network

<table>
<thead>
<tr>
<th>Description</th>
<th>This scenario is about underwater sensor networks for environmental monitoring of the seas. Such networks typically involve a number of underwater sensor nodes located in a more or less regular pattern in the sea, taking measurements, and communicating them to a central measurement database through relay nodes, e.g. buoys, passing ships or autonomous underwater vehicles (AUVs). Furthermore they embed a highly distributed software system, controlling the operation of the sensors and the transfer of the sensed data to the target destination. In this scenario we assume a service oriented architecture where sensors dynamically select suitable relay nodes to send their data to the central database based on QoS properties of the available services. Under normal conditions the perceived threat level and security requirements are low, so data are transmitted unencrypted, as this saves energy in the sensor nodes and thus prolongs their lifetime. However, abnormal conditions may occur, for example accidents causing spill of chemicals or radioactivity into the sea, where collected data is regarded as sensitive information which should be shielded from unauthorised access, and encrypted communication and only highly trusted relay nodes should be used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow</td>
<td>1. The sensor network is operating under normal conditions without encrypted communication. 2. An accident occurs causing spill of radioactive material into the sea. 3. In this situation the emergency management organisation does not want data to be available to the public and increases the security requirements and the threat level. 4. The sensor network reacts by reconfiguring to encrypt the data sent over the network and to only use highly trusted relay nodes. 5. After a week it becomes clear that the leak was small and caused no serious consequences. Thus the security requirements and the threat level is set back to normal. 6. The sensor network reconfigures to transmit data unencrypted and enable the use of less trusted relay nodes again.</td>
</tr>
<tr>
<td>Alternate workflow</td>
<td></td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Emergency management organisation, Sensor network owner</td>
</tr>
<tr>
<td>Outcome</td>
<td></td>
</tr>
</tbody>
</table>
A system architect of a Lead System Integrator (LSI) starts working on a brand new very large critical information system (CIS); this kind of system is typically known as a system of systems (SoS). Due to its considerable size and complexity, this (civil) contract imposes the use of the NATO Architecture Framework (NAF) for the operational analysis; the tooling implementing NAF can be freely selected. For the following phases (i.e. system/software analysis, logical architecture, physical architecture), the customer has not imposed any specific framework but a service oriented approach is required. The LSI has selected its own proprietary framework. These choices cannot be changed, but additional methods and tools can be added in order to implement processes not covered by NAF or the proprietary framework.

At first the CIS is described as a black box. Through his work the system architect starts modelling the system in terms of different services that interact between themselves and with external actors and/or the system environment. Early in the contract, the customer involvement is important and part of the system architectural work will consist in eliciting and clarifying user requirements.

At a certain point during the architecting work, and as scheduled in the risk management plan, a system security engineer is called to intervene. At this stage, the security needs are still vague. The system security engineer works with the customer to elicit and clarify the security needs. He progressively models the customers security needs as trust contracts. The system security engineer uses the system architect’s work to understand the scope of the security analysis. He also uses the decomposition of the SoS in services to specify trust contracts on each of the sub-services identified by the system architect. His tooling environment allows him to check online8 the consistency of his trust contracts when multiple services are combined to provide a higher level service. At this stage, the system security engineer does not know if the services will be rendered by people, equipment (hardware / software) and/or procedures.

For some time, the system architect and the system security engineer work in parallel on the same CIS model. The tooling ensures model consistency9. When the system security engineering has finished his work and that his analysis has been officially approved (internally or by the customer), the tooling allows him to “validate” his modelling work. This has for effect to “lock” the services for the system architect. The “locked” services can be further decomposed, but they cannot be fused or split, and their interfaces cannot be changed without invalidating the security work.

The granularity at which the system security engineer works may vary from between services, but in all cases it corresponds to a service that the LSI will subcontract to another company or assign internally to a development team within the company. This model-based contracting is performed in an identical way for internal development teams or external sub-contracting; the only differences are on the legal level, but this is out of scope of the present scenario. The model-based contracting is supported by: (i) the NAF architecture; (ii) the trust contract set on the subcontracted service.

Six months later, the LSI is provided with the sub-contracted service by its sub-contractor or internal development team. The LSI tests that the trust contract is indeed satisfied before integrating the new service in the CIS.

After integrating all sub-contracted services, the LSI can provide the customer the complete CIS with full confidence that the global required trustworthiness will be provided. The conformance to the trust contracts is monitored and evaluated at run-
time by the end-user.

<table>
<thead>
<tr>
<th>Workflow</th>
<th>TBD</th>
</tr>
</thead>
</table>

**Stakeholders**

- Customer assumed to be here both the contracting party and the end-user; in conditions when this is not the case, the customer is assumed to allow access to the system end-user.
- Lead System Integrator (LSI) represented by a system architect, a system security engineer, and some internal development team(s)
- LSI external sub-contractor(s).

Note: the role “system security engineer” is used here to represent a role that does not yet exist, with respect to the work on security contracts as they will be elaborated by Aniketos. It is subsumed that the system security engineering will need the competencies of both a security analyst and a security architect.

**Outcome**

Customer security needs are mapped to trust contracts. Trust contracts are used for model-based contracting.

---

**Scenario 29A: Designing and implementing a secure sub-system**

**Description**

A software development team receives a model-based contract to provide a service within 6 months. This service will be integrated in a large Critical Information Systems with stringent security requirements by a Lead System Integrator (LSI). The model-based contracting is supported by: (i) a NAF architecture; (ii) the trust contract on the service to be developed. Some details of the general architecture which are not of use might be obfuscated. The development team is free to implement the service as its likes, as long as it can prove that the service and trust contracts hold.

At first the service is described as a black box. Through his work the software architect starts modelling the service in terms of different functions and sub-functions. Functions are then mapped on logical components and logical components are mapped to a physical infrastructure.

At a certain point during the architecting work, and as scheduled in the risk management plan, a security analyst is called to intervene. The security analyst uses the architect’s work to understand the scope of the security risk analysis, and the service trust contract as a “target security level”. He also uses the decomposition of the service in functions, logical components and physical components to select the supporting assets to protect. He performs a threat & vulnerability analysis. Finally the security analyst specifies security objectives and security requirements.

Risk mitigation is performed by the security architect, who specifies the adequate safeguards to satisfy the security requirements.

For some time, the software architect, the security analyst and the security architect may work in parallel on the same models of the service. Consistency and traceability are maintained at all times.

The developers code and integrate the service.

Compliance to the trust contract is tested before delivery to the LSI in due time.

<table>
<thead>
<tr>
<th>Workflow</th>
<th>TBD</th>
</tr>
</thead>
</table>

**Stakeholders**

- System/software development team represented here by a software architect, a security analyst, a security architect, developers and integrators.
- Lead System Integrator (LSI) viewed here as a customer by the system/software development team.

**Outcome**

Trust contracts are used as input to security risk assessment. Model-based security risk assessment run as a specific viewpoint in parallel to the mainstream system/software engineering allows for overall consistency, traceability and productivity gains.
### Scenario 30A: Transparent contextual service composition

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
</table>
| A European ISP company aims to provide customers with technical support on 24/7 basis. The ISP company provides technical support via its staff, and Dan is employed here. The company’s main goal is to enable Dan to communicate with customers anytime/anywhere, maximizing security and quality of service while minimizing costs. Thus, the ISP company signed a contract with a mobile phone operator to enable Dan to support customers wherever in Europe. Dan is provided with free voice calls and a full-fledged Internet connection, which includes support to various services: VOIP, IMAP mail access, file sharing. Unfortunately, the contract does not provide convenient rates when technical managers are located outside Europe. To cope with such limitation, Dan’s company developed a client that is installed on smart-phones, which automatically selects the best service composition so that Dan can contact customers. Such client also provides a common user experience to the user, and is an example of unified communications. The composition is transparent to Dan; he can however specify his policies in terms of needed communication services, quality of service, and security. The client will compose on the fly different services:

- A roaming operator that provides basic voice calls. The client can negotiate temporary contracts with these operators. The most trusted operators will be prioritized, but there is also a certain trade-off between quality and cost. The same operator might offer different services at different prices.
- An internet provider service (which might be different from the roaming operator) to let Dan exploit basic Internet functionalities (surfing the web and checking e-mail). The contract will have to guarantee a high-level of availability.
- A VOIP service that lets Dan contact his customers at convenient rates. Also, the VOIP service should provide robust encryption of the calls.
- A file sharing service that enables Dan to connect to his company’s File server. Such service should ensure that no-one will sniff either Dan’s files or access the company’s files, not even the local country government.

Service recomposition is often required due to several causes: (a) Dan is a world traveller, therefore a recomposition should be performed for every country he visits; (b) Some service providers might prove to be unreliable and violate their contract; (c) Dan’s company trust about service providers changes over time; (d) Dan decides to modify his policy.

Whenever an adequate service composition is not viable due to laws that forbid the usage of some service in the country, the client might set up a VPN connection with a server in Europe, so that Dan can access all services through the VPN. However, the establishment of the VPN itself should guarantee adequate data encryption.

<table>
<thead>
<tr>
<th>Workflow</th>
</tr>
</thead>
</table>
| 1. Technical manager is offered access to composite services to support customers effectively, regardless of his location  
2. The best service composition is chosen on the fly based on the manager’s policy  
3. Composition is made transparent to the manager by a client installed on his mobile device by creating contracts with service providers |
4. Whenever required, the client performs a service recomposition to deal with contract violations, policy changes, and context changes.

<table>
<thead>
<tr>
<th>Alternate workflow</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholders</td>
<td>ISP Customers, ISP Company, Technical Manager, Mobile Operator, Roaming Mobile Operators, Online Service Providers (VOIP, VPN, IMAP, P2P)</td>
</tr>
</tbody>
</table>
| Outcome           | • Customers have full time support by the ISP company  
|                   | • Dan can contact customers regardless from his location  
|                   | • Transparent service recomposition to allow Dan interact with customers irrespective of context changes, policy modifications, service level agreement violations |

**Scenario 30B: Dan doesn’t want to disclose his identity**

| Description | Dan’s policy specifies that he wants to remain anonymous to the roaming operators and service providers when outside Europe. In a normal protocol run the Provider should never know the identity of Dan.  
However, there are several factors that inhibit Dan to keep its identity secret. This might apply, for instance, if all VOIP service providers accessible in the country require consumers to provide their identity. Maybe such policy is due to laws in the country.  
In order to deal with this issue, if Dan decides not to disclose his identity, he might exploit a service that creates a temporary identity (e.g. a temporary e-mail account). The client will create a new service composition that includes the service creating the temporary identity provider, so that the VOIP service is used via the temporary identity. |
| Workflow | 5. Dan needs to use a VOIP service that requires him to disclose his identity, though he doesn’t want to  
6. The client enacts a service recomposition that involves a temporary identity provider, so that the VOIP service can be used |
| Alternate workflow | - |
| Stakeholders | Technical Manager, Mobile Operator, VOIP Service Provider, Temporary identity Provider |
| Outcome | 1. Dan can use a VOIP service without disclosing his identity |
### Scenario 31A: Renting a movie

**Description**
An end-user uses his smart-phone to access a web-portal service to select a video (movie) he wishes to purchase while he is travelling back home (he is a commuter and he wishes to watch the movie when he is back home). In order to get access to the purchase page of the portal the user needs to be (previously) logged into the portal.

Logging into the portal (to get access to its various services) means that the user needs to provide the username and password (the credentials) [The end-user provides his credentials to the web application].

His identity is managed by an Identity Management System, which allows the user to access a group of services/applications authenticating only once.

The end-user usually selects his movies from this web-portal because he knows when doing so the movies are legal ones and he is not incurring in the risk of infringing any law concerning copyrights, etc.

In addition to this, sometimes he downloads some adult-rated movies and he knows that this service provider doesn’t use his data (e.g. e-mail for spamming…); privacy in this case is of paramount importance.

However the movie he has chosen is not currently available from his preferred web-portal and the system offers to select the movie from another site (another provider) – this could be done automatically while keeping the same level of “legality” by the web-portal (the system accesses a federated service-provider and therefore the user doesn’t need to give his credentials again and in addition he can rest assured that the movie is “legal”). In this case the other alternative provider is a trusted one (the trust is pre-determined by the system).

Alternatively (especially in the case the other service-provider is not federated) the user gets a warning informing him that he is going to get his movie from another different service provider: At this point he can continue or refuse the transaction. In this case the other alternative provider is not trusted (and the system could not establish an acceptable level of trust).

**Workflow**

1. A service end-user provides his identity information (just once) and uses a composite service form his preferred web-portal in order to choose a movie to purchase/download.
2. Services offered are trusted (and the trust levels are automatically determined).
3. An alternative service-provider is being used/proposed but…
4. The alternative provider is not trusted enough (the system could not determine an acceptable trust level based on end-user requirements and/or based on other parameters).
5. The end-user receives a warning about the different trust level and at this point he can refuse the new alternative provider
6. In case the end-user refuses the new alternative service provider the system removes it from the trusted pool of service-providers.

**Stakeholders**
Service end-user, service providers, network providers

**Outcomes**

1. The end-user can access several composite services with a single log-in to his preferred web-portal
2. The trust of the service providers is automatically determined/guaranteed
### Scenario 32A: Dynamic service analysis and adaptation

**Description**
During a public event a crisis situation occurs that requires the assistance of the emergency services. The Fire service sets up a Silver Command Centre on the periphery of the event to interact with the Gold Command Centre at the Police HQ. The event organiser already has a set of services that provide access to their management and billing databases, and which contain a list of attendees at the event.

During the crisis situation, it’s realised that the organiser’s database is of particular use to the Fire and Police services, which are able to use it in managing the crisis (e.g. by allowing the identification of attendees with medical conditions).

The Fire service, using its existing service deployment, connects to the event organiser’s service in order to interact with the attendee database. However, the Fire service is already connected to the Police network, and this new connection introduces a potential data flow vulnerability that may allow the event organiser access to restricted Police databases.

The Aniketos platform immediately identifies this risk, and to mitigate it spawns proxy Police and Fire access services to ensure there is no sensitive data flow from the Police network back through to the event organiser via the Fire service.

**Workflow**
1. Fire service connects to untrusted external service.
2. Aniketos platform analyses the new service composition and identifies a security risk.
3. Aniketos platform mitigates the risk by spawning new (or identifying existing) services to act as proxies to ensure secure data flow.
4. Administrators are notified about the change.
5. Analysis continues each time the service composition changes to ensure security is maintained.

**Alternate workflow**
3b. The Aniketos platform is unable to identify a suitable secure alternative, and the link to the external service must be refused.

**Stakeholders**
Service users, Aniketos platform.

**Outcome**
The Fire service is able to access the organiser’s attendee database without compromising the security of the Police network.

### Scenario 32B – Adaptation impacts functionality

**Description**
During a crisis situation, the Fire service connects to the organiser’s database service in order to access attendee information. The Aniketos platform discovers and acts on a perceived risk to the Police service’s data, spawning additional services and re-composing to compensate.

A user of the Silver command Fire service systems tries to access the Police database. Rather than using the existing services, they try to do this from the newly created services, and due to the restrictions on data transfer find themselves unable to access the data they need.

The result is a delay in access to the appropriate data, which could result in serious consequences for the management of the crisis.

**Stakeholders**
Service users, Aniketos platform.

**Outcome**
Access to data is restricted

**Assets**
- Access to data is restricted

**Countermeasures**
TBD
**Scenario 33A: Composing services across insecure networks**

| Description | The SuperWidget company has a set of services that are used as part of their design workflow that handle the blueprints for their famous widget components. These services are all managed in-house on a protected network. They’ve recently teamed up with DoobryWorld to work on a new joint widget design, and are planning to integrate their two design workflow systems together. With the aid of the Aniketos platform they are able to generate a new composite service to allow this integration to take place. However, because the services are now interacting across multiple sites, data is now passing over a set of insecure Internet links. The Aniketos platform discovers this and cryptographic capabilities are enabled or recomposition takes place in order to maintain security levels. Once the collaboration is complete the two systems are disentangled. In this case, the additional unnecessary cryptographic measures can be removed in order to ensure the efficiency of the system. |
| Workflow | 1. Two sets of services are composed across an untrusted network.  
2. The Aniketos platform discovers a potential risk to data security through analysis of the composition and information about the network.  
3. The Aniketos platform automatically and dynamically adds cryptographic capabilities or services in order to maintain security.  
4. The services are ‘decomposed’  
5. The unnecessary cryptographic capabilities are removed. |
| Alternate workflow | - |
| Stakeholders | Service provider, service users, Aniketos platform. |
| Outcome | Data security and efficiency are maintained respectively. |

**Scenario 34A: Policy Reconciliation**

| Description | A high security banking system uses assured Multi Level Secure (MLS) service components to manage their internal trading systems. The system is dynamic, connecting to other assured systems when needed to perform time-critical operations. The policies of the system being connected to must be reconciled carefully to ensure they are consistent and that appropriate security mechanisms are put in place when mandated by a change in policy (e.g. when connecting to a system that has particular security requirements). When performing connections, the Aniketos platform automatically performs policy reconciliation, acting on the result to ensure security is maintained. When automatic solutions to security mismatches can be found (e.g. the system needs an updated firewall or set of firewall definitions and these can be automatically sourced) the platform performs the adaptation. When an automatic adaptation can’t be achieved, the system must prevent the connection, or alert an administrator to the problem. |

---

10 Multi Level Secure Systems are generally accredited systems allowing simultaneous access to multiple sensitivity levels of data, often using mandatory access control such as Bell-La Padula.
### Scenario 34B – Policy negotiation

**Description**
In the event that multiple policies cannot be reconciled, negotiation might take place between service providers. A malicious service provider might use this opportunity to try to affect the security policies of a service to make them weaker in order to attack the service.

**Stakeholders**
Malicious service provider, trustworthy service provider.

**Outcome**
Policy requirements are reduced, making a system vulnerable to attack.

**Assets**
-

**Countermeasures**
TBD

### Scenario 35A: Annotating images

**Description**
Angus is creating a service composition for annotating images with personal information stored in a separate database. The service is intended to match the image with data from the database, and then annotate the image appropriately.

Angus’s service must adhere to his company’s data policy, which stipulates that the personal information can only be accessed by or provided to people with special clearance. Consequently, any service accessing the database of personal information must itself have an appropriate access level.

Although the original images are not private, once annotated they must then be handled appropriate to the higher access level.

The access level of the annotation service must be sufficient for it to have full access to the personal information database. However, not all of the images passed into the service will actually be annotated. Angus is able to pre-certify his annotation service to guarantee that no personal information leaks into the images that are not altered.

The result is that, in complying with the policy, the Aniketos platform increases the access requirements for those images which are altered by the annotation service. However, for those that are not altered, the Aniketos platform retains their lower access requirements, even though they have passed through a service that has special clearance.

**Workflow**
1. Angus creates a service to connect to another service requiring high access level.
2. Angus pre-certifies his service to prove that only some of the data managed by it is ‘contaminated’ by the data requiring high access.
3. Once deployed, appropriate access levels are applied to the data.
output from Angus’s service, so that only contaminated data is set to have the higher access level.

**Alternate workflow**

- 

**Stakeholders**

Service provider, service users, Aniketos platform.

**Outcome**

The access level of the data is set appropriately, so that access to data of low importance does not become restricted.

### Scenario 35B: Access level inflation

**Description**

Angus is not able to certify his service to show that unaltered data does not provide a channel for leaking personal information. In this case, all of the images that pass through the annotation service must have their access levels increased. This is the case even if the image itself is altered in a way independent of any private data.

**Stakeholders**

Service provider, service users.

**Outcome**

Access levels of data increases needlessly.

**Assets**

- 

**Countermeasures**

TBD

### Scenario 36A: Making Aniketos available to stakeholders

**Description**

Kostas is a service developer, who exploits the design time support of the Aniketos platform to extend his development offerings with security assertions, focusing on the trust level of the provided services. In order to announce his services, have access to the methodologies and make use of the tools of the Aniketos platform, he needs be a subscriber. The subscription level should be classified as for commercial use, since security assertions target to sensitive and highly commercial oriented applications and systems. Thus, the pricing model to be applied to Kostas’ subscription to Aniketos is a per usage fee, which offers Kostas the capability for designing composite services, taking into account the definition of various trust levels.

After having developed and deployed a composite service, Kostas decides to exploit the runtime support of the Aniketos platform determine an up-to-date trust level and monitor the security status of the service components involved. In that case, Aniketos is exploited by Kostas to handle the dynamic aspects of composite service adaptation and recomposition. While a monthly flat rate is necessary for supporting the composite service provisioning at runtime, with typical trust levels, this rate should be transformed to a per usage pricing model for ensuring that a higher security level is maintained. While a specific service offered by Kostas becomes popular and gets high rating scores, the flat rate fee for maintaining Kostas’ service into the Aniketos marketplace may be subject to discounts or other offerings (i.e. free use of the runtime support for a set of services).

On top of that, Kostas recently released a new trust-based composite service and wants to promote this service to potential application providers and/or service designers through the Aniketos marketplace. In order to do so, the Aniketos marketplace offers Kostas the capability for advertising this service, by paying an additional flat rate fee.

Vasilis is an application provider who decides to interact with the Aniketos platform (through the selected service provider) in order to consume the published services, based on configurable trust and security
levels. In principle, during application execution, the interaction with the platform is transparent to Vasilis and it is based on the service provider, who has made a contract with the Aniketos marketplace and access the set of the offering services through it. Thus, application providers and subsequently service end users should be granted access to the Aniketos marketplace for free.

**Workflow**

1. A service developer registers into Aniketos
2. Aniketos invokes an online payment service to enable the service developer to access Aniketos
3. Subscription credentials are used to verify that the specific service developer is granted access to the design time support tools, methods and marketplace functionality.
4. A Service Provider registers into Aniketos
5. Aniketos invokes an online payment service to enable the service provider to access Aniketos
6. Subscription credentials are used to verify that the specific service provider is granted access to the runtime support tools
7. Service recomposition and/or service adaptation to higher trust levels is enabled through additional subscription fees on per usage basis
8. An Application Provider enters the Aniketos marketplace to discover the most appropriate services to be exploited in the deployed applications

**Alternate workflow**

- A Service Provider requests for an advertising service to promote the release of a new service

**Stakeholders**

Service Developer, Service provider, Aniketos marketplace, application providers

**Outcome**

Aniketos stakeholders access the provided functionalities, based on their role and the subscription level.

---

**Scenario 37A – Trust-Driven Composition of Services**

**Description**

A service provider wishes to offer a complex service to a service consumer. This could be the composition of simpler ones in such a way that the consumer request is satisfied. Although the services used in the composition are hidden to the consumer.

The composite service is built by a certified compositor that is in charge to release the service in addition to its security contract and its level of trust. Both the level of trust of the complex service and its contract are automatically calculated as function of the contracts and the trust levels of the compounded services. The topology and dynamics of the service workflow influence such trust level.

In order to combine services for generating the complex service, it is important to measure both the level of trust of the complex service, i.e., how much the consumer trust the complex service respects the contract, and the level of trust of the services used in the composition, i.e. how much the Compositor trusts each services of the composition.

**Workflow**

1. A consumer asks for a complex service
2. Composition of simple services based on their contracts and their level of trust
3. Provide the complex service with a contract and a level of trust
4. Execute the complex service and monitor its behaviour.

**Alternate workflow**

4b. If the execution of the complex service does not respect its security contract, then the level of trust has to be downgraded for the
corresponding services. Indeed, if the trust level of a composed service changes, then the composite service must be in the position to replace it, without altering its own trust level.

If the composite service trust level is updated due to a failure in the security contract, then the corresponding failing sub-services must be identified and its trust level revoked.

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Service developer and composer, service provider, service consumer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome</td>
<td>The consumer obtains a service that satisfies his requirements.</td>
</tr>
</tbody>
</table>

**Scenario 38A: Productive usage of IDE integrated tools and plug-ins**

**Description**

Tom, a service developer, uses multiple and highly customized IDEs (Eclipse, Sun NetBeans and Microsoft Visual Studio), which he learned in a lengthy learning process. He regularly uses at least 3 plug-ins and he is open-minded towards additional integrated tools and plug-ins if they increase his productivity. Nevertheless, he is sceptical at first, whether the Aniketos design-time support modules, integrated in his IDEs, can be used in a time-saving manner. Based on his previous experience, it’s highly problematic if the plug-ins or additional integrated components do not provide sufficient documentation, training, and support, and if the longevity of the provider can’t be guaranteed. The Aniketos design-time support modules can prove that they provide Tom with sufficient documentation, training, and support, and that they will be available long-term. After learning and applying the basic features of the modules, he is convinced that using the modules in the IDEs don’t hinder his development productivity.

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Service developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow</td>
<td>TBD</td>
</tr>
<tr>
<td>Alternate workflow</td>
<td></td>
</tr>
</tbody>
</table>

**Workflow**

TBD

**Alternate workflow**

**Stakeholders**

Service developer

**Outcome**

The service developer can efficiently and easily use the integrated tools to analyse the service components and the composite services.

**Scenario 39A: Modelling and Reasoning about secure and trustworthy services**

The CEO of an IT company is worried about the electronic exchange of confidential documents (e.g. financial reports, employment contracts, invoices, market analyses) among managers of different departments, the president and vice-president, and the CEO himself. Currently the company is not enforcing any security mechanism to ensure confidentiality, integrity, authenticity, etc. Some of the people exchanging these documents use to travel all around the world, and send/receive messages (including electronic copies of these documents) through unreliable communication channels.

The CEO has heard of the Aniketos platform, which seems a promising solution to improve this scenario. Thus, the company hires Mike, an Aniketos-certified engineer. Mike starts his job by using the Aniketos design-time part of the platform to model the problem space.

Mike will design a dynamic service-based system. Instead of relying on a

---

11 The content of this scenario is based on survey results:
single mobile operator and its default roaming service, Mike wants to exploit on-the-fly composition of communication services. These are temporary contracts with service providers for a variety of services: voice, data connection, web browsing, VOIP, IMAP e-mail, file sharing, VPN, and so on. Mike plans to develop an Aniketos-compatible client to install on the mobile devices of managers. The client will take care of security by defining adequate service compositions that ensure secure exchange of documents. Composition and recomposition will occur transparently to the users.

Mike starts by modelling the organizational setting, using the Aniketos design-time platform, in terms of the social actors and technical systems. At this development stage, Mike is interested in organizational modelling at the business level. Therefore, technical systems are abstracted by their stakeholders. For instance, a software service providing file sharing is abstracted in terms of the organization providing that service. Among the roles Mike identifies, there are the different departments, their managers, the vice-president, the president, the CEO, the CIO, etc. An initial understanding of the business relations between these actors is provided in terms of social dependencies. For instance, the financial manager depends on the vice-president to get the financial report approved, and on the administrative department to process payments.

Secondly, Mike analyzes the requirements of the actors in the organization. We focus here on the goals of the financial manager. While modelling requirements, Mike should represent complementary perspectives:

- The goals that the system should meet. For instance, a main goal of the financial manager is to ensure employees will be paid by fulfilling the financial report. This goal can be decomposed and refined hierarchically: timesheets should be collected, the financial report should be prepared, the report should be approved, and payments should be processed. To achieve these goals, the financial manager depends on other actors: personnel should fill timesheets, the vice-president should approve the timesheet, the administrative department should process payments. There might be alternative ways to achieve these goals. For instance, the financial report can be approved by the CEO, and payments can be processed by external companies.

- The trust relations between involved actors. At the organizational level, trust can be applied to the fulfilment of goals, execution of tasks, provision of resources. For example, the financial manager might trust the administrative department to process payments, while he distrusts external companies.

- Security policies expressed at the social/business level. These policies express some properties that can be verified against the requirements. Examples of security policies that should be checked are: need-to-know, untrusted delegation chains, unauthorized delegation, trust conflicts, etc.

- The influence of the surrounding context on goals. For example, the dependency on external companies might be allowed only if the administrative department has too much work. Also, the financial manager might be unable to send the report to the vice-president if a secure connection cannot be established.

- The quality expected from the system. For example, availability should be higher or equal to, the cost per minute should be less than, encryption
should be performed with a key of at least x bits, etc.

- **Risks and opportunities.** Risks endanger the mission of the system, whereas opportunities can be exploited to perform better. For example, an event such as “vice-president goes on holiday” might be a threat to get the financial report approved, which might result in delays in salary payments. Conversely, an event like “vice-president authorizes administrative manager to approve financial reports” is an opportunity for the financial manager to get the financial report approved and speed up payments.

While modelling, Mike runs different automated reasoning algorithms on the models he created. Automated reasoning can be invoked through the Aniketos design-time platform support, and typically relies on state-of-the-art solvers. For example, Mike wants to verify (i) the possibility to achieve a goal given a certain context; (ii) the impact severity of a set of risks; (iii) the existence of alternatives (variants) to satisfy a set of goals; (iv) the violation of security policies. These techniques help Mike refine the models.

After this modelling and reasoning phase is terminated, Mike exploits these models to define a business-level architectural model of the system. The architecture is defined without prior knowledge about the actual participants. The architecture is expressed in terms of the social commitments that agents (actors) playing certain roles will make and take at runtime. Commitments define interfaces between different social actors in a contractual form. The architecture should ensure trustworthiness via an adequate set of commitments. Some commitments might already exist in the organization (e.g. the commitment of the administrative department to keep financial reports confidential). Other commitments should be guaranteed by the system-to-be. For instance, if the finance manager distrusts an external company to process payments, he might introduce a third-party trusted actor committed to verify payment processing.

Mike can run automated reasoning techniques against the architectural models. He will be interested in checking properties concerning the trustworthiness of the architecture. This means, for example, verifying if commitments include compensations (in case of commitment violation), the availability of composed services in certain contexts, the existence of alternative providers for a given service, etc.

| Workflow | 1. Model the dependencies between actors in the organizational setting  
| 2. Model the requirements of the stakeholders via goal models  
| 3. Reason about security, trust, and risk in the goal models  
| 4. Derive a trustworthy architecture based on commitments  
| 5. Reason about security and trustworthiness of the architecture |
| Alternate workflow | - |
| **Stakeholders** | IT Company, system engineer (Mike), financial (and other) managers, administrative department, External accounting company, vice-president |
| **Outcome** | • The company can understand if the organizational setting meets its security requirements  
| • The system is specified independently of the actual service providers  
| • The system architecture ensures trustworthiness and takes into consideration security concerns |
Scenario 40A: Different levels of authentication

Description
Homer uses different services compliant with the Aniketos platform. All of these require a different amount of authentication information of Homer. For example:
- Using an online search engine to look for the closest store to buy some donuts requires the GPS location of Homer.
- Online shopping for a present necessitates a valid credit card or a banking account.
- An e-government service located in Germany depends on the usage of the new German identity card (Personalausweis).

Homer only wants to give a minimal amount of authentication information to an internet service. As an end user he wants to be able to use different methods of authentication providing different identity information depending on the required internet service.

Workflow
1. The end user decides which kind of authentication he wants to provide.
2. The service provider decides if the authentication is sufficient to use its service.

Alternate workflow
2b. The end user offers unnecessary information to a service provider.

Stakeholders
End users, service providers

Outcome
A service provider only receives the necessary authentication information.

Scenario 40B: Providing Unnecessary Information

Description
Mr. Burns is a service provider with limited ethics. He demands valid credit card information to use his service although this information is not necessary. He then harvests credit card information and sells it to people wanting to commit economic fraud.

Stakeholders
Unethical service provider

Outcome
Information can be misused

Assets
Credit card information

Countermeasures
TBD

Scenario 41A: Handling multi-tenant systems

Description
SmartCompany is a service developer and provider. It offers a customer relationship management (CRM) that is based on a complex composition of Web services. The consumed Web services are partly offered by third party service providers and partly developed by SmartCompany. The CRM offered by Smart Company is multi-tenant. That means that the data belong to different customers (f. e. large enterprises). The CRM operates on data of high business value and may also be subject to strict privacy laws. Therefore, SmartCompany has to provide a high level of security.

Workflow
TBD

Stakeholders
Service end users, service providers, network providers

Outcome
A composite service enables the utilization and management of the stored data.
### Scenario 42A: Cargo Dispatching

<table>
<thead>
<tr>
<th>Description</th>
<th>A conveyance named CleverTransport uses a composite service to manage its cargo. One service provides the disposal of the cargo. The routing is provided by another service. A third service calculates the amount of tolls and manages its payment. Based on the routing, a fourth service announces actual traffic jams and alternate routes. The usage of alternate routes influences the other services. Moreover, another service provides to manage the billing and payment of the used services.</th>
</tr>
</thead>
</table>
| Workflow | 1. CleverTransport manages the disposal of its cargo.  
2. Based on this disposal, a service component provides the routing.  
3. Another service component manages the calculation and billing of the necessary tolls.  
4. During the shipping, a service component changes the routing if traffic jam makes it necessary and announces these changes to the affected parties.  
5. CleverTransport receives a billing of another service component, which manages the billing and payment of the affected services and the necessary tolls. |
| Alternate workflow | 4b. The planned routing can be maintained. |
| Stakeholders | End-user, service provider |
| Outcome | Several separate services are bundled to support and to simplify the routing process of an end user. |

### Scenario 42B: Malicious routing service provider

<table>
<thead>
<tr>
<th>Description</th>
<th>A routing service provider could provide a non-optimized route.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholders</td>
<td>End user, service provider</td>
</tr>
<tr>
<td>Outcome</td>
<td>The toll amount increases and/or the gain of the service provider increases.</td>
</tr>
<tr>
<td>Assets</td>
<td>Time and money</td>
</tr>
<tr>
<td>Countermeasures</td>
<td>Create a composite service using two independent routing services</td>
</tr>
</tbody>
</table>

### Scenario 43A: E-Mobility Management

<table>
<thead>
<tr>
<th>Description</th>
<th>The management of the power supply of electrically powered cars can be provided by a composite service. This composite service has to invoke the contractor (of electricity), the common carrier, the e-mobility provider, and the charging station. It also provides methods for authentication of the end user and for billing and account management.</th>
</tr>
</thead>
</table>
| Workflow | 1. An end user authenticates himself and recharges the batteries of its electrically powered car.  
2. A composite service organizes the authentication, the power delivery, and the billing of the invoked participants. |
| Alternate workflow | 2b. The end user can manually compose the required services, and use the Aniketos platform for verification. |
| Stakeholders | End users, service providers, network providers |
| Outcome | An end user is able to recharge the batteries of an electrically powered car and to pay the expenses. A composite service can manage and organized all the invoked parties. |
Scenario 44A: Payback Card\textsuperscript{12}

**Description**
A webshop named CleverShop wants to offer a loyalty program for its customers. A company ShopMore provides a service that combines, in cooperation with BigBank, a loyalty program and a credit card. This composite service has to provide the loyalty program and the credit card functionality as well as the information flow of the customer related data.

**Workflow**
1. An end user purchases different products at CleverShop.
2. He decides to register this order to the loyalty program.
3. He selects a credit card for payment.
4. CleverShop transfers the credit card information and the invoice value to BigBank.
5. CleverShop transfers the following information to ShopMore:
   - a customer identification (f. e. a membership number)
   - the business volume
   - a classification of the purchased good
6. CleverShop processes the order.
7. An evaluation of the purchase behavior is provided by ShopMore to CleverShop for CRM.

**Alternate workflow**
3.b The end user decides that he does not want to use the credit card functionality of the loyalty program. If so, he has to use another credit card or provide additional information to manage the billing.
7b. ShopMore sells the evaluation to another party.

**Outcome**
A composite service supports a loyalty and credit card program.

---

Scenario 44B: Malicious service provider

**Description**
A malicious composite service can store all the gained data including the credit card information to establish and sell a detailed costumer profile.

**Stakeholders**
End users, service providers

**Outcome**
Misuse of data

**Assets**
Credit card information

**Countermeasures**
TBD

\textsuperscript{12} This scenario is inspired by the Use Case Description “Outsourced Branded Credit Card Operation in the Context of Utility Banking”.

ANIKETOS
Scenario 45A: Service Provider reacts to changes in the infrastructure requiring (semi-automatic) changes in the security configuration of an offered service

Description
Ernie is a service developer/provider offering a customer relationship management (CRM) that is based on a complex composition of Web services. The consumed Web services are partly offered by third party service providers and partly developed by him. As the CRM offered by Ernie operates on data that of high business value and may also be subject to strict privacy laws, it has to provide a high level of security. This is especially true, as the system is multi-tenant, i.e., the CRM needs to guarantee that the different customers (large enterprises) do not interfere each other even though they are using (on a technical level) the same backend system and services.

In this scenario, Ernie decides to replace a business analytics service provided by the company “TheObvious” for sales forecasting by a more powerful one offered by the company “IntelligenceAnalytics”. While, compared to the offerings of IntelligenceAnalytics, the service provided by TheObvious offers a less powerful forecasting (i.e., less functionality), its security guarantees are much better. Therefore, Ernie has to add additional security measurements to ensure the same high security standards to his customers (and thus, fulfilling the SLA he has signed with his customers).

Workflow
1. Ernie looks for an alternative “Sales Forecasting” services on the Aniketos marketplace. This search results in a list of different Forecasting services.
2. Ernie decides to compare the security and trust properties of the service he is currently using with the offering of IntelligenceAnalytics using the Aniketos verification module.
3. Ernie is notified by the Aniketos verification module that the service offered by IntelligenceAnalytics does not fulfil all the required security properties. For example, the analyses suggest adding an additional authentication and encryption service to fulfil the basic security requirements. Moreover, the addition of a pseudonymization service is suggested.
4. Ernie build a (secure) composite service that pseudonymizes the input send to the sales forecasting services of IntelligenceAnalytics that sales forecasting services and adds an additional authentication and encryption layer.
5. Ernie analyses the service composition using the Aniketos verification component. This time, the verification is successful.

Alternate workflow
3b. The verification component is not able to suggest a solution that fulfils the required security properties. In this case, Ernie can either stay with her current solution, switch to a different service offering that provides a higher built-in security assurance, provide manually a solution, or weaken the SLA with her customers.

Stakeholders
Service developer, Service composer, Aniketos platform, Aniketos marketplace

Outcome
Ernie is able to replace service while retaining (with respect to the security guarantees) the SLA with his customers.
### Scenario 46A: Billing for usage of a composite service

**Description**
Gwendolyn is a service developer/provider offering a customer relationship management (CRM) that is based on a complex composition of Web services. The consumed Web services are partly offered by third party service providers and partly developed by her. As the CRM offered by Gwendolyn operates on data that of high business value and may also be subject to strict privacy laws, it has to provide a high level of security. This is especially true, as the system is multi-tenant, i.e., the CRM needs to guarantee that the different customers (large enterprises) to not interfere each other even though they are using (on a technical level) the same backend system and services. For her CRM system, Gwendolyn decides to charge a monthly fee on a per user basis (i.e., fixed costs based on the number of accounts). Still, the services her applications consumes (this can also include the hosting) are billed on a per usage base (i.e., number of Web service calls). Thus, she needs to access detailed usage information of both her own customers and the services she consumes.

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Stakeholders</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Service providers, Aniketos Market place, Service aggregator/compositor</td>
<td>Gwendolyn is able to offer a competitively priced service.</td>
</tr>
</tbody>
</table>

### Scenario 47A: Emergency Service Coordination Exercise

**Description**
A local civil defence organisation wishes to test the combined preparedness of several emergency response services (fire, police, ambulance, mountain rescue, etc.) to a major incident. Recent experience has shown that significant delays occur when each organisation wishes to access and/or share data with its peers. Each organisation specifies security policies to protect their data. In the past, these policies were assigned independently, without much consideration for external access. The relevant organisations have agreed to coordinate their policies more, by defining trust relationships and altering their policies to take account of these new relationships. The Aniketos design-time tools are used to identify the static service composition security elements and hence the composition policies that are to be distributed amongst the organisations that are federated.

To test the suitability of these new arrangements, the organisations agree to simulate a major incident: a large landslide occurs, cutting off both a major road and access to a remote village, causing multiple road crashes and chemical spills, and fires caused by gas leaks. Apart from deaths and injuries, traffic needs to be diverted; specialists need to be brought in to deal with chemical contamination and (food and medical) supplies need to be brought to the village, thus affecting multiple agencies.

During the simulation, it is found that policy set complexity has grown to the point that Policy Decision Point (PDP) performance has slowed to an unacceptable level, adversely affecting the operational teams. While the Aniketos platform has successfully defined the design-time service security policies, actual policy performance depends on features outside its scope: how the additional composition policies interact with existing internal policies of each participating organisation, the profile (types and temporal characteristics such as arrival rates) of access requests and the...
The coordinating body identifies policy decision performance as a major risk and decides that a testbed is needed in which to do performance experiments and to provide feedback both to the organisations themselves (e.g., to upgrade their policy infrastructure or refine their policies) and to Aniketos design-time tool users, to make them more aware of performance constraints and hence the performance impact of their design decisions.

**Workflow**

1. Developers use the Aniketos platform to develop the design-time static composition of Civil Defence, Fire and Police information services.
2. At run time, the Civil Defence service connects to the Fire service and Police service.
3. The Civil Defence simulates typical service request patterns, issuing access requests to exercise the PDPs serving the service composition.
4. Performance data is collected and compared against expected service levels.
5. Bottlenecks are identified and triaged.
6. Composition policies and/or PDP infrastructure are improved, as appropriate.

**Alternate workflow**

Coordination of first response to the emergency is hindered by unexpected delays caused by poorly performing access control systems. This means that service compositions do not form in a timely manner, with detrimental effects on the emergency response.

**Stakeholders**

Service providers, Aniketos platform users (end-users and developers).

**Outcome**

Services participating in the composition can obtain and share data in a timely fashion - the performance overhead of the federated access control system is acceptable.

### Scenario 48A: (Re)Composing services for time-critical financial trading

**Description**

A financial trading company (DealMaker) relies on two main sources of market information: FastInfo and SafeInfo. The former has more timely information and simpler access requirements but a lower trustworthiness level than the latter. However, in practice DealMaker finds that FastInfo data is often reliable so its trustworthiness level (e.g., as assigned by other users) may have been set too low. Simplifying the data flows, e.g., by not waiting for corroboration from SafeInfo would make it easier to act more quickly on time-sensitive information, and thereby to make more profit, but with more risk.

So DealMaker has a choice – to change its information service composition strategy or not. The decision will depend on (re-)balancing its Trust and Performance objectives in response to new trust ratings (and hence new trust rankings) becoming available. Since there is some uncertainty regarding the costs and benefits, DealMaker decides to obtain some measurements to help quantify the pros and cons of any change. To estimate the performance benefits, DealMaker uses the Aniketos platform to recompose the services and deploys the recomposed services on a testbed in which DealMaker can measure the expected performance benefits of the new security compositions. DealMaker can then enter
these predicted performance improvements into their financial models to estimate the revised financial returns, for comparison with the additional trading risks incurred. This scenario can be generalised to consider a more typical $m$ information providers with $m > 2$, with the objective being to choose a subset of $n$ information providers with $0 < n < m$ while maximising Profit (equivalently, minimising Cost) and minimising Risk. Aniketos can focus on Trust and Performance, as a proxy for Risk and Return, respectively.

**Workflow**

1. The Aniketos platform composes at least two alternative financial information services.
2. Based on external information and/or changed business objectives, the Trust information on which the Aniketos platform based its service composition changes.
3. The Aniketos platform run time tools propose alternative service compositions, but there is still some uncertainty in respect of the revised Trust measures.
4. The performance characteristics of the service recomposition proposals are estimated by experimentation.
5. The “best” service recomposition proposal is chosen, based either on expected revenue or costs.

**Alternate workflow**

DealMaker continues with a sub-optimal service composition, reducing its ability to make profitable trading decisions because of delays in obtaining critical information.

**Stakeholders**

Service users, Aniketos platform.

**Outcome**

Alternative service recompositions are proposed by (re)balancing security, performance and risk objectives.

### Scenario 48B: Disaffected employee sabotages the policy infrastructure

**Description**

One of the information services, FormerlyTrustedInfo, favoured by DealMaker lays off Alice, a key member of its network operations staff. Having insider information, Alice sabotages its policy infrastructure, e.g., by reducing available computing or network resources, or changing policies. The changes are subtle so the sabotage is not recognised immediately. However, the Aniketos platform recognises that there is a performance problem with the FormerlyTrustedInfo service and modifies its trustworthiness level accordingly. The Aniketos platform alerts the FormerlyTrustedInfo service provider and shares its performance model with FormerlyTrustedInfo to assist the latter to fix the problem. In the meantime, Aniketos switches to LowerTrustInfo service (because of the new trust rankings) with a view to reinstating FormerlyTrustedInfo when its problems are fixed and the trust rankings are updated.

**Stakeholders**

Aniketos platform user, (formerly) trustworthy service provider.

**Outcome**

Service composition quality is reduced, but only temporarily

**Assets**

Policy infrastructure

**Countermeasures**

Aniketos monitors effect of composition security on composed service performance
Scenario 49A: Unified communications (such as group chat) within virtual teams

Description

Increasingly, simply-connected 1:1 seller-buyer relationships (value chains) are being replaced with highly-connected value networks in which parts of different enterprises need to work together as peers, perhaps temporarily, to achieve business objectives. Consequently, virtual teams are formed comprising Buyer, Seller, Intermediary1, Intermediary2, etc., and are drawn from several enterprises participating in a federation. There is a complex web of access control rules governing communication within such virtual teams. The Aniketos platform can take Trust and Contract information at design-time and use this information to develop federation-level policies governing the underlying federation relationships. Policy authors at each of the enterprises participating in the federation need to confirm that these policies are consistent with their local policies.

Access control decisions cannot be made just at the start of a communication session when access control conceptual models such as Chinese Walls are needed, or a new media type (such as a spreadsheet) is introduced.

Note that orchestration policies have $O(n^2)$ complexity, where $n$ is the number of participating enterprises in the federation and that remote access control decisions are generally more expensive (e.g., slower) than local access control decisions. Thus foreknowledge of performance characteristics at federation (composition) design time (when the Aniketos tools create the federation policies) is highly valuable. If the distributed access control system has already been “calibrated” in the sense that a well-qualified performance model exists, the performance impact of the federation-level policies can be estimated. The Aniketos platform user is able to make better design time decisions and consequently, fewer performance problems would be expected when the policies are deployed.

Workflow

1. At least two independent entities wish to participate in a federation.
2. The business needs, responsibilities and dependencies within the federation are codified in a set of contracts.
3. Organisational, Trust and Contract metadata is shared between the federation members.
4. Each entity provides privacy-preserving/mock implementations of their local access control infrastructure to facilitate the creation of federation-wide performance models.
5. The Aniketos platform is used to create federation level policies, to enable federation members to compose communication services to enable smooth running of the federation.
6. Administrators at each entity enable access subject to the federation-level policies.

Alternate workflow

At least one entity participating in a (business) federation is unable to satisfy its contractual membership terms, because its communication services are found to be unable to participate in a timely manner in approved service compositions. This weakens the overall cohesiveness of the group.

Stakeholders

Federation members, Aniketos platform.

Outcome

Communication services within the business federation are composed in such a way as to support beneficial communication while respecting access controls.
Scenario 49B: DDOS attack occurs on one of the federation partners

**Description**
One of the federation partners, PaymentProvider, suffers a DDOS attack by web activists and consequently its access control system over the web (port 80, port 443, etc.) and similar entry points is brought down. However, some communication modes continue to function normally. As part of its business continuity plan, PaymentProvider provides a stub infrastructure which the Aniketos platform can use to test whether multi-party communication services can be maintained, in respect of security and performance. The Aniketos platform finds a new service composition that enables basic communications to be maintained.

**Stakeholders**
Aniketos platform user, federation partners.

**Outcome**
Service composition lose functionality

**Assets**
Policy infrastructure

**Countermeasures**
Aniketos platform uses a mock federation partner instance to recompose services in response to unexpected events in the business environment.

---

Scenario 50A: Reputation Based Trustworthiness

**Description**
A service provider TrustedMedia aggregates services from a number of other service providers in communications and media sectors including TV, phone, online games, news and entertainment to fulfil customer requests. For each type of service providing the same (or similar) functionality, TrustedMedia has to select from a collection of services from a number of service providers. In order, to help its own business reputation and reuse of its services, TrustedMedia needs to selects the services that meet the highest expectations. Hence, it decided to rank services in each service type (e.g. A, B) based on their trustworthiness into A1, A2, A3, B1, B2, B3, etc. TrustedMedia then selects the services with best trustworthiness. Reputation of the services from TrustMedia’s own experience (e.g. QoS) and user feedback (QoE) is used to ensure that the chosen services are trustworthy and hence available, reliable, secure, etc. During the execution of the composite service, the trustworthiness of one of the component services deteriorates caused by annoying delay in the component service response. Aniketos platform which monitors the reputation of the services decides that the deteriorated component service should be replaced with another service that was the 2nd in the rank of services with the same functionality.

**Workflow**
1. In order to select the best service, TrustedMedia needs to rank component services per functionality based on the reputation of each service.
2. The reputation is determined using a combination of monitored QoS and user experiences.
3. TrustedMedia selects component services in the 1st rank to compose a new service.
4. During the running of composite services the trustworthiness of one of the component services may be reduced by Aniketos platform.
5. The composed service is adapted by replacing the now more trusted component service in 2nd rank.

**Alternate workflow**
1. Aniketos platform could not detect services with low reputation or changes in trust and reputation in the runtime.
2. Composite service total/partial failure and reduced trust in composite service, providers and Aniketos platform.

**Stakeholders**
Service providers (including composer), services, users, Aniketos
Scenario 50B: Trustworthiness Management Threats

**Description**
TrustedMedia relies on Aniketos platform to manage the trustworthiness of component and composite services through reputation. Many of the user ratings collected by the Aniketos platform are dishonest or malicious aiming to improve/damage the trustworthiness of a target component service and consequently increase/decrease its usage and profit. However, Aniketos platform is already aware of these threats and provides mechanisms to filter out bad ratings. As a result, threats have minimal effect on the monitoring and computation of service trustworthiness in Aniketos platform.

**Workflow**
1. A number of malicious users collusively or separately decide to improve/damage the Aniketos platform trust in a service to increase/decrease profit.
2. The users rate the service according to their plans.
3. Aniketos platform detects the bad ratings and ignores them when computing the trustworthiness of the service.

**Alternate workflow**
1. Malicious users identify the vulnerability in Aniketos platform and manage to control service reputation according to their goals.
2. Users of Aniketos platform e.g. service composers, think that Aniketos platform is not reliable in managing trustworthiness.

**Stakeholders**
Aniketos platform, users, services, service providers.

**Outcome**
Aniketos platform is resilient against attacks by malicious user ratings.

**Assets**
Trustworthiness level

**Countermeasures**
Aniketos identifies dishonest ratings and disregards them

Scenario 51A: Aniketos Training and tutorials

**Description**
Björn is the lead developer of a small software company. He has read about the security advantages offered by Aniketos-enabled services and wants to learn more. First, he visits the Aniketos website and discovers the Aniketos User’s Guide along with the Aniketos Developer’s Guide; those two documents are always up-to-date, and contain all the information he needs to start composing secure services. The User’s Guide tells him about the end-user benefits of Aniketos, while the Developer’s Guide contains detailed information about each Aniketos component, their APIs and other information – such as example code – necessary to start development. API documentation is generated with an automated tool to ensure that it is always up to date.

Afterwards, Björn can attend (or send his developers to) classroom trainings or peruse distance learning material.

- **Distance learning**: Björn looks through Aniketos training and tutorial material available online, noting that all material is properly labeled according to the target audience, role, area of interest, and required level of expertise and that all courses are built around the same scenarios and themes for consistency. Finally, he picks several distance learning courses based on his own preferences; he watches a webinar and the accompanying slides, and downloads a few simple source code fragments to practice with. While perusing one of the
courses, he notes that the example code’s interaction with Aniketos is not explained too clearly; he uses the online feedback form to notify the training’s creators of this. Eventually, the training will be updated based on his comments.

- Classroom training: Björn sends Anette – one of his developers – for one of the Aniketos classroom trainings. She attends the training and listens to the trainer giving an exposition on relevant topics; she also receives a printed handout containing the presented material. During the course of the training seminar, theoretical sections alternate with hands-on exercises. Sometimes Anette is prompted to start up a virtual machine on her own laptop and solve an exercise. She finds that the exercise environment is set up beforehand, so she needs to concentrate only on the interesting and challenging parts of the assignment. Whenever she gets stuck in the middle of an exercise, she turns to the trainer for assistance. Likewise, if something is unclear during the presentation, she requests clarification, which the trainer is happy to provide. On course completion she is asked to fill out an assessment form. She rates the quality of the training material, trainer performance, learning environment and learning infrastructure. Upon leaving, she takes away a copy of the source code examples.

2. The developer chooses distance learning material appropriate to his own expertise and area of interest.  
3. The material consists of non-interactive presentations (webinars), slide sets, and exercises along with example programs.  
4. After completing the tutorial session, the developer evaluates its usefulness and provides feedback via an online feedback form.  
5. Distance learning material is evaluated and updated based on feedback. |
| **Alternate workflow** | 1. The developer signs up for an Aniketos classroom training.  
2. The training session consists of theoretical sections along with hands-on exercises.  
3. The developer uses a virtual machine environment – distributed to all training participants – to solve the exercises.  
4. After the training session, all participants fill out an assessment form.  
5. The training session is evaluated and updated based on the results of the assessment forms. |

| **Stakeholders** | Service developer and composer |
| **Outcome** | Björn and his developers become proficient in using Aniketos to compose secure services, and developing secure services in general. |
Scenario density
The following overview gives an approximate "density" on how the of the various scenarios relate to the different features of Aniketos. We saw this as an exercise in order to determine the coverage of the scenarios, and subsequently the project itself. This density is given as different font sizes in a sort of directory-tree like list. The scenario identifiers are listed after each feature. Some of the features have zero scenarios because they were defined before we started with the scenarios based on common concepts from dynamic service composition with respect to security and trust.

<table>
<thead>
<tr>
<th>Font scaling explanation</th>
<th>Number of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Italics</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>11 Bold Italics</td>
<td>2</td>
</tr>
<tr>
<td>12 Bold</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>14 Bold</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>16 Bold</td>
<td>7</td>
</tr>
<tr>
<td>18</td>
<td>8</td>
</tr>
</tbody>
</table>

**Design Time** (2) 12, 51
Design-time service specification (1) 38

Modelling of security and trust props of a composite service (6) 10, 24, 28, 30, 38, 49
Define security goals, props, policies, regulation, requirements (5) 1, 10, 13, 30, 41
  Define trust level (3) 17, 26, 50
    Threat binding (1) 29
    Map security requirements to trust contracts (1) 28
    Decompose contracts according to service decomposition (1) 28
    Education of service composers (threat awareness) (1) 10
    Defining offered security/trust props for a service component (1) 25

Development tool integration (1) 28

Service planning (0)
  Make use of a template (2) 23, 24
  Make use of a workflow specification (0)
  AI planning (0)

Service discovery (1) 1

Design-time service composition (0)
  Static binding (0) 42
  Dynamic binding (1) 25
  Create service bus (1) 22

Service validation (design- and run-time) (6) 17, 23, 28, 29, 32, 33

Check contract fulfilments (calculate/evaluate…) (5) 24, 26, 30, 34, 48
  Trust levels (3) 5, 25, 37
  Security properties (3) 7, 10, 35
<table>
<thead>
<tr>
<th><strong>Runtime</strong></th>
<th>(2) 8, 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment at service provider</td>
<td>(1) 19</td>
</tr>
<tr>
<td>Establish security contract</td>
<td>(3) 24, 26, 41</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>End-user service experience</strong></th>
<th>(0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inform end-user</td>
<td>(7) 2, 11, 15, 25, 30, 32, 34</td>
</tr>
<tr>
<td>End-user involvement</td>
<td>(5) 21, 30, 40, 43, 44</td>
</tr>
<tr>
<td>End-user assurance</td>
<td>(4) 7, 20, 30, 31</td>
</tr>
<tr>
<td>Register subjective/cognitive trust</td>
<td>(0)</td>
</tr>
<tr>
<td>Simplify service invocation</td>
<td>(1) 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>End-user context/environment</strong></th>
<th>(2) 18, 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime service recomposition</td>
<td>(8) 2, 15, 16, 22, 23, 31, 47, 48</td>
</tr>
<tr>
<td>Runtime service planning</td>
<td>(0)</td>
</tr>
<tr>
<td>Runtime service discovery</td>
<td>(1) 30</td>
</tr>
<tr>
<td>Dynamic binding</td>
<td>(3) 11, 25, 30</td>
</tr>
<tr>
<td>Make use of service bus</td>
<td>(1) 22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Runtime service adaptation</strong></th>
<th>(5) 26, 27, 32, 33, 45</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information to service end-users</td>
<td>(0)</td>
</tr>
<tr>
<td>Service monitoring</td>
<td>(0)</td>
</tr>
<tr>
<td>Detect change</td>
<td>(1) 22</td>
</tr>
<tr>
<td>Detect vulnerability or attack (threat)</td>
<td>(2) 14, 27</td>
</tr>
<tr>
<td>Detect contract violations</td>
<td>(1) 11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Threat notification</strong></th>
<th>(2) 19, 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send threat notification</td>
<td>(1) 27</td>
</tr>
<tr>
<td>Mediate threat notification</td>
<td>(1) 27</td>
</tr>
<tr>
<td>Receive threat notification</td>
<td>(0)</td>
</tr>
<tr>
<td>Subscribe a service to notifications</td>
<td>(1) 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Vulnerability detected</strong></th>
<th>(3) 14, 15, 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notify service consumers</td>
<td>(0)</td>
</tr>
<tr>
<td>Lower trust level</td>
<td>(3) 16, 26, 50</td>
</tr>
<tr>
<td>Change security properties</td>
<td>(0)</td>
</tr>
</tbody>
</table>

| **Business opportunities** | (2) 36, 46 |