

 <p>Secure and Trustworthy Composite Services</p>  <p>Seventh Framework Programme: Call FP7-ICT-2009-5 Priority 1.4 Trustworthy ICT Integrated Project</p>	Deliverable ID:	Preparation date:
	D5.1	01 September 2011
	Milestone: Released	
	Title:	
	<p align="center">ANIKETOS platform design and platform basis implementation</p>	
Editor/Lead beneficiary (name/partner):		Vasilis Tountopoulos/ATC
Internally reviewed by (name/partner):		Edith Félix/Thales, Achim Brucker /SAP
Approved by:		Executive board
<p>Abstract:</p> <p>Aniketos is about establishing and maintaining trustworthiness and secure behaviour in a constantly changing service environment. The project aligns existing and develops new technologies, 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.</p> <p>This deliverable presents the specifications and provides a baseline implementation of the Aniketos platform, which facilitates as a reference to the envisaged functionalities, which have to be supported, as well as guidance to the technical partners on the way to plug their individual components into the platform. The deliverable exploits the work performed in all technical workpackages to provide the design aspects needed for the interfaces and the communication of the Aniketos platform components with the environment components both at design-time and runtime.</p>		
Dissemination level		
PU	Public	<input checked="" type="checkbox"/>
CO	Confidential, only for members of the consortium (including Commission Services)	

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

Aniketos consortium

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



SINTEF ICT (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-20009 Donostia - San Sebastian
Gipuzkoa (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: Pucci Paolo
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



UNIVERSITÀ DEGLI STUDI
DI TRENTO

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.dblue.it

Contact: Valentino Meduri
valentino.meduri@dblue.it



Wind Telecomunicazioni S.P.A.
(WIND)
00148 Roma, Italy
www.wind.it

Contact: Rita Spada
MariaRita.Spada@mail.wind.it



CITY OF ATHENS - IT COMPANY

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	vii
Executive summary	1
1 Introduction	3
1.1 Aniketos motivation and background	3
1.2 Summary	3
1.3 Structure of this document	4
1.4 Relationships with other deliverables	4
1.5 Contributors	5
1.6 Acronyms and abbreviations.....	5
1.7 Change log	6
2 Aniketos Platform Overview	7
2.1 Overview of the Aniketos Platform Objectives	7
2.2 System level Requirements	8
2.2.1 Functional Requirements	8
2.2.2 Non-functional Requirements.....	9
2.3 The Aniketos Architecture	10
2.4 Layered-based architectural design.....	11
2.5 Development Methodology	14
3 Analysis of technological background	16
3.1 Introduction.....	16
3.2 Development of the Aniketos Platform	16
3.2.1 Eclipse Plug-in Mechanism	16
3.3 Web service composition standard languages	18
3.3.1 Business Process Modelling Notation	18
3.3.2 Business Process Execution Language	20
3.4 Security Framework Technologies	23
3.4.1 Introduction	23
3.4.2 Overview of security specifications.....	24
3.4.3 OpenID and OAuth.....	28
3.4.4 SAML2 & XACML.....	30
3.5 Communication Technologies	31
3.5.1 Enterprise Service Bus.....	31
3.5.2 SOAP Web Services	34
3.5.3 REST Web Services	34
3.6 Application Server Technologies.....	35
3.6.1 OSGi	35
3.7 Persistency and Information Storage Technologies	38
3.7.1 MySQL Database Server	38
3.7.2 PostgreSQL Database System	39
3.7.3 MongoDB	39
3.7.4 Conclusion.....	39
4 Description of the Aniketos components	40
4.1 Aniketos Platform components	40
4.1.1 Socio-technical Security Modelling Tool.....	40
4.1.2 Model Transformation Module	41

4.1.3	Trustworthiness Component.....	44
4.1.4	Verification module.....	47
4.1.5	Security Property Determination Module.....	51
4.1.6	Secure Composition Planner Module.....	55
4.1.7	Security Policy Monitoring Module.....	58
4.1.8	Threat Response Recommendation Module.....	59
4.1.9	Service Threat Monitoring Module.....	61
4.1.10	Notification module.....	66
4.1.11	Community Support Module.....	70
4.1.12	Threat Repository Module.....	72
4.1.13	Marketplace.....	76
4.1.14	Training Material Module.....	78
4.2	Environment components.....	79
4.2.1	Service Composition Framework.....	79
4.2.2	Service Runtime Environment.....	81
4.2.3	Identity Management Service.....	91
4.3	Summary of interfaces.....	93
5	Design of the Aniketos Marketplace.....	103
6	Aniketos Baseline Implementation.....	106
6.1	Aniketos Platform at design-time.....	106
6.1.1	Design-time related Aniketos Components.....	106
6.1.2	Template Eclipse Plug-in.....	107
6.2	Runtime Platform.....	108
6.2.1	Template OSGi Bundle.....	110
6.3	Implementation of the Aniketos Platform and Environment Components.....	111
7	Conclusions.....	114
References	115
8	Annexes.....	117
8.1	Runtime Template Component.....	117
8.1.1	Pre-requisites.....	117
8.1.2	Downloading.....	117
8.1.3	Running.....	119
8.1.4	Service Implementation.....	121
8.2	Template Eclipse Plugin and OSGI Component(s).....	126
8.2.1	Pre-requisites.....	127
8.2.2	Getting started.....	127
8.2.3	Installing BPMN2.....	127
8.2.4	Obtaining our (eu.aniketos.wp3.dummy.*) plugins and bundles.....	128
8.2.5	Executing the plugin and bundles.....	130

List of figures

Figure 1: Goal: establish and maintain security and trustworthiness in composite services	3
Figure 2: The objectives of the Aniketos project	7
Figure 3: Overview of the Aniketos Platform	10
Figure 4: Components in the Aniketos platform and in the environment	12
Figure 5: A layer-based conceptual representation of the Aniketos architectural design	13
Figure 6: The basic steps for delivering the Aniketos platform, based on the agile software development methodology	14
Figure 7: Continuous interaction of WP5 with the development and assessment phases	15
Figure 8: The Eclipse plug-in architecture (please refer to [14])	17
Figure 9: The Eclipse plug-in layers (please refer to [14]).....	17
Figure 10: Service composition: orchestration approach (please refer to [15])	18
Figure 11: BPMN Diagram for travel booking process (please refer to [18])	19
Figure 12: Example of a BPEL process (please refer to [20]).....	22
Figure 13: Technologies and specifications for Web Service – based platforms: a) First-Generation, b) Second Generation (please refer to [24]).....	24
Figure 14: SSL protection across multiple services (please refer to [25])	27
Figure 15: WS-Security Framework concept (please refer to [25])	28
Figure 16: The OAuth Authorisation process steps (please refer to [31]).....	30
Figure 17: An example topology for ESB (please refer to [35])	31
Figure 18: The Mule ESB Architecture (please refer to [36]).....	33
Figure 19: The OSGi Architecture (please refer to [40])	36
Figure 20: The logical architecture of PRRS	84
Figure 21: The interfaces of the Service Monitoring Module	86
Figure 22: The runtime interfaces of MUSIC towards Aniketos	90
Figure 23: The Architecture of the Aniketos Marketplace.....	103
Figure 24: Register Service User Interface	104
Figure 25: Searching for Services	105
Figure 26: Typical processes related to design-time service composition (please refer to D1.2 [1]) .	106
Figure 27: The design-time related Components	107
Figure 28: General processes related to runtime reaction to changes and monitoring (please refer to D1.2 [1]).....	109
Figure 29: The Runtime Platform	110
Figure 30: Import project dialogue box.....	129
Figure 31: SVN Repository Information.....	129
Figure 32: Eclipse "Check out as" project wizard.....	130
Figure 33: Check out project selection.....	130
Figure 34: New Launch Configuration and Plugin tab	131
Figure 35: Eclipse Runtime: Plugin configuration tab	131
Figure 36: Show the View panel created by scp.client	132
Figure 37: The user interface of the template in runtime Eclipse	133
Figure 38: Output of the template shown in console of development Eclipse	133

List of tables

Table 1: Categorisation of functional requirements for the development of the Aniketos Platform.....	8
Table 2: Non-functional requirements for the development of the Aniketos Platform	9
Table 3: Web services standards and XML security specifications	25
Table 4: WS-Security Framework Standards ([25]).....	28
Table 5: Open source ESB implementations and their corresponding license	32

Table 6: Relation of D5.1 implementations to D1.2 specifications.....	94
Table 7: Status overview of the Aniketos platform and environment components implementation...	111

Executive summary

The main objective of the Aniketos project is to establish and maintain security and trustworthiness in composite services. In this context, this deliverable focuses on the design aspects and the baseline implementation of the Aniketos platform, which provides the reference point for the development activities of the next period.

The document identifies the layered view on the Aniketos architecture, which presents the currently adopted components to support the functionalities envisaged for both the Aniketos platform and the environment. In that sense, it specifies the interfaces of the components to security and trustworthiness attributes, when designing, implementing, deploying and running composite services, while emphasises on the design of the Aniketos Marketplace, which is being developed to store the security specifications of the Aniketos compliant services and service compositions.

The deliverable elaborates on the methodological approach to develop the Aniketos components and release the integrated Aniketos platform prototypes in two phases, specifically on M22 and M39. It also presents the standardised approaches that have been adopted in Aniketos to provide design-time and runtime support to secure service composition and analyses the security framework technologies, which mainly focus on the support of security attributed in Web-based serviced-oriented systems.

The document is concluded with the principles of the baseline implementation of the Aniketos platform at design-time and runtime, while summarising the guidelines for developing the Aniketos platform and environment components, as a n attempt to provide a unified approach towards the integrated view of the Aniketos platform support to design-time and runtime secure service composition and deployment.

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 may be composed of multiple services from many different providers, and the end user may have little knowledge in the way of guarantee that a particular service or service supplier will actually offer the security claimed.

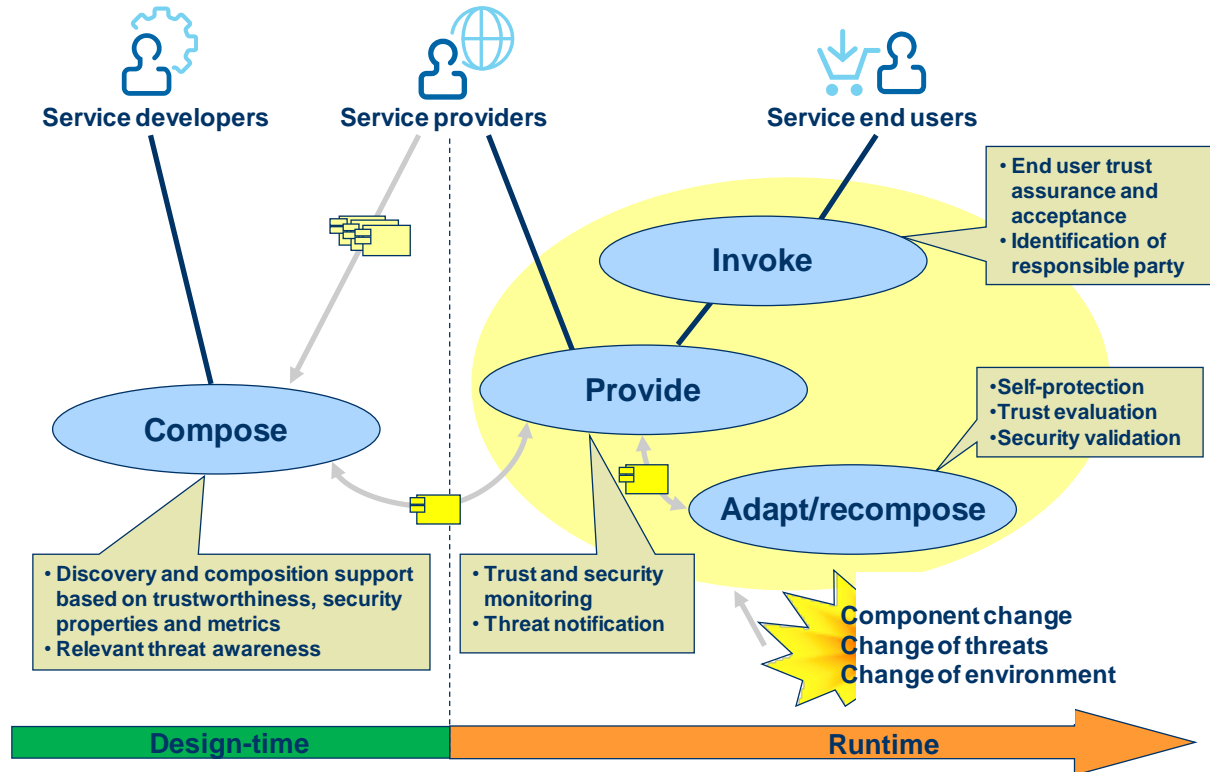


Figure 1: Goal: establish and maintain security and trustworthiness in composite services

As depicted in Figure 1, Aniketos is about establishing and maintaining trustworthiness and secure behaviour in a constantly changing service environment. The project aligns existing and develops new technology, methods, tools and security services that support the design-time creation and run-time dynamic behaviour of composite services, addressing service developers, service providers and service end users.

Aniketos provides methods for analysing, solving, and sharing information on how new threats and vulnerabilities can be mitigated. The project constructs a platform for creating and maintaining secure and trusted composite services. Specifications, best practices, standards and certification work related to security and trust of composite services are promoted for inclusion in European reference architectures. Our approach to achieving trustworthiness and security of adaptive services takes account of socio-technical aspects as well as basic technical issues.

1.2 Summary

This deliverable covers the need for a baseline prototypical implementation of the Aniketos platform early in advance, in order to guide the development activities of the project and provide a common background for the Consortium on the envisaged technologies that are necessary to build such a platform.

The deliverable takes advantage of the work performed in all technical work-packages and exploits the project design aspects needed to make the specifications for the interfaces and the communication of the Aniketos platform components with the environment components both at design-time and runtime. The baseline implementation facilitates as a reference to the envisaged functionalities, which have to be supported, as well as guidance to the technical partners on the way to plug their individual components into the platform.

The deliverable provides the specifications of all the developed interfaces, based on which the foreseen interactions between the Aniketos components are established, as well as the way to interoperate with external platforms. The final outcome of this prototype is a baseline implementation conforming to common standards and implementation approach, which will be exploited by the Consortium to provide their work towards delivering an integrated platform close to project end.

1.3 Structure of this document

The document is structured as follows:

- Section 2 makes an overview of the Aniketos technological areas and research objectives, which guide the development of the Aniketos platform, and summarises the system level functional and non-functional requirements. Then, Section 2 refers to the Aniketos envisaged functionalities and presents the architectural design aspects of the actual Aniketos platform implementation. Moreover, this section elaborates on the methodological approach to develop the Aniketos components and release the integrated Aniketos platform prototypes.
- Section 3 makes an overview of the existing state-of-the-art in the areas reflecting the Aniketos developments. In that respect, it presents the standardised approaches that have been adopted in Aniketos to provide design-time and runtime support to secure service composition. It analyses the security framework technologies, which mainly focus on the support of security attributed in Web-based serviced-oriented systems. Section 3, also, presents the communication technologies that are adopted to support the interaction between the distributed Aniketos components, facilitating the interaction with external platforms as well.
- Section 4 summarises the Aniketos platform and environment components, focusing on the description of the interfaces and the methods, which have been designed and have been implemented, as first prototypes in a baseline form, to offer a common realisation background on how the Aniketos platform can work in real life scenarios, from an integrated perspective.
- Section 5 describes the design of the Aniketos Marketplace, which is being developed to store the security specifications of the Aniketos compliant services and service compositions.
- Section 6 elaborates on the Aniketos baseline implementation; it distinguishes between the Design Time Platform implementation, which facilitates the required functionalities for discovering existing services, selecting service components, validating services, establishing contracts, assembling service compositions and deploying services, and the Runtime Platform implementation, which facilitates the required functionalities for monitoring service execution, validating services, reacting to changes in service provisioning, recomposing services and reconfiguring contracts.
- Section 7 summarises the scope of this deliverable and provides links to future work in WP5
- Section 8 summarises, as Annex to this document, the guidelines for developing the Aniketos components towards a unified approach to facilitate the integrated view of the Aniketos platform support to design-time and runtime secure service composition and deployment.

1.4 Relationships with other deliverables

The baseline implementation of the Aniketos platform, which is presented in this document, reflects the work that has been performed in the first year of the project in all the technical work-packages (WPs). The reference point for this document consists of the Aniketos Deliverable “D1.2: First

Aniketos architecture and requirements specification”, which presents the architectural aspects of the whole project. In principle, D5.1 is affected by all the Aniketos deliverables, which are active in this period in WP1-WP4, but we mainly refer to “D2.1: Models and methodologies for embedding and monitoring trust in services”, “D3.1: Design-time support techniques for secure composition and adaptation” and “D4.1: Methods and design for the response to changes and threats”, which activities ran in parallel to D5.1.

1.5 Contributors

ATC has shared the major effort in the preparation of this document, but all project partners with effort in WP5 or mainly involved in WP1-WP4 (SINTEF, ESI, CNR, LJMU, THALES, ELSAG, ATOS, SAP and ITALTEL) have significantly contributed mainly to the specification of the components’ interfaces and the baseline development of the Aniketos platform and environment components, as well as the preparation of the design-time and runtime templates. It should be noted that interactions and discussions with the rest of the Consortium partners was necessary to align the activities in this deliverable with the relevant ones, performed in the technical WPs. To this end, the partners, with effort allocated in the target WPs, acted as the contact points.

1.6 Acronyms and abbreviations

ASD	Agile Software Development	S&D	Security and Dependability
CDDL	Common Development and Distribution License	SAML	Security Assertion Markup Language
CMM	Contract Manager Module	SCPM	Secure Composition Planner Module
CSV M	Composition Security Validation Module	SDK	Software Development Kit
CVS	Concurrent Versions System	SOA	Service-Oriented Architecture
ESB	Enterprise Service Bus	SOAP	Simple Object Access Protocol
GPL	GNU General Public License	SPDM	Security Property Determination Module
IDE	Integrated Development Environment	SRF	Serenity Runtime Framework
JDT	Java Development Tools	SSL	Secure Sockets Layer
OAuth	Open Authorization	UDDI	Universal Description, Discovery, and Integration
ORM	Object Relational Mapping	W3C	World Wide Web Consortium
PDE	Plug-in Development Environment	WSDL	Web Services Description Language
PDP	Policy Decision Point	XACML	Extensible Access Control Markup Language
PEP	Policy Enforcement Point	XKMS	XML Key Management
PRRS	Platform for Runtime Reconfigurability of Security	XML	Extensible Mark-up Language
PVM	Property Verification Module	XRI	Extensible Resource Identifier
RCP	Rich Client Platform		
REST	Representational State Transfer		

1.7 Change log

No change log entries.

2 Aniketos Platform Overview

2.1 Overview of the Aniketos Platform Objectives

Following the main objective of the Aniketos project to “establish and maintain security and trustworthiness in composite services” as shown in Figure 1, this section provides in detail the main research and technological areas, which the Aniketos platform functionalities should span across.

The Aniketos platform stays upon the Environment systems, as it was described in D1.2 [1]. Service composition, service runtime and service storage have been identified as the principal groups of components, which facilitate the environment functionalities. Upon them, the Aniketos platform stands as the overlay layer to support security and trustworthiness attributes, when designing, implementing, deploying and running composite services. In that sense, the Aniketos objectives are visualised in Figure 2.

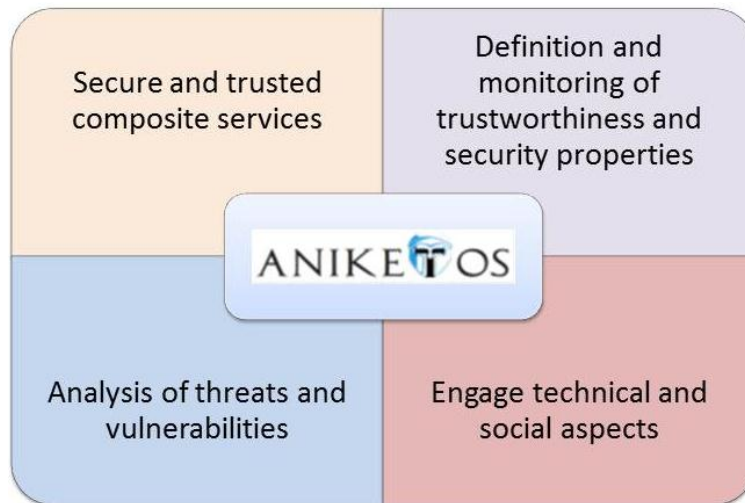


Figure 2: The objectives of the Aniketos project

In more details, the project delivers the Aniketos platform, which targets to advance the state-of-the-art in the area of service composition by creating and maintaining *secure and trusted composite services*. Through the appropriate specification of methods and development of tools and services, the Aniketos platform aims to support the whole service life cycle in service engineering, ranging from service implementation, discovery and composition to service management, adaptation and reconfiguration.

As future internet services can be dynamically composed or evolved, the Aniketos platform defines trust models and security policies, through which the interested stakeholders can define, validate and monitor *trustworthiness* and *security properties*. These properties can be used to overcome the shortcomings in service engineering when dealing with security violation issues.

Security violations can occur when system and services are vulnerable to intruders, which may affect the set security standards and the quality of experience received by the users. Towards this direction, the Aniketos platform tries to address potential loss on service availability and end user trust by efficiently analysing, solving and sharing information on how *new threats and vulnerabilities* can affect service composition and can be mitigated, so that the composed services can be automatically reconfigured to adapt the new conditions.

On top of that, the Aniketos platform adds a *socio-technical perspective* to the way that security and trustworthiness are addressed in service engineering. Since service and service-based system target to highly business-oriented environments, the respective processes, which are being supported through the deployment of the appropriate composite services, are governed from both technical and social aspects, which should be tackled together once security and trust are considered.

In parallel to the above main axis of the Aniketos platform objectives, the relevant design and implementation activities target to the promotion and contribution to open European reference architectures and specifications, best practices, standards and certification work related to security and trust of composite services. Through the instantiation and deployment of the platform in real life applications, the project aims to demonstrate and evaluate the practical usability of the platform and investigate on real end user needs, acceptance and trust of new composite services in realistic case studies representing highly relevant services critical and typical to the future European infrastructure.

2.2 System level Requirements

Based on the work performed in D1.2 [1] and the identification of system level functional and non-functional requirements, this section presents the generic functionalities, which should lead the work in WP5.

2.2.1 Functional Requirements

In Section 3 of D1.2 [1], a number of user level functional requirements have been identified for the project life cycle. This list has been assessed to address and guide the work performed in the technical WPs (WP1-WP5). On this Section, only those requirements which refer to the integrated platform are considered, which are mapped to general functionality categories, as shown Table 1.

Table 1: Categorisation of functional requirements for the development of the Aniketos Platform

Category	Short Description	Relevant Requirements	
Resource Processing	Analysis and process of services, including security and trust	S01A-3	Trustworthiness ranking
		S22A-5	Marketplace allows service providers to renegotiate trust
		S30B-2	Aniketos infrastructure
		S31A-5	Dynamic management of circle of trust
		S32A-1	Analyse federated services
		S36A-1	Pay for trust determination
		S14A-4	Rank services
Resource Sharing	Notification and awareness of service attributes	S36A-4	Advertisement on the platform
		S50B-1	Distributed manipulation of trust
		S01A-1	Publish trust properties
		S04A-2	Upload specifications
		S04A-3	Usage conditions
		S05A-2	Trust awareness
		S11B-2	Event logging
User Interaction	Interaction with the end user	S11B-1	End user choosing
		S18A-1	End user trust threshold
		S20A-1	Interface for checking trustworthiness
		S20A-2	Interface for informing about used components
Community and Training Support	Administration and support for exchanging knowledge about Aniketos developments	S02B-1	Guidelines to composition with end user involvement
		S04A-1	Developer centre
		S12A-1	Aniketos overview
		S12A-2	Aniketos infrastructure
		S12A-3	Aniketos supportive services

Category	Short Description	Relevant Requirements	
		S28A-2	Methodology for using Aniketos
		S36A-2	Register for Aniketos support
		S38A-1	Support service developers
Quality Metrics	Reflecting non-functional platform characteristics	S23B-1	Predicted computation needs
		S23B-2	Required response time for selecting algorithm

2.2.2 Non-functional Requirements

This section analyses the non-functional requirements for developing the Aniketos platform components (no special focus is given to address non-functional requirements through the implementation of the environment components).

The Aniketos platform should satisfy a set of non-functional requirements, which will ensure the normal operation of the system and the provision of a proper environment for the desired system level functionalities, which were analysed in the previous section. The non-functional requirements are depicted on Table 2.

Table 2: Non-functional requirements for the development of the Aniketos Platform

ID	Non-functional Requirement	Description
NF1.	Persistence	The system should provide back-up options for the contents of the services description and their associated profiles. All the repositories should be based on implementation able to tackle persistence problems, which ensure that data is maintained in case of a node failure.
NF2.	Portability	The developed modules should target to be adapted to multiple end user devices, thus the relevant technologies to be as device agnostic as possible.
NF3.	Performance	This requirement has to do with QoS characteristics, such as the response time of an Aniketos compliant service request or the needed resource utilisation schema to support the Aniketos functionalities
NF4.	Scalability/Expandability	The platform should be able to handle the increasing size of services and the simultaneous users accessing the supported functionalities, as well as being open to new ones.
NF5.	Availability	The platform should ensure that users have always access to data and associated assets 24/7 with 99.9% reliability. This requirement entails stability in the presence of localized failure.
NF6.	Usability	The platform should have an attractive and intuitive User Interface. All the UI level functionalities should be validated on their usability through user-centric assessment and observation.
NF7.	Interoperability/conformance with standards	The platform should conform to standards when developing the individual components and their interfaces, in order for the system to be maintainable and to be able to interface with existing solutions or be extended with future functionalities.

ID	Non-functional Requirement	Description
NF8.	Security	The platform should support security mechanisms, which cater for the resources identification and the trusted provision of services in a SOA-based paradigm

2.3 The Aniketos Architecture

In order to accomplish the defined goals and objectives, as they were also analysed in Section 2.1, the Aniketos platform is designed so as to address security and trustworthiness in service composition in a three mode manner; *design-time support*, *runtime support* and *community support*. Figure 3 makes an overview of the Aniketos platform, as it has already been introduced in [2].

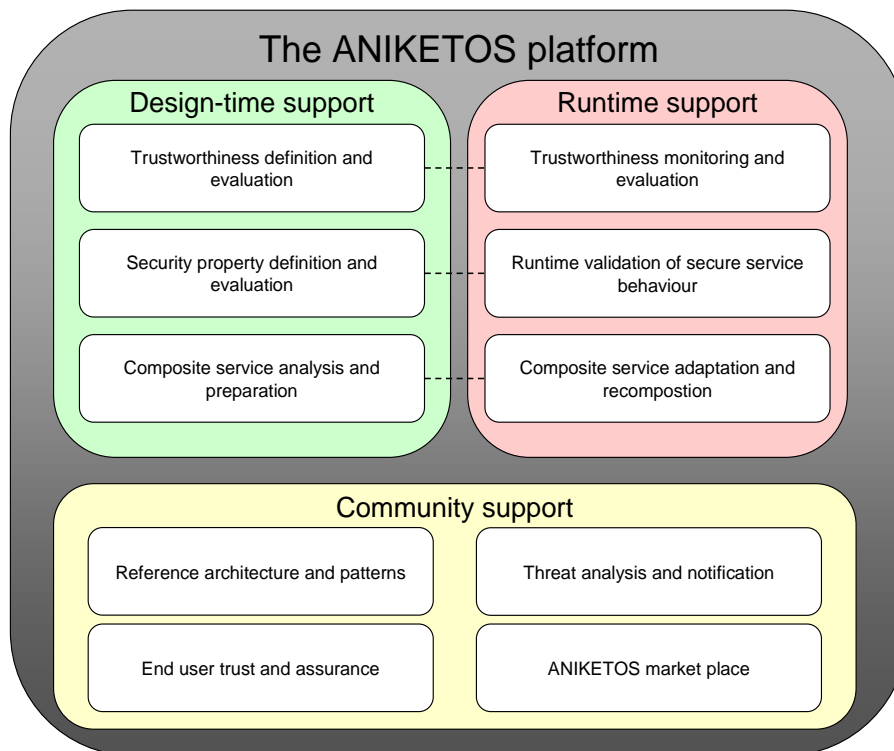


Figure 3: Overview of the Aniketos Platform

At *design time*, the Aniketos platform consists of methodologies and tools that define and evaluate trustworthiness and risk-based security properties over and between external service components.

This phase mainly involves a principal platform role, which is the **service developer**. The main functionalities, which are offered to the service developer, are summarised in the following:

- Perform service discovery and composition based on security properties and metrics, not just functional descriptors
- Select service providers and components by trustworthiness aspects for service composition
- Get informed about known threats and vulnerabilities

At *runtime*, the Aniketos platform is exploited to monitor and evaluate the trustworthiness and security violations of service components, also considering contextual information, such as any change in operation conditions and users' behaviour.

This phase mainly involves a principal platform role, which is the **service provider**. The main functionalities, which are offered to the service provider, are summarised in the following:

- Allow for proactive increase in trustworthiness by asking for more credentials
- Receive emerging threat notifications on potential damages in case of attack
- Assess the potentials for dynamic adaptation or recomposition of the service

In the *community support* mode, the Aniketos platform acts as the repository for providing all the necessary material to the interested stakeholders, including example services, demonstration material, tutorials, development patterns and guidelines. All the potential Aniketos platform user roles are involved in the community support mode, as follows:

Service developers:

- Find the Aniketos reference architecture and design patterns
- Provide threat analysis to guide design-time composition based security goals or service components included in the composite service

Service Providers:

- Provide notification to trigger runtime adaptation/recomposition based security goals or service components included in the composite service

The community support enables a certification programme that allows single-point-of-trust, enables responsibility handling and assures the end user in an easy and understandable way how he/she should relate to this service. The Aniketos marketplace offers a way of requesting/offering service components with defined security and trustworthiness properties, facilitating post-project continuation and development through revenue income.

2.4 Layered-based architectural design

Based on the specifications of D1.2 [1], this section maps the logical view of the Aniketos platform and environment components to a conceptual representation, which allocates the current implementation to a layered architecture, which is an instantiation of a Service-Oriented Architecture (SOA)-based system reference architecture [3][4][5].

SOA is widely used as an architectural pattern for contemporary systems development and integration. Its main principle is the separation of functionality into different units which interoperate with each other over a network. These units are usually developed as Web services and are reused by the system in order to achieve the desired functionality. A 'Web service' is defined by the World Wide Web Consortium (W3C), as "*a software system designed to support interoperable machine-to-machine interaction over a network*" [6]. The Web services can interoperate by exchanging data with each other directly, or by being coordinated by a central mechanism of the system. In general, SOA focuses on loose coupling of services in terms of used technology and programming language. By using common standards, web services have little or no dependency on each other, a fact that offers flexibility and scalability to the system. SOA is considered to be the evolution of the distributed systems architecture approach.

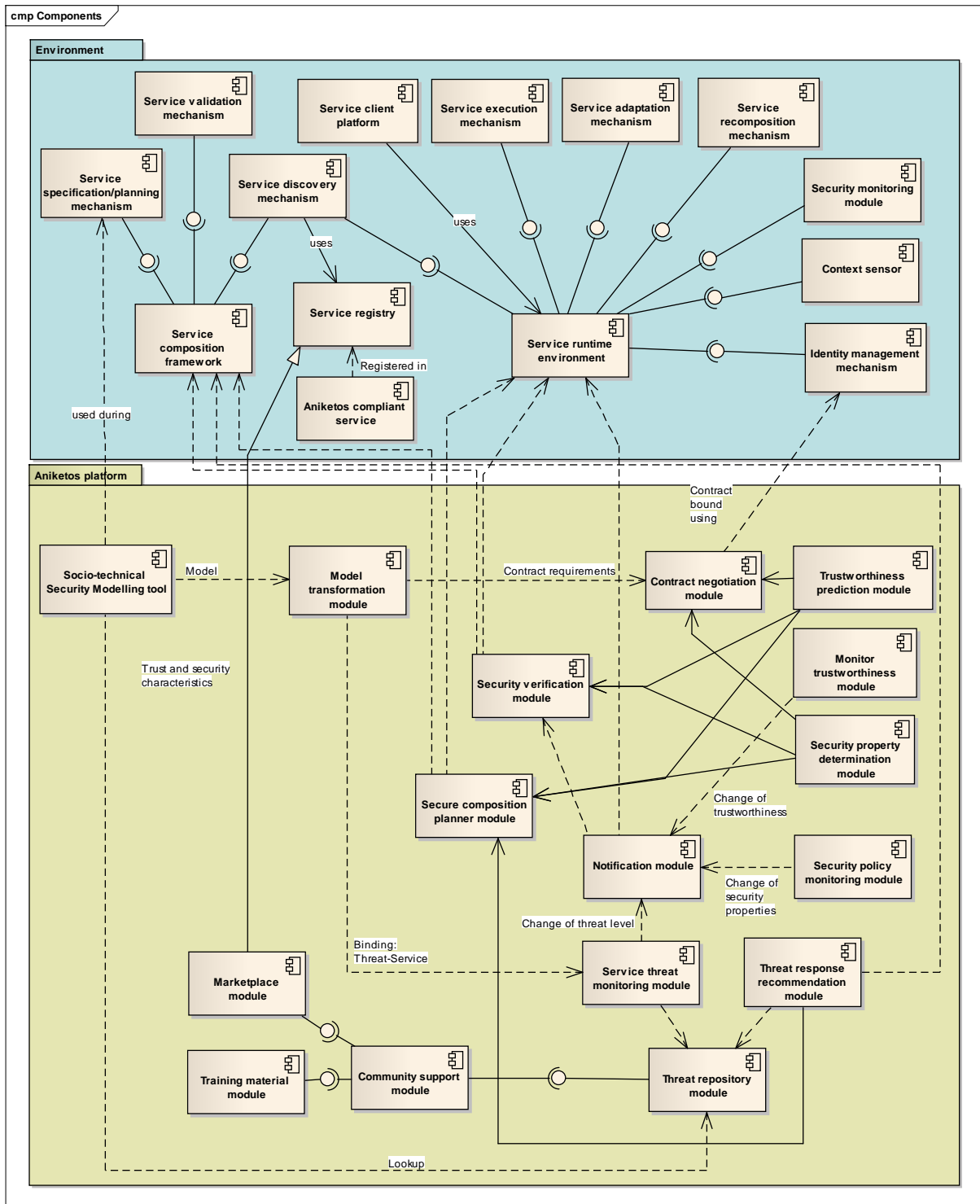


Figure 4: Components in the Aniketos platform and in the environment

Figure 4 makes an overview of the components in Aniketos platform and the environment, as they have been identified in D1.2 [1]. This view can be conceptually transformed into the layered-based representation of the Aniketos architectural design, which is shown in Figure 5.

In this layered structure, the top layer consists of the Interaction Layer, which enables the communication with the target end users. The components placed there offer a graphical representation of the tools to enable the users to interact with the rest of the Aniketos platform and environment components, both at design and runtime. At the same level, we could include the web services exposed externally by Aniketos to be consumed by other systems, platforms, tools, or devices.

Furthermore, all the resources exposed via HTTP protocol by the system to enable the interconnection of the components, ranging from RESTful and/or SOAP web services, can be placed here as well.

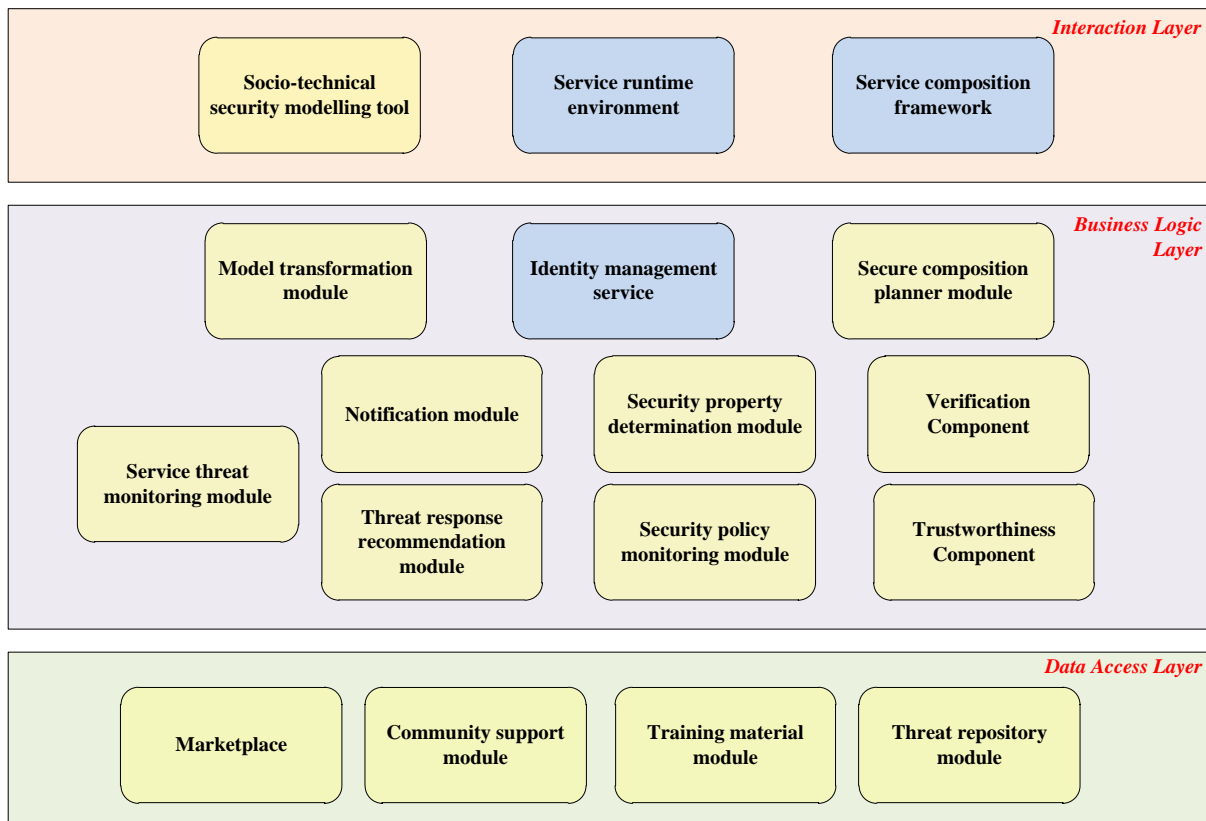


Figure 5: A layer-based conceptual representation of the Aniketos architectural design

At a second level, the Business Logic Layer hosts the components responsible for the business logic of the platform. More precisely, this layer hosts the components dealing with the analysis of security properties and trustworthiness in secure service composition, the monitoring of threats and the assessment of vulnerabilities. All of the Aniketos platform and environment components maintain at least a part exposing some kind of business logic.

At a third level, the Data Access Layer contains all the components, which are responsible for providing data access functionalities to the upper layer components. These objects encapsulate complexity of the data source models, including the service specifications, the threat definition part, the supporting documentation and training material, and offering a simplified facade to be used directly by the consumer components.

The following should be clarified that the internal structure of each component maintains software blocks, which may refer to more than one layer. For example, the Marketplace component exposes an interaction layer module, which enables end user to search for service specifications, including specific security characteristics. On the other hand, the socio-technical security modelling tool offers an interface for the end users to specify security assertions in service compositions, but it also maintains a business logic behind, which refers to the association model actors to certain notations.

For simplicity reasons, Figure 5 only allocates the components only to that layer, which seems to be closer to the main responsibility of each component.

With respect to Figure 4, Figure 5 makes the following grouping of components (which is close to the actual implementation):

- The Trustworthiness Component includes the functionalities of Trustworthiness Prediction and Trustworthiness Monitoring;

- The Verification Component encapsulates the functionalities of the Contract Manager Module, the Property Verification Module and the Composition Security Validation Module;
- The Marketplace maintains a service discovery mechanism, which facilitates accessing the relevant Service Registry;
- The Service Composition Framework includes service specification/ planning and service validation mechanisms;
- At the time being, the Service Runtime Environment incorporates the mechanisms for service execution, recomposition, adaptation and monitoring.

2.5 Development Methodology

This section describes the development methodology, which has been adopted in Aniketos to ensure that the project developments are produced on time and are compliant to the quality standards set in the project.

A number of available software design methodologies have to be examined [7], in order to come up with this methodology which better fits the Aniketos objectives. All of the proposed methodologies in the literature expose specific benefits and drawbacks and target to different level of detail the design and specification of the appropriate architecture to guide the software development.

For the sake of Aniketos, the project follows the Agile Software Development (ASD) methodology, which is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.

There are many specific ASD methods. Most of them promote development, teamwork, collaboration, and process adaptability throughout the life-cycle of the project. ASD goes beyond traditional software development processes and exploits an evolutionary method that is an iterative and incremental approach to software development. Thus, the requirements and design phases are iteratively met with the development phase to incrementally produce system software releases, which can be assessed over the suitability, the maturity and the immediate business value (through the envisaged piloting phase). On top of them, ASD foresees an intense testing phase, in which the unit testing is achieved from the developer's perspective and the acceptance testing is conducted from the customer's perspective.

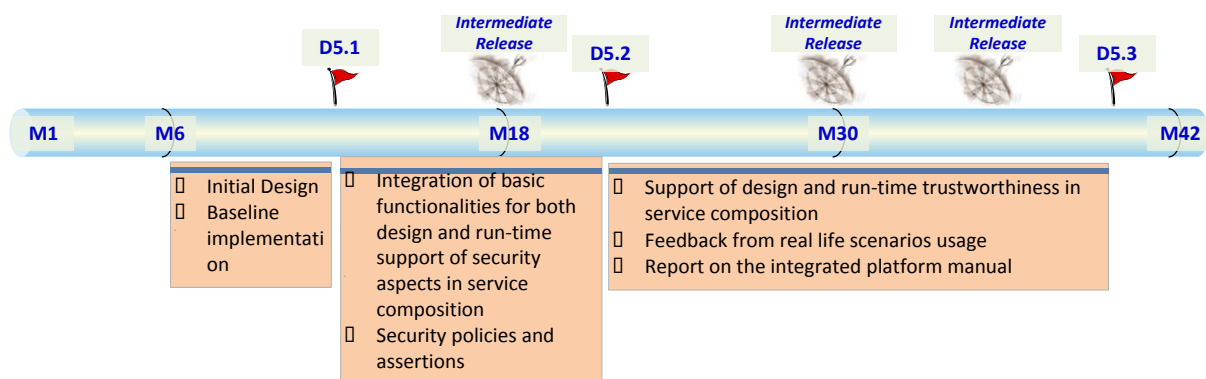


Figure 6: The basic steps for delivering the Aniketos platform, based on the agile software development methodology

Following the general project time plan, which identifies the interactions of WP5 with other WPs, the ASD leads to a time plan for WP5 as depicted in Figure 6. ASD states that “*the best architectures, requirements, and designs emerge from self-organising teams*”, while “*working software is the primary measure of progress*”. Thus, the Aniketos software methodology approaches the end user requirements iteratively by frequently delivering working software prototypes. These prototypes

enable the project development and business teams to work together and maximise the quality of the produced output.

ASD, which bears similarities with the rapid application prototyping (RAP) methodology [8], is suitable for the Aniketos development strategy, mainly because:

- A parallel process between requirements gathering from the end users and developing of the respective software environment can be followed, leading to business oriented people to actively participate in the specification of use cases and the evaluation of the system developments and provide valuable feedback in an iterative way;
- The work on individual research areas is split among small groups comprising the separate WP development teams;
- As an IP research project, Aniketos can be benefit of frequent releases to align the work done among the individual teams;
- The producing releases can be exchanged among senior technical teams and business oriented groups to evaluate the effectiveness of the platform in real business situations.

In order to ensure compliance of all developments to a common strategy, a continuous and iterative interaction of WP5 with the core development phase and the evaluation and assessment phase (through the deployment of the appropriate case studies) is adopted, as shown in Figure 7, while the following procedures have been defined:

- WP5 defines the way that the Aniketos platform components are being implemented, through the specification of templates for delivering the design and run time functionalities;
- WP1-WP4 implementation is guided by these templates, which provide their interfaces based on the envisaged functionalities and their high-level specifications in D1.2 [1], detailed respectively in D2.1 [9], D3.1 [10] and D4.1 [11];
- WP5 wraps the technologies developed in WP1-WP4 and delivers prototypes of the Aniketos platform, which can interoperate with the environment components and offer certain functionalities at design and run time;
- The integrated package is exploited by the piloting activities to deploy real life scenarios (case studies) on the use of Aniketos;
- The deployed case studies are assessed on the usability of the Aniketos platform to express real business needs and advance current business practices in the selected domains.

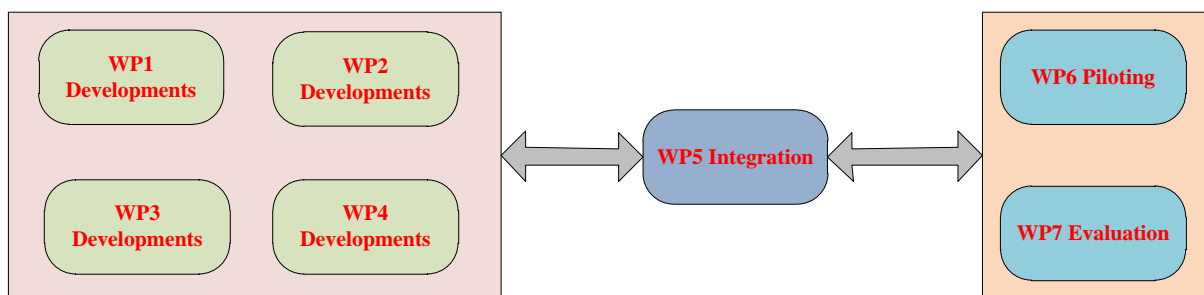


Figure 7: Continuous interaction of WP5 with the development and assessment phases

3 Analysis of technological background

3.1 Introduction

This section presents the background which is relevant to the developments of the Aniketos platform. As previously mentioned in this document and recalling the project research and scientific objectives as depicted in Figure 2, the Aniketos development targets to support security and trustworthiness in service composition both at design time and runtime. As such, the technologies, which are deemed necessary for doing so span across the design time specification of composite services, the analysis of existing security mechanisms and the runtime deployment and management of composite services. More specifically, the remaining of this section is split as follows:

- It adopts Eclipse as the Software Development Kit (SDK) and presents the Eclipse plug-in mechanism for supporting the service modelling and composition at design time.
- It presents the main Web service composition standard languages which are core to Aniketos secure composition concepts.
- It analyses the security framework technologies giving emphasis on the security specifications for Web-based systems.
- It presents the communication technologies for enabling the remote Aniketos platform components to interoperate with each other and with external environment components and other third party platforms.
- It focuses on the technologies for the runtime support of security and trustworthiness in service composition.
- It elaborates on the technologies for the data access layer, considering persistency and data storage functionalities.

It should be noted that the analysis of the Aniketos-related technologies in the next paragraphs of this Section 3 is based on the respective bibliography, which is referred in line with text. Where appropriate, the original information has been reworked to fit the needs of the Aniketos project. In that respect, the presentation of the Aniketos relevant technologies in Section 3 is not an intellectual property of the Aniketos Consortium, but belonging to the corresponding references.

3.2 Development of the Aniketos Platform

3.2.1 Eclipse Plug-in Mechanism

As per [12], Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the runtime system, in contrast to some other applications where functionality is typically hard coded. The runtime system of Eclipse is based on Equinox [13], an OSGi standard compliant implementation.

This plug-in mechanism is a lightweight software framework. In addition to allowing Eclipse to be extended using other programming languages, such as C and Python, the plug-in framework allows Eclipse to work with typesetting languages like LaTeX, networking applications such as telnet, and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and Concurrent Versions System (CVS) support is provided in the Eclipse SDK, with Subversion support provided by third-party plug-ins.

With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every developed plug-in integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are "created equal". Eclipse provides plug-ins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plug-ins

include a UML plug-in for Sequence and other UML diagrams, a plug-in for Database Explorer, and many others.

Eclipse is an open platform [14], i.e. it is designed to be easily extensible by third parties. Around the Eclipse SDK new products and tools (plug-ins) can be created, as shown on the Eclipse plug-in architecture in Figure 8. A plug-in is nothing but another java program, which extends the functionality of Eclipse in some way. Each Eclipse plug-in can either consume services provided by other plug-in or can extend its functionality to be consumed by other plug-ins. The plug-ins are dynamically loaded by Eclipse at run time on demand basis.

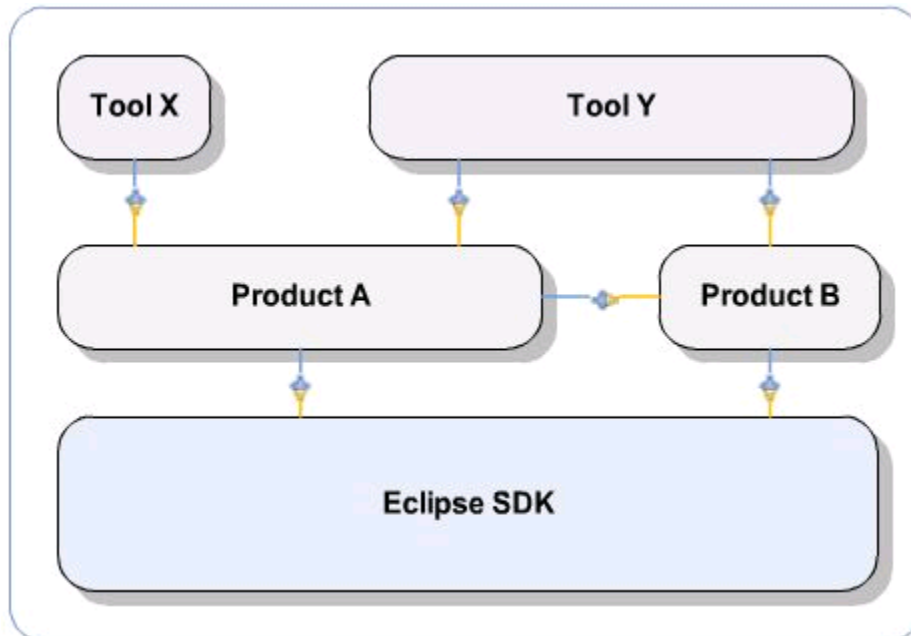


Figure 8: The Eclipse plug-in architecture (please refer to [14])

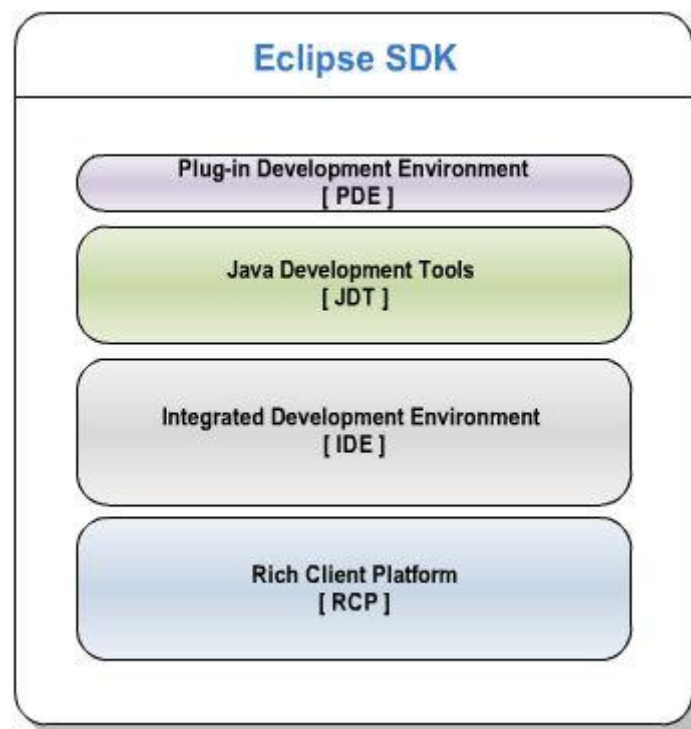


Figure 9: The Eclipse plug-in layers (please refer to [14])

Eclipse SDK consists of the following layers, as shown in Figure 9:

- Rich Client Platform (RCP): On the bottom is RCP which provides the architecture and framework to build any rich client application.
- Integrated Development Environment (IDE): It is a tools platform and a rich client application itself. We can build various form of tooling by using IDE for example Database tooling.
- Java Development Tools (JDT): It is a complete java IDE and a platform in itself.
- Plug-in Development Environment (PDE): It provides all tools necessary to develop plug-ins and RCP applications.

All the layers in eclipse SDK are made up of plug-ins. Except from a small kernel, almost all of the functionality of eclipse is located in different plug-ins.

A plug-in can be delivered as a jar file. A plug-in is self-contained bundle in a sense that it contains the code and resources that it needs to run: code, image files, resource bundles etc. A plug-in is self-describing by means of describing which it is and what it contributes to the world. It also declares what it requires from the world.

3.3 Web service composition standard languages

Web service composition can be viewed in a process-oriented perspective, so a language to define how the web services have to be composed into business processes is needed.

There are two ways the services can be composed, namely orchestration or choreography. Here we will focus on orchestration, in which a central process invokes operations to be performed by the web services involved and coordinates the execution of the operations. The central process is a web services itself and is available through interfaces to clients that want to consume the new composite service. The orchestration approach is shown in Figure 10.

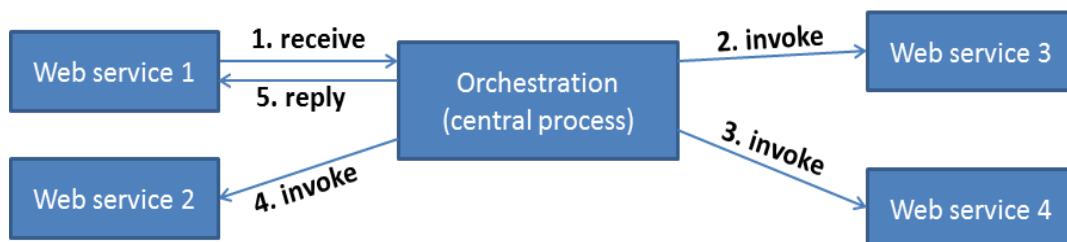


Figure 10: Service composition: orchestration approach (please refer to [15])

In process-oriented approach, service composition is described by the means of workflow languages and technologies. The composition workflow defines the operations to invoke and define the execution order of the invocations.

We will consider the following workflow languages, considered as de facto standards for Web Service Composition, namely:

- Business Process Modelling Notation (BPMN), which is a graphical representation used to model the processes within a workflow;
- Business Process Execution Language for Web Services (BPEL4WS, also known as BPEL); it is an XML-based service composition languages for building, specifying and executing business processes for web services composition

3.3.1 Business Process Modelling Notation

As said, service orchestration needs to define the sequence of steps within a business process, including conditions and exceptions, and then creates a central controller to implement the sequence. In a SOA environment, the individual steps of the sequence are implemented by operations on web services. The sequence can be implemented with a variety of different techniques. For complex orchestrations, a tool to create a visual model of the sequence can be used. From the visual model it should be possible to generate the code that executes that sequence, typically within a dedicated run-

time environment. This is the typical BPM approach [16]. BPMN is today's standards for orchestration.

BPMN was created by the Business Process Management Initiative (BPMI) and has been designed specifically with web services in mind. Its first goal was to provide a notation for process modelling that was readily understandable by all business users. Business users range from the business analysts that create the initial drafts of the processes to the technical developers responsible for implementing the technology that will perform those processes.

BPMN is a flow chart based notation for defining business processes. The modelling of the business process aims at capturing the ordered sequence of business activities, i.e. the workflow. BPMN proposes language constructs that facilitate the modelling of workflows at a high level.

BPMN is well suited for modelling web services based business processes and is closely related to service composition languages like BPEL, described in the next paragraph. It can support different methodologies as well as different modelling goals (e.g., orchestration and choreography).

The modelling in BPMN is done by expressing business processes through the so called business diagrams. A business process diagram is based on flowchart techniques, so it's an intuitive notation easily understandable by technical as well as business users.

A business process diagram is a simple diagram built up by using a small set of graphical elements. The core set of graphical elements can be grouped into four main categories [17]:

- Flow objects: events, activities and gateways;
- Connecting objects: sequence flow, message flow and association
- Swimlanes: pool and lane
- Artifacts: data object, group and annotation

As an example of the process model, Figure 11 shows a BPMN model for a travel booking process. The process begins with the receipt of a request for a travel booking. After a check on the credit card, reservations are made for a flight, a hotel and a car. The car reservation may take more than one attempt before it is successful. After all three reservations are confirmed, a reply is sent.

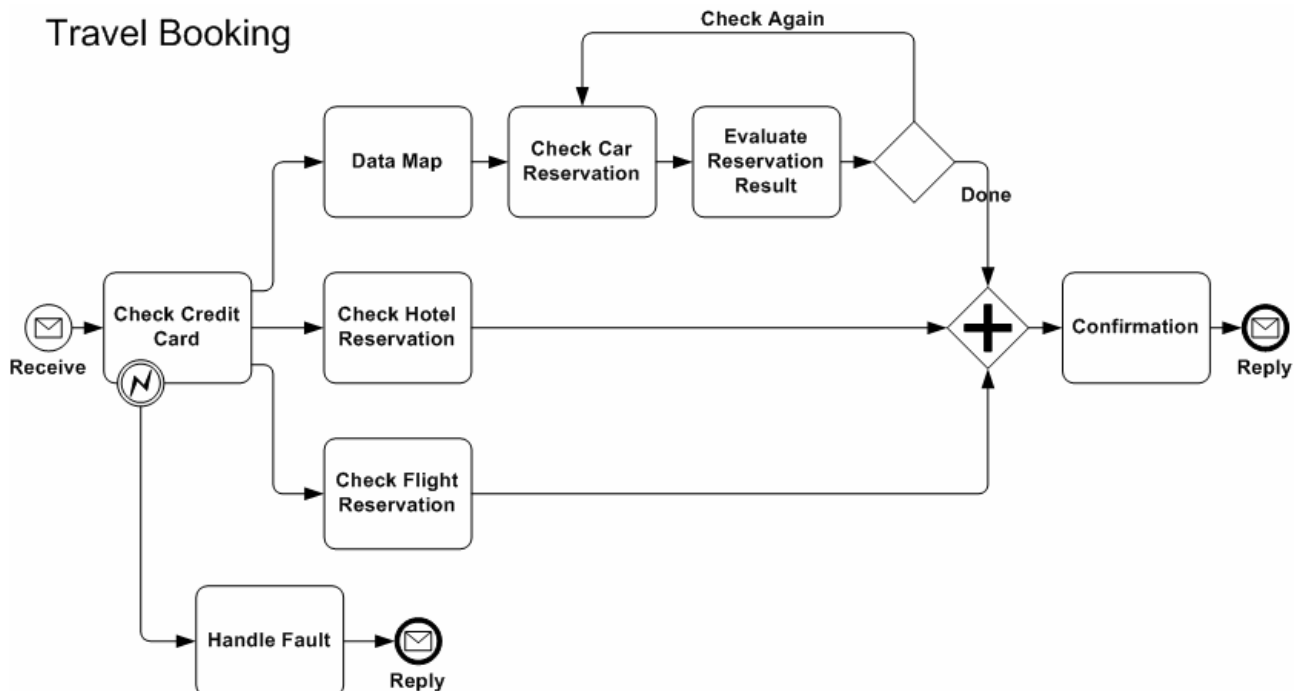


Figure 11: BPMN Diagram for travel booking process (please refer to [18])

BPMN diagrams, as the example in Figure 11, can be used within many methodologies and for many purposes, from high-level descriptive modelling to detailed modelling intended for process execution. When one of the purposes of the process model is to define process execution and create the BPEL file for this purpose, then the process model will have to be developed with a modelling tool designed for this purpose. The diagram itself will not display all the information required to create a valid BPEL file.

A BPMN diagram is intended to display the basic structure and flow of activities and data within a business process. Therefore, a modelling tool is necessary to capture the additional information about the process that is necessary to create an executable BPEL file. This means that a Business Process developed by a business analyst can be directly applied to a BPM engine instead of translating it into other languages.

3.3.2 Business Process Execution Language

The Business Process Execution Language (BPEL), standardized by OASIS in 2004 [19], is one of the most important technologies of SOA. It consists in an XML-based language for building, specifying and executing business processes for web services composition and orchestration. Through BPEL, a programmer can describe a business process in an executable or abstract way.

- An executable business process specifies the exact details of a business process and can be executed by an orchestration engine. BPEL is usually used to specify this kind of business process.
- An abstract business process does not include the internal details of the process flow and is not intended to be executed. It is rarely used and it is usually defined to describe the behaviour of a service without knowing in which business process it will take part.

One of BPEL's primary roles is to model Web service interactions on a distributed system by defining a new web service which is the composition of a set of existing services. This new composed web service has a proper WSDL interface that provides a set of operations like any other web services. Only through these operations, it is possible to invoke the executable business process.

BPEL basically consists of a set of primitive and structure activities.

The first ones are used for common tasks such as:

- Invoking other web services : <invoke>
- Waiting for a client to invoke the business process: <receive>
- Manipulating data variable: <assign>
- Indicating faults and exceptions: <throw>
- Generating a response: <reply>
- Waiting for some time: <wait>
- Terminating the process: <terminate>

The structure activities are the combination of these primitive activities. They enable the possibility to build more complex algorithms and so to specify the steps of the business process.

- <sequence>: to define a set of activities that will be invoked in an ordered sequence
- <while>: to define loops, etc.
- <flow>: to define a set of activities that will be invoked in parallel
- <switch>: Case-switch construct for implementing branches
- <pick>: to select one or several alternative paths

In Figure 12, an example of a BPEL process is shown. It is composed only by a sequence of three activities.

In order to be executed a BPEL process requires a BPEL engine, which also provides the possibility to control the process instances that are executing. Some of the most popular BPEL engines based on Java EE are Apache ODE, ActiveVOS, jBPM, Oracle BPEL Process Manager and OW2 Orchestra.

BPEL is often associated with BPMN, which also seeks to streamline the BPM modelling process. Unlike BPEL, BPMN has a visual component that makes it easier to understand for business people not familiar with programming.

```

<?xml version = "1.0" encoding = "UTF-8" ?>
<process name="BPELProcess"
  targetNamespace="http://xmlns.oracle.com/BPELProcess"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/
    03/business-process/"
  xmlns:xp20="http://www.oracle.com/XSL/Transform/java/
    oracle.tip.pc.services.functions.Xpath20"
  xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
  xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
  xmlns:client="http://xmlns.oracle.com/BPELProcess"
  xmlns:ora="http://schemas.oracle.com/xpath/extension"
  xmlns:orcl="http://www.oracle.com/XSL/Transform/java
    /oracle.tip.pc.services.functions.ExtFunc">

  //PARTNERLINKS
  //List of services participating in this BPEL process
  <partnerLinks>
    <partnerLink name="client" partnerLinkType="client:BPELProcess"
      myRole="BPELProcessProvider"
      partnerRole="BPELProcessRequester"/>
  </partnerLinks>
  //VARIABLES
  //List of messages and XML documents used within this BPEL process

  <variables>
    //Reference to the message passed as input during initiation
    <variable name="inputVariable"
      messageType="client:BPELProcessRequestMessage"/>
    //Reference to the message that will be sent back to the requester during callback
    <variable name="outputVariable"
      messageType="client:BPELProcessResponseMessage"/>
  </variables>

  // ORCHESTRATION LOGIC
  // Set of activities coordinating the flow of messages across the
  // services integrated within this business process

  <sequence name="main">
    // Receive input from requestor.
    <receive name="receiveInput" partnerLink="client"
      portType="client:BPELProcess" operation="initiate"
      variable="inputVariable" createInstance="yes"/>

    <assign name="Assign">
      <copy>
        <from variable="inputVariable" part="payload"
          query="/client:BPELProcessProcessRequest/client:input"/>
        <to variable="outputVariable" part="payload"
          query="/client:BPELProcessProcessResponse/client:result"/>
      </copy>
    </assign>

    //Asynchronous callback to the requester
    <invoke name="callbackClient" partnerLink="client"
      portType="client:BPELProcessCallback" operation="onResult"
      inputVariable="outputVariable"/>
  </sequence>
</process>

```

Figure 12: Example of a BPEL process (please refer to [20])

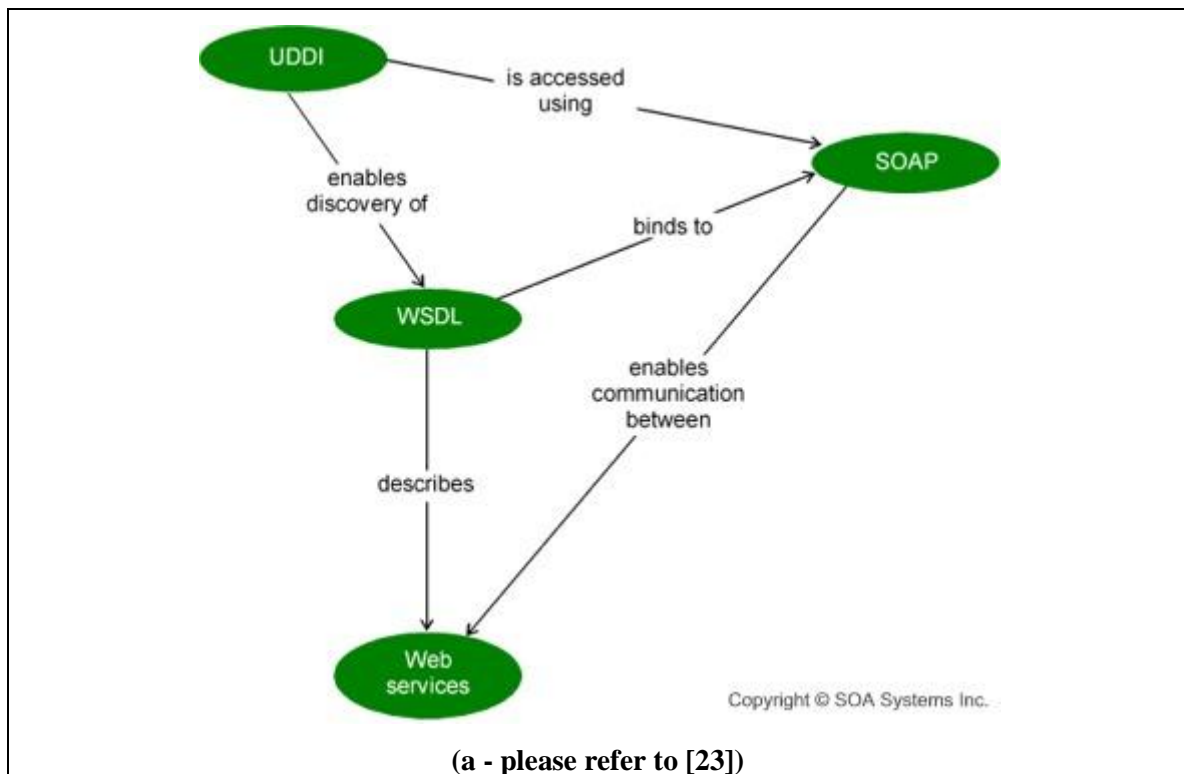
3.4 Security Framework Technologies

3.4.1 Introduction

As per [21], “the Web services platform is defined through a number of industry standards that are supported throughout the vendor community. First-generation technology platforms comprise the following core open technologies and specifications”:

- Web Services Description Language (WSDL)
- XML Schema Definition Language (XSD)
- SOAP (formerly the Simple Object Access Protocol)
- UDDI (Universal Description, Discovery, and Integration) [22]
- WS-I Basic Profile

These specifications have been around for some time and have been adopted across the IT industry. However, the platform they collectively represent seriously lacks several of the quality of service features required to deliver mission critical, enterprise-level production functionality.



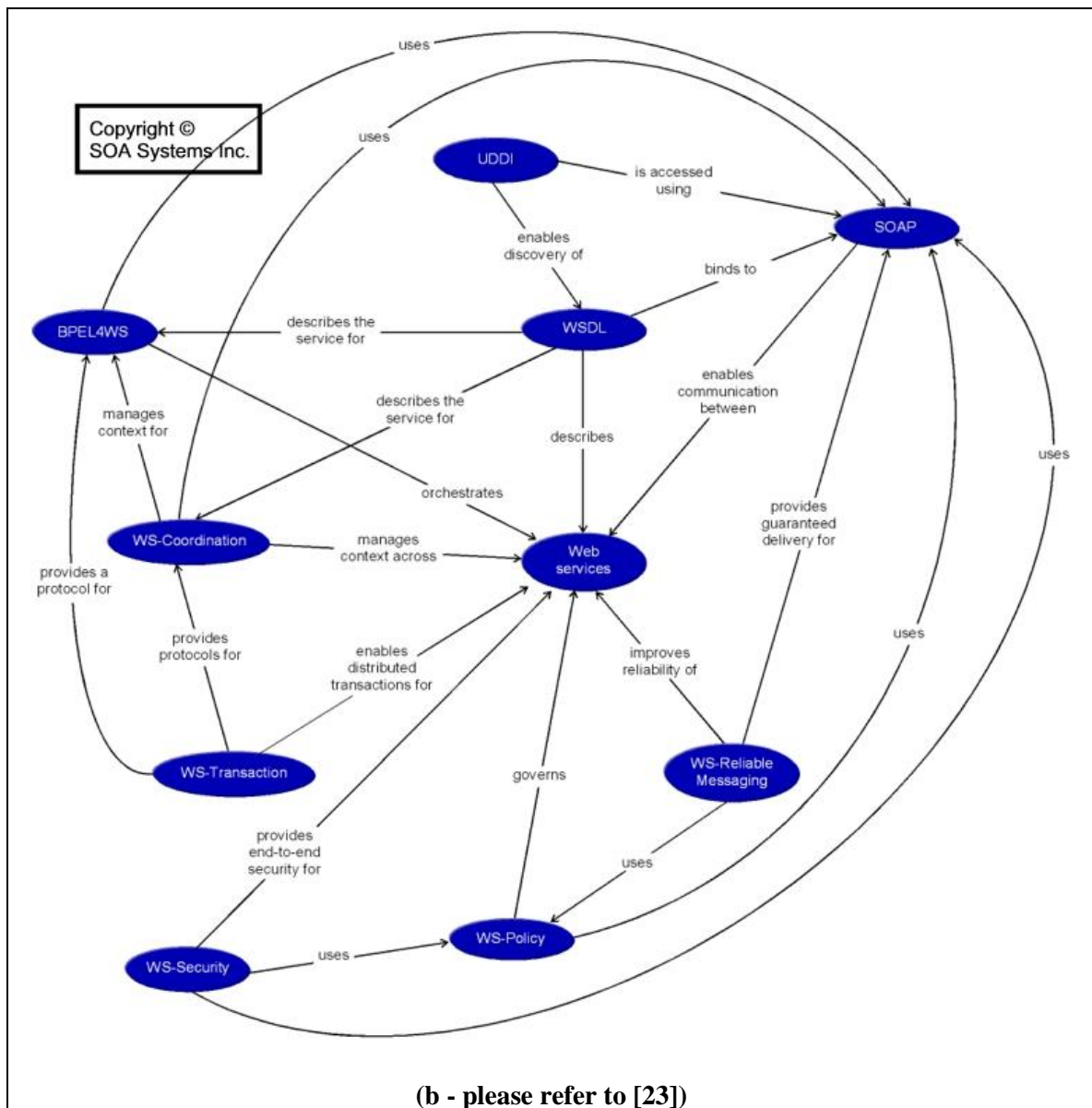


Figure 13: Technologies and specifications for Web Service – based platforms: a) First-Generation, b) Second Generation (please refer to [24])

Some of the greatest quality of service-related gaps in this first-generation platform lie on the areas of message-level security, cross-service transactions, and reliable messaging. These, along with many other extensions, are being provided by a second-generation Web services platform(s), consisting of a set of Web services technologies and specifications which will be briefly described in the next section.

3.4.2 Overview of security specifications

This section makes an overview of the security specifications and relevant standards, as they are exhaustively presented in [25]. For the sake of completeness of this document, these aspects are reproduced here, based on the analysis given in [25]. Once again, the information provided in this section is not an intellectual property of the Aniketos Consortium, but belonging to the corresponding reference [25].

As the complexity and sophistication of application and business logic within Web services increases, so does the risk associated with putting business intelligence “out there”. The purpose of this section is to create an awareness of the general aspects of Web services security. While the security framework

established by many specifications that provide standards for XML and Web services is relatively new, most of the principles behind these standards are not. The fundamental characteristics of a primitive security architecture are just as relevant to service-oriented environments as they are to traditional distributed applications.

The following Table 3 shows some of the most important Web services standards and XML security specifications, organized by how they relate to the common security requirements of *identification*, *authentication*, *authorization*, *confidentiality* and *integrity* in the realm of service-oriented architectures.

Table 3: Web services standards and XML security specifications

<p>Identification The recipient of a message needs to be able to identify the sender.</p>	WS-Security Framework
<p>Authentication The recipient of a message needs to verify that the claimed identity of the sender is valid.</p>	Extensible Access Control Markup Language (XACML)
<p>Authorisation The recipient of a message needs to determine the level of the sender's security clearance. This can relate to which operations or which data the sender is granted access to.</p>	Extensible Rights Markup Language (XrML)
	XML Key Management (XKMS)
	Security Assertion Markup Language (SAML)
	OpenID
	OAuth
<p>Confidentiality The contents of a message cannot be viewed while in transit, except by authorized services.</p>	WS-Security Framework
	XML-Encryption
	Secure Sockets Layer (SSL)
<p>Integrity A message remains unaltered during transmission, up until actual delivery.</p>	WS-Security Framework
	XML-Digital Signatures

These and other specifications form the building blocks that can be assembled to create service-oriented security models. In the following, a brief overview of each specification is given.

XML Key Management Specification (XKMS)

XML Key Management establishes standard means of obtaining and managing public keys. Even though XKMS is compatible with a number of public key infrastructure (PKI) technologies, it does not require any of them, and removes the need for integrating proprietary PKI products.

The XML Key Management specification consists of two complementary standards: the XML Key Registration Service and the XML Key Information Service specifications. Together, they allow for the integration of a number of security technologies, including digital signatures, certificates, and revocation status checking. For instance, XKMS can enlist XML-Digital Signatures to protect the integrity of XML document content.

Extensible Access Control Markup Language (XACML) and Extensible Rights Markup Language (XrML)

The XACML specification consists of two related vocabularies: one for access control and one that defines a vocabulary for request and response exchanges. Through these languages, the creation of fine-grained security policies is possible.

It is important not to confuse XACML with the WS-Policy specification, which also can be used to define policies, and is considered part of the WS-Security framework. The main conceptual difference between these two languages is that while XACML is based on a well-defined model that provides a formal representation of the access control security policy and its working, WS-Policy has been developed without taking into consideration this modelling phase [26]. An additional specification that may be relevant to certain environment, if someone transports files with different digital formats, is the Extensible Rights Markup Language (XrML).

Security Assertion Markup Language and OpenID

Single sign-on technologies address an administration problem that has emerged when an enterprise environment consists of applications that independently control user access lists. If a single sign-on system is not already in place, adding Web services can contribute to the decentralized proliferation of user credentials. By opening up new integration channels, more users may be required to access applications. This can lead to an ever-increasing maintenance effort.

Popular technologies for single sign-on include the Security Assertion Markup Language (SAML) [27] and the OpenID [28]. They provide mechanisms for both authentication and authorization processes. Both request and response message formats are defined to facilitate the transmission of necessary credentials within a Web service activity.

SAML and OpenID are decentralized standards, meaning that are not controlled by any website or service provider.

OAuth (Open Authorization)

OAuth (Open Authorization) is defined in [29] as an open standard for authorization. OAuth allows users to share their private resources stored on one site with another site without having to hand out their credentials, typically username and password. As per [29], “OAuth allows users to hand out tokens instead of credentials to their data hosted by a given service provider. Each token grants access to a specific site for specific resources and for a defined duration. This allows a user to grant a third party site access to their information stored with another service provider, without sharing their access permissions or the full extent of their data.”

XML-Encryption and XML-Digital Signatures

These two key specifications protect the actual content within XML documents.

The XML-Encryption [30] specification contains a standard model for encrypting both binary and textual data, as well as a means of communicating information essential for recipients to decrypt the contents of received messages.

XML-Digital Signatures establishes a standardized format for representing digital signature data. Digital signatures establish credibility within a message, as they assure the recipient that the message was in fact transmitted by the expected partner service. It also provides a means of communicating that the message contents were not altered in transit, as well as support for standard non-repudiation. As with the XML-Encryption standard, XML-Digital Signature also supports binary and textual data.

Secure Sockets Layer

The Secure Sockets Layer (SSL) technology enables transport-level security. Accessing a secured site using a browser is a fairly safe procedure, when communication is encrypted. That is because the connection is exclusive to the client browser and the Web server that acts as the gateway to internally hosted application logic.

As per [25], once a Web service transmits a message it does not always know where that message will be travelling to, and how it will be processed before it reaches its intended destination. A number of intermediary services may be involved in the message path, and a transport-level security technology will protect the privacy of the message contents only while it is being transmitted between these intermediaries (see Figure 14).

In other words, SSL cannot protect a message during the time that it is being processed by an intermediary service. This does not make SSL unnecessary within a service-oriented communications framework, it only limits its role. A number of additional technologies (discussed throughout this section) are required to facilitate the message-level security required for end-to-end protection.

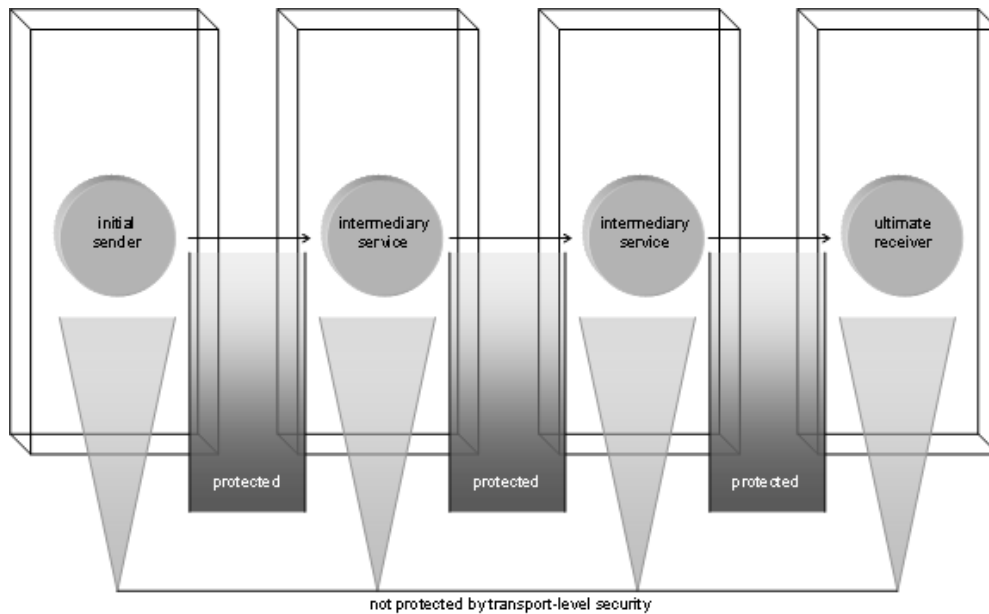


Figure 14: SSL protection across multiple services (please refer to [25])

WS-Security framework

This important document establishes fundamental and conceptual security standards, and also defines a set of supplementary specifications that collectively form a Web service-centric security framework. WS-Security (also known as the Web Services Security Language) can be used to bridge gaps between disparate security models, but also goes beyond traditional transport-level security to provide a standard end-to-end security model for SOAP messages.

Because service-oriented environments sometimes require that intermediaries be involved with the delivery of messages, an end-to-end security model is needed, so that the contents of SOAP messages remain protected throughout a message path. This is different from the traditional point-to-point model, for which transport-level security has generally been sufficient. In fact, you could view a message path involving intermediaries as a series of point-to-point connections.

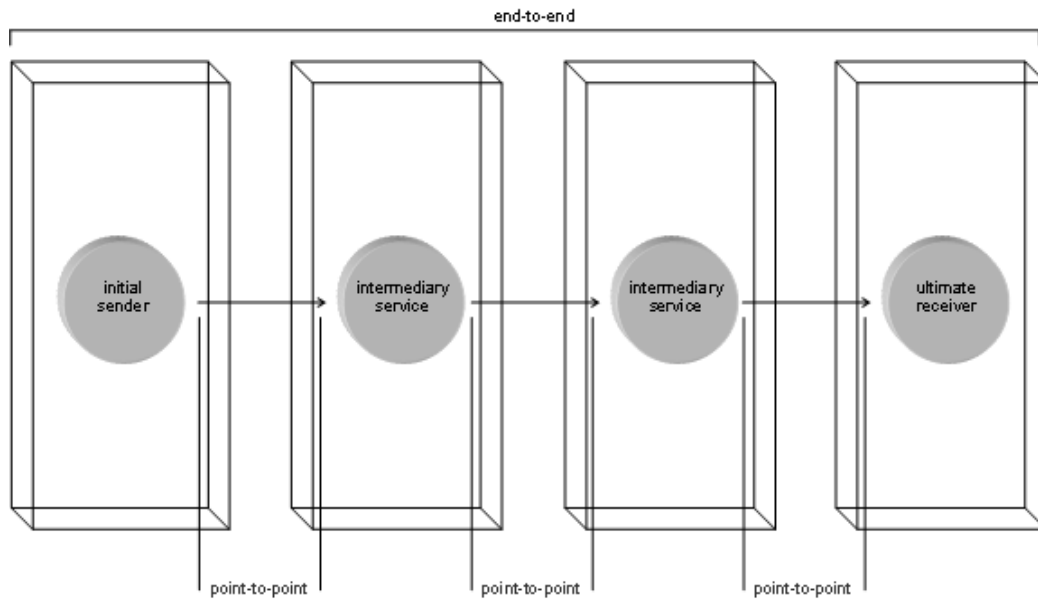


Figure 15: WS-Security Framework concept (please refer to [25])

To secure a message path from end-to-end, WS-Security implements security measures through the use of SOAP header blocks that travel with the message.

Although the WS-Security framework complements a number of the specifications described earlier in this section, it is supported further by a series of supplementary standards.

Table 4: WS-Security Framework Standards ([25])

WS-Security Framework		
WS-Policy	WS-Trust	WS-Privacy
WS-SecureConversation	WS-Federation	WS-Authorization

3.4.3 OpenID and OAuth

OpenID promotes an open source and decentralized system for identity management purposes, especially intended for federated authentication. Through OpenId, a user identifier is converted to an URL or Extensible Resource Identifier (XRI), which is a scheme and resolution protocol for Internet, specifically designed for digital identity in a cross-domain SSO environment, provided by an Identity Provider. In this way, users can be authenticated by any server that supports this specification, without creating a new user account. User must only select a suitable identity provider.

A user accepts an identity assertion from any identity provider (although clients are free to whitelist or blacklist providers). All that OpenID is supposed to do is to allow an OpenID provider to prove that the user is the delegated to make use of specific resources.

OpenID Scenario

- User wants to access his account on flickr.com.;
- Flickr .com - the “*Relying Party*” in OpenID speech, asks the user for his OpenID, an URL or XRI depending on OpenID version used;
- User enters his OpenID. The Identity Provider and the relying party establish a link (via a shared key), used for signing the messages between them. By using this link, the need for verifying each message’s signature is removed;

- Flickr.com redirects the user to his OpenID provider. After transforming this ID into a canonical URL. The relying party then begins a search process to find the OP Endpoint URL, which is required to perform the authentication process;
- User authenticates himself to the OpenID provider. The Identity Provider verifies the user is allowed to start an authentication process against that provider;
- OpenID provider redirects the user back to flickr.com, setting the result of the authentication process;
- Flickr.com allows the user to access his account. The relying party verifies the information provided by the Identity Provider.

OAuth [31] is intended for delegated authorisation, to grant access to functionality, data access, etc... without having to deal with the original authentication provider. A user registers with an identity provider, which provides authorization tokens, which it will accept to perform actions on the user's behalf. Meaning the user is authorizing a third-party service access its personal data, without giving out a password. OAuth "sessions" generally live longer than "user sessions".

OAuth is a service that is complementary to, but distinct from OpenID. The process of the OAuth and the relevant steps are seen in Figure 16.

As an example: Flickr uses OAuth to allow third-party services to post and edit a person's picture on their behalf, without them having to give out their flicker username and password.

Google [32] provides a paradigm on how to use and practise with the OAuth authorisation "dance" scenario.

OAuth Scenario

- User is on flickr.com and wants to import his contacts from gmail.com;
- Flickr.com - the "Consumer" in OAuth speech, redirects the user to gmail.com - the "Service Provider";
- User authenticates himself to gmail.com (which can happen by using OpenID as we described previously);
- gmail.com asks the user whether he wants to authorise flickr.com to access his contacts;
- User makes his choice;
- gmail.com redirects the user back to flickr.com.;
- Flickr.com retrieves the contacts from gmail.com.;
- Flickr.com informs the user that the import was successful.

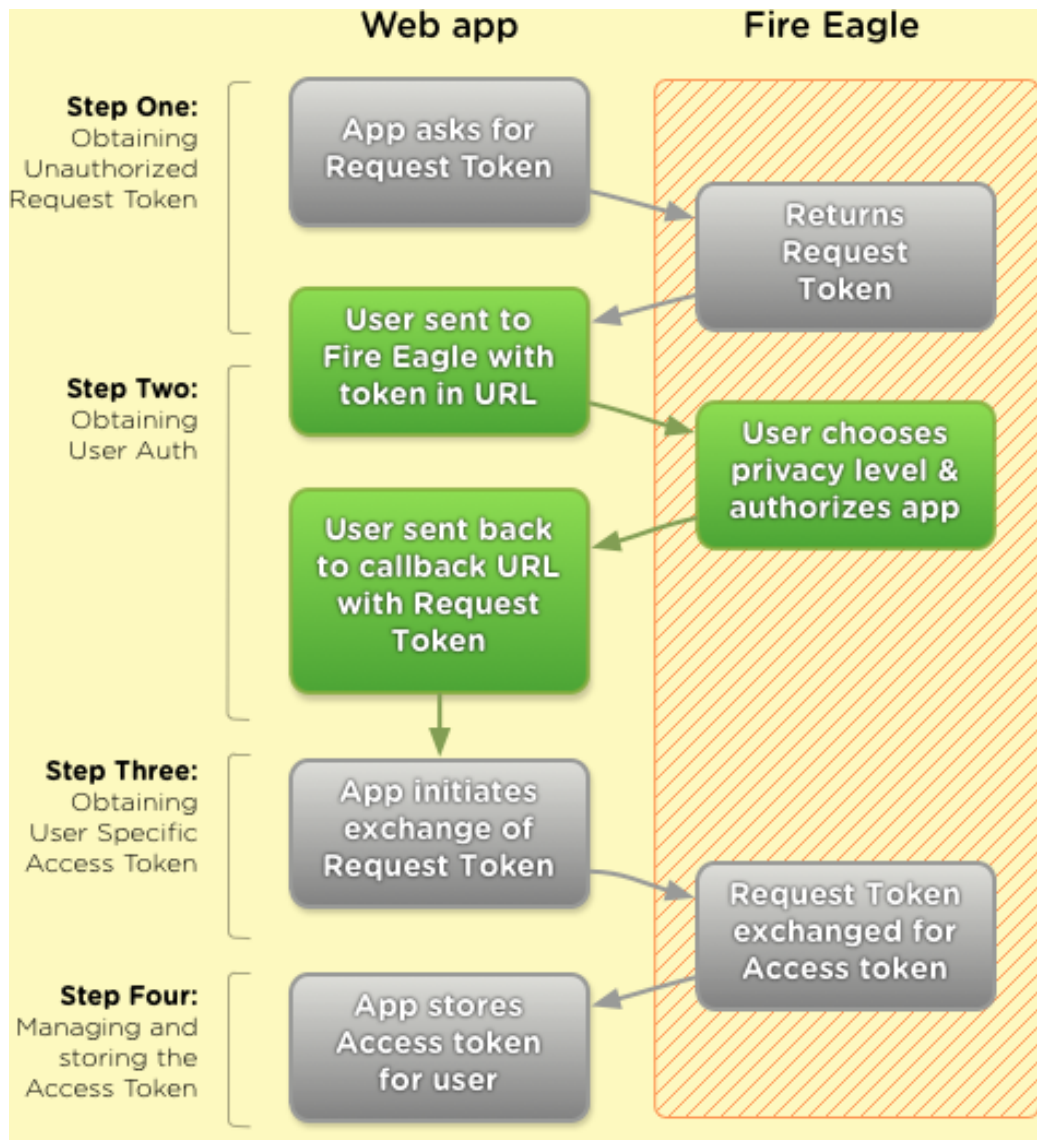


Figure 16: The OAuth Authorisation process steps (please refer to [31])

3.4.4 SAML2 & XACML

A critical component for the implementation of federated identity is the SAML, an XML framework for exchanging data. The Liberty Alliance Project builds on SAML to deliver more extensive federated identity.

On the other hand, OASIS has provided XACML that allows the designer to acquire this structure in an easier way.

Both specifications apply to the same domain: authentication, authorisation and access control domain. Although they belong to the same domain, they address different problems. SAML focuses on providing a mechanism for making authentication and authorisation assertions and communicating them between cooperating parties, whereas XACML focuses on providing the mechanism for arriving at those authorization decisions. In addition to process the authorization requests, XACML also defines the way for creating the complete infrastructure of rules, policies, and policy sets to arrive at the authorization decisions.

For example, imagine a scenario where a subject makes a request to access a resource. The Policy Enforcement Point (PEP) checks with the Policy Decision Point (PDP) before making an access decision. Generation of the request to access the resource and the subsequent response granting or

denying access falls under the domain of SAML. XACML addresses the exchange of policy decisions between the PEP and the PDP.

3.5 Communication Technologies

The communication infrastructure of the Aniketos Platform will follow the design principles of the Service Oriented Architecture. The case studies that will be examined during the projects lifetime involve a SOA-based system, i.e. a system that packages functionality as a suite of interoperable services, which can be used within multiple, separate systems from several business domains.

A SOA system can be implemented with a number of different technologies. In this section, we examine how an Enterprise Service Bus (ESB) can assist in the implementation of a reliable SOA-based system. Finally, this section compares REST and SOAP Web services, which are the main communication technologies available today.

3.5.1 Enterprise Service Bus

An ESB [33][34] is a software architecture construct, which provides fundamental services for complex architectures via an event-driven and standards-based messaging engine (the bus). Developers typically implement an ESB using technologies found in a category of middleware infrastructure products, usually based on recognized standards.

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

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

Figure 17 depicts the architecture of a SOA system implemented with an ESB. The system integrates new and legacy applications that need to communicate with each other and exchange information. A variety of technologies, protocols and message formats are used.

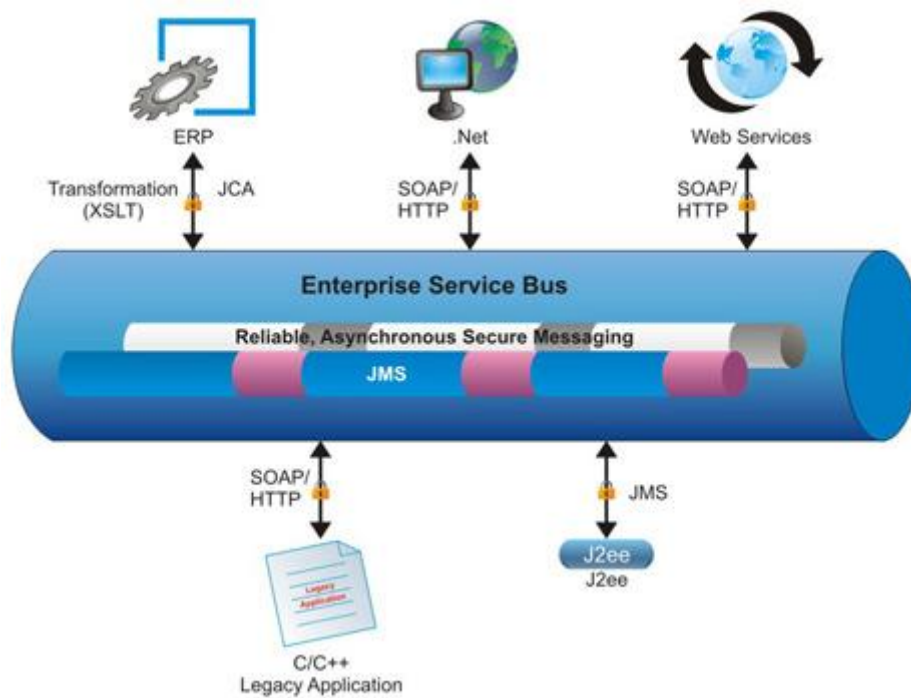


Figure 17: An example topology for ESB (please refer to [35])

The ESB hides the implementation details of one module from other components. All components send and receive messages to the ESB. The ESB is responsible for delivering the messages to the recipients in the appropriate format.

If one component is momentarily unavailable or is being replaced by a new one, the system is not affected. The ESB waits for a component to be available again in order to deliver the message.

3.5.1.1 Major ESB Solutions

A study on the available ESB implementations in the literature has been performed in order to come up with the appropriate solution to be adopted in Aniketos. The study has been based on certain criteria, which are mainly lie on the assumptions that the project should adopt a reliable open source solution, which is widely used in other commercial and R&D activities and can be directly integrated into the project, without any major need for specialised customisation.

Table 5 summarises the most common open source ESB implementations and their corresponding license scheme.

Table 5: Open source ESB implementations and their corresponding license

System	License	Comments
Mule ESB http://www.mulesoft.org/	CPAL (OSI-approved license, bundles other open source libraries that are made available under their respective licenses).	Most widespread used ESB system
Apache Service Mix http://servicemix.apache.org/home.html	Apache license	Includes JBI container, enterprise support available
Open ESB http://java.net/projects/open-esb/	CDDL-1.0	-
Petals ESB http://petals.ow2.org	LGPL 2.0	-
Swordfish http://www.eclipse.org/swordfish/	Eclipse Foundation Software User Agreement	Project in incubation phase

Given these solutions, the Mule ESB and Apache Service Mix have been selected in Aniketos to support the communication between the different components. The selection has been based on the following factors:

- Both Mule ESB and Apache Service Mix are open source popular solutions for an ESB implementation;
- They have extensible documentation and support communities in case the project faces any problems importing them into Aniketos;
- The exploitation of two popular ESB solutions can maximise the acceptance and independence of the Aniketos platform to specific ESB implementations.

3.5.1.2 Mule ESB

Mule ESB™ [36] is world's most widely used open source enterprise service bus. It has been downloaded millions of times and has thousands productions deployments. Mule ESB architecture is depicted in Figure 18. This lightweight architecture and simplified development model allows developers to easily create and integrate application services. Mule ESB takes the complexity out of integration, enabling developers to easily build high-performance, multi-protocol interactions between heterogeneous systems and services.

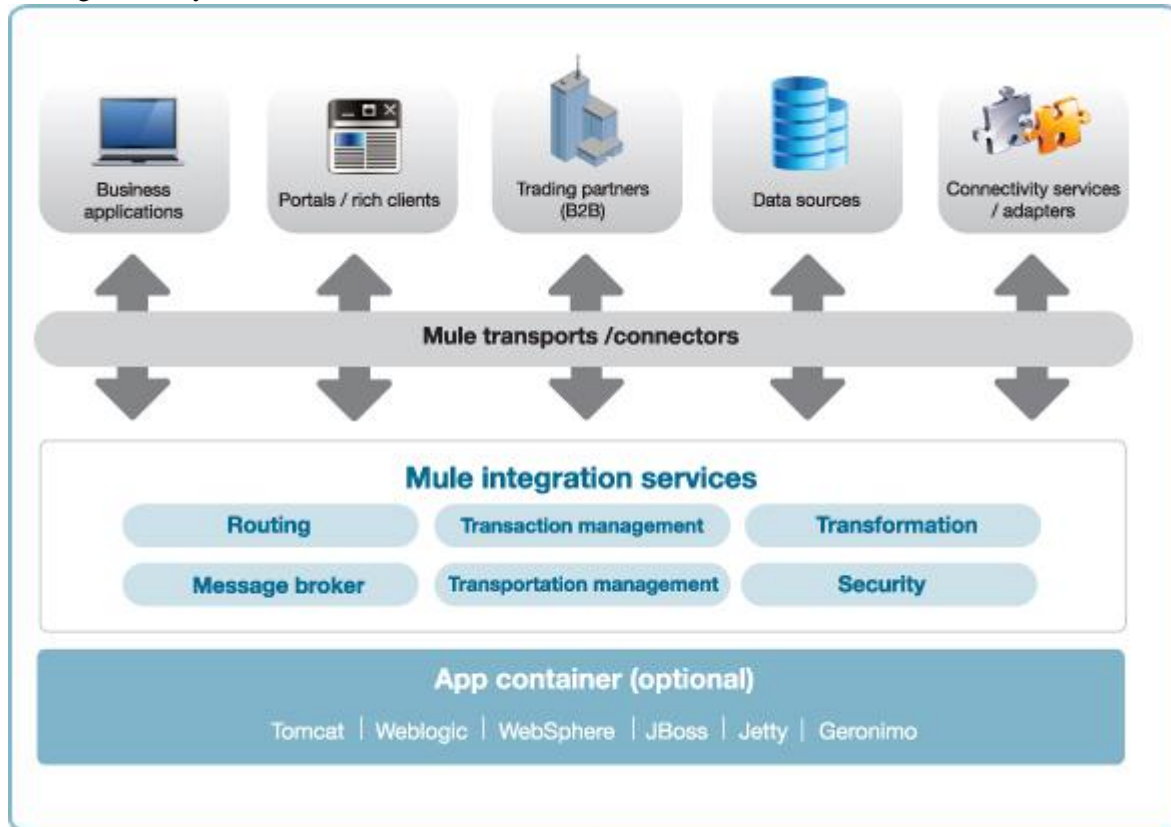


Figure 18: The Mule ESB Architecture (please refer to [36])

The main features of Mule ESB are:

- Service Mediation
 - Message Routing
 - Separate business logic from messaging
 - Shield services from message formats and protocols
 - Enable location-independent service calls
 - Provides protocol bridging
- Message Routing
 - Route messages based on content and complex rules
 - Filter, aggregate and re-sequence messages
- Data Transformation
 - Exchange data across applications with varying data formats
 - Manipulate message payload, including encryption, compression and encoding transformations

- Format messages across heterogeneous transport protocols data types
- Service creation & hosting
 - Expose end-points, EJBs, Spring beans, and Plain Old Java Objects (POJOs) as services
 - Host services as a lightweight service container

3.5.1.3 Apache ServiceMix

Apache ServiceMix [37] is an enterprise-class open-source distributed ESB and SOA toolkit. It was built from the ground up on the semantics and APIs of the Java Business Integration (JBI) specification JSR 208 and released under the Apache License.

The main features of Apache ServiceMix are:

- Reliable messaging with Apache ActiveMQ
- Messaging, routing and Enterprise Integration Patterns with Apache Camel
- WS-* and RESTful web services with Apache CXF
- Loosely coupled integration between all the other components with Apache ServiceMix NMR Including rich Event, Messaging and Audit API
- Complete WS-BPEL engine with Apache ODE
- OSGi-based server runtime powered by Apache Karaf

3.5.2 SOAP Web Services

SOAP [38], originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. SOAP exchanges messages in Extensible Mark-up Language (XML) format. It usually relies on other Application Layer protocols like Remote Procedure Call (RPC) and Hypertext Transfer Protocol (HTTP) for message negotiation and transmission.

A SOAP Web Service is formally described through a Web Services Description Language (WSDL) model, which is also based on XML. All major programming languages offer libraries tools for creating services and clients either from an existing WSDL definition (contract-first) or by code-first techniques.

SOAP is considered to be a part of WS-* specifications, which loosely refer to all specifications, associated with web services, and are maintained or supported by various standards bodies and entities. One big advantage of SOAP over older technologies is that since it can use HTTP as the transfer layer, it can tunnel easily over existing firewalls and proxies, without modifications to the SOAP protocol, and over the existing infrastructure. The formal definition, which WSDL model offers, makes the interoperability between systems built with different technologies easier.

However, because of the verbose XML format, SOAP can be considerably slower than competing middleware technologies such as CORBA. This may not be an issue when only small messages are sent.

3.5.3 REST Web Services

Representational State Transfer (REST) [39] is a style of software architecture for distributed hypermedia systems such as the World Wide Web. The term Representational State Transfer was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation. Conforming to the REST constraints is referred to as being 'RESTful'.

A RESTful web service (also called a RESTful web API) is a simple web service implemented using HTTP and the principles of REST. It is a collection of resources, with three defined aspects:

- The base URI for the web service, such as <http://example.com/resources/>

- The Internet media type of the data supported by the web service. This is often JSON, XML or YAML but can be any other valid Internet media type.
- The set of operations supported by the web service using HTTP methods (e.g., POST, GET, PUT or DELETE).

Unlike SOAP-based web services, there is no "official" standard for RESTful web services. This is because REST is an architecture, unlike SOAP, which is a protocol. Even though REST is not a standard, a RESTful implementation such as the Web can use standards like HTTP, URI, XML, etc.

RESTful web services are a key part of the "Web 2.0" momentum and are used by many applications and services. Developers prefer them over SOAP web services due to their simplicity and better performance. SOAP web services are used more often in enterprise application, usually in conjunction with other WS-* specifications that offer a formal model for security, workflow, notifications and other concepts.

3.6 Application Server Technologies

3.6.1 OSGi

The OSGi framework [40] is a module system and service platform for the Java programming language. The OSGi framework tries to overcome Java's inherent modularity limitations and to implement a complete and dynamic component model.

OSGi can be used in order to:

- Ensure that code dependencies are satisfied before allowing the code to execute and thus avoid `ClassNotFoundException`s;
- Verify that the code dependencies are consistent with respect to required versions and other constraints'
- Easily share classes between modules;
- Package an application as logically independent JAR files and be able to deploy only those pieces that are actually needed for a given installation;
- Declare which code is accessible from each JAR file and enforce the visibility (what is externally visible and what is not);
- Implement an extensibility (plug-in) mechanism;
- Start, stop, deploy, un-deploy and replace modules dynamically.

The OSGi Service Platform is composed of two parts: the OSGi framework and OSGi standard services. The framework is the runtime that implements and provides OSGi functionality. The standard services define reusable APIs for common tasks, such as Logging and Preferences.

The OSGi specifications for the framework and standard services are managed by the OSGi Alliance. The current version of the specification is release 4.2. Figure 19 depicts the architecture of the OSGi framework.

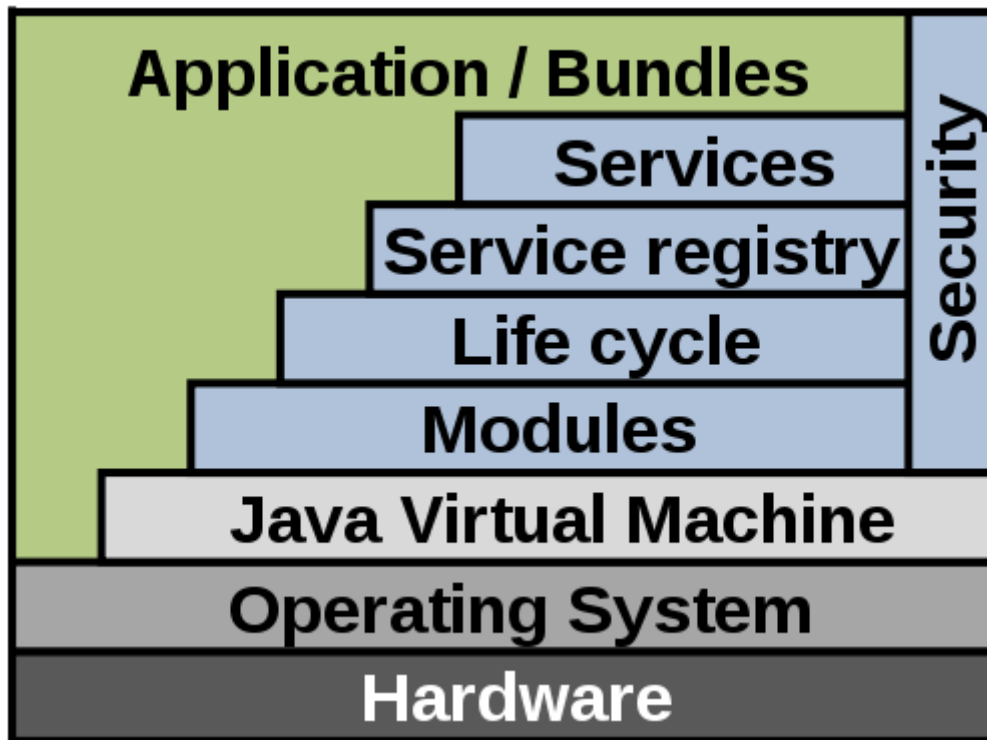


Figure 19: The OSGi Architecture (please refer to [40])

The OSGi framework consists of three horizontal layers:

- **Modules Layer:** The layer that defines encapsulation and declaration of dependencies (how a bundle can import and export code)
- **Life-Cycle Layer:** The API for life cycle management for (install, start, stop, update, and uninstall) bundles.
- **Services Layer:** The services layer connects bundles in a dynamic way by offering a publish-find-bind model for plain old Java Interfaces (POJI) or Plain Old Java Objects (POJO).

The Service Registry defines the API for managing services (ServiceRegistration, ServiceTracker and ServiceReference).

The OSGi framework also defines a Security layer that handles the security aspects by limiting bundle functionality to pre-defined capabilities.

3.6.1.1 Popular OSGi Containers

A study on the available OSGi Containers implementations in the literature has been performed in order to come up with the appropriate solution to be adopted in Aniketos. The study has been based on certain criteria, which are mainly lie on the assumptions that the project should adopt a reliable open source solution, which is widely used in other commercial and R&D activities and can be directly integrated into the project, without any major need for specialised customisation.

The available solutions examined are summarised here:

- **Apache Felix** - an implementation of the OSGi R4 Service Platform and other OSGi-related technologies
- **GlassFish (v3)** - application server for Java EE
- **JBoss** - Red Hat's JBoss Application Server
- **JOnAS 5** - open source Java EE 5 application server
- **OpenEJB** - open source OSGi-enabled EJB 3.0 container that can be run both in standalone or embedded mode

- **SpringSource dm Server** - open source microkernel-based server constructed of OSGi bundles and supporting OSGi applications

Given these solutions, the GlassFish Application Server has been selected in Aniketos, mainly due to the fact that [41]:

- It is Java EE certified;
- It bears high performance, as it is the fastest open source application server available;
- It is easy to be used;
- It integrated clustering support and sophisticated high-availability capabilities;
- It supports Web services;
- It provides multi-IDE support, while a standalone Oracle GlassFish Server plug-in for the Eclipse IDE is available.

3.6.1.2 Oracle Glassfish Application Server

GlassFish [42] is an open source application server project led by Sun Microsystems (owned by Oracle corporation) for the Java EE platform. The proprietary version is called Sun GlassFish Enterprise Server. GlassFish is free software, dual-licensed under two free software licences: the Common Development and Distribution License (CDDL) and the GNU General Public License (GPL) with the classpath exception.

The newest version of the server is GlassFish 3.1, which is based on world's first implementation of Java EE 6 with an OSGi based flexible, lightweight, extensible platform. It requires a small memory footprint. It is fully featured with production-ready features such as clustering and high availability provides optimized runtime performance and ready for enterprise deployments.

Main features include:

- Modular, Extensible and Innovative
- Provides dynamic invocation of services. Only initializes the required services reducing server footprint. One can start small with Web Profile and grow with their needs
- Oracle GlassFish Server 3.1 has ~262 modules
- End-to-end extensibility. Both server runtime and management interfaces are extensible
- Improved Developer Productivity
- Supports Java EE 6 standard based development with simplified programming model 29%
- Embedded GlassFish and maven plugin for unit testing
- Sophisticated tools integration with NetBeans and Eclipse
- Production Ready
- Improved clustering and high availability. 34% better high availability performance over GlassFish 2.1.1
- Integration with SSH for remote node management
- Preferred fail-over by load-balancer plugin
- Metro HA: Reliable messaging sequence failover, secure conversation session failover
- Improved automatic delegated transaction recovery
- Support for clustering of message queue brokers in embedded mode
- Supports wide range of platforms
- Advanced Management
- Centralized administration of clusters
- Improved DAS scalability (100 managed instances)
- RESTful API for management and monitoring
- Application versioning support
- Application scoped resources

- Improved JDBC monitoring such as statement leak detection and reclaim, tracing SQL queries, application based connection pool monitoring, custom validation template for JDBC connection pool

3.6.1.3 Apache Felix OSGi Container

Apache Felix [43] is a community effort to implement the OSGi R4 Service Platform and other interesting OSGi-related technologies under the Apache license. Felix and Equinox are considered to be the reference implementations of OSGi specifications.

The Felix project is organized into subprojects, where each subproject targets a specific OSGi specification or OSGi-related technology. Some of the more important Felix sub-projects are:

- Config Admin
- Dependency Manager
- Event Admin
- Framework
- Gogo
- HTTP
- iPOJO
- Log
- Maven Bundle Plugin
- Preferences
- Remote Shell
- Service Component Runtime (OSGi Declarative Services implementation)
- Web Console

Many enterprise application servers, like Glassfish, ServiceMix and JOnAs use Felix internally.

3.6.1.4 Conclusion

It is proposed that Glassfish application server will be used for the deployment of Aniketos runtime components. Felix may be used during development and testing phases.

3.7 Persistency and Information Storage Technologies

3.7.1 MySQL Database Server

MySQL [44] is considered to be the most popular object relational database system. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

Free-software-open source projects that require a full-featured database management system often use MySQL. MySQL is the M in the LAMP acronym, which stands for Linux-Apache-MySQL-PHP and refers to the open source technologies used by numerous frameworks and applications.

MySQL works on many different system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, Mac OS X, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris and others. It is a very mature system, very good documented and with a wealth of resources available. Many graphical interface applications for the management and administration of a MySQL database are available, both free and paid.

Although MySQL was the database of choice for almost all open source projects, the acquisition of Sun by Oracle, which brought MySQL under Oracle's control, made many developers sceptical about using it in new projects. Although Oracle promised to continue supporting the database system and keep offering the community edition, it is generally thought that a sudden policy change from Oracle's

part is not to be excluded. For this reason PostgreSQL is considered as an alternative for open source projects.

3.7.2 PostgreSQL Database System

PostgreSQL [45] is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It includes most SQL:2008 data types. It also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, among others, and exceptional documentation.

PostgreSQL SQL implementation strongly conforms to the ANSI-SQL:2008 standard. It has full support for subqueries (including subselects in the FROM clause), read-committed and serializable transaction isolation levels. And while PostgreSQL has a fully relational system catalogue, which itself supports multiple schemas per database, its catalogue is also accessible through the Information Schema as defined in the SQL standard

It is released under a liberal open source license: the PostgreSQL License (MIT-style license) and is thus free and open source software. As with many other open-source programs, PostgreSQL is not controlled by any single company — a global community of developers and companies develops the system.

3.7.3 MongoDB

MongoDB [46] is a popular NoSQL database solution. NoSQL refers to a broad class of database management systems that differ from classic relational database management systems (RDBMSes) in some significant way. These data stores may not require fixed table schemas, usually avoid join operations and typically scale horizontally. NoSQL databases provide no SQL interface and usually rely on much simpler interfaces that use associate arrays or key-value pairs. Besides their simplicity, a big advantage of many NoSQL technologies is that they use a distributed architecture, which allows them to easily be deployed in a cloud system. A NoSQL system can be deployed in many servers and a failure of a server can be tolerated.

MongoDB is a popular open source, high performance, schema-free database written in C++ programming language. The database is document-oriented and manages a collection of JSON-like documents. MongoDB can run in almost all operating systems and binaries are available for Windows, Linux, OS X and Solaris.

MongoDB uses JavaScript for making queries. It offers official drivers for all popular programming languages (C, C++, C#, Haskell, Java, Javascript, Perl, PHP, Python, Ruby, Scala).

Some of the advantages of MongoDB are summarised in the following:

- It is easy to be installed;
- A PHP module is available;
- Replication is very easy, including master-master support
- It supports automated sharding
- It is well documented

3.7.4 Conclusion

It is proposed that the Aniketos Components, bearing a persistent layer need, utilise MySQL database server to implement it. However, it should be avoided to use features that are specific to MySQL. Also, the use of an ORM (Object Relational Mapping) tool, like Hibernate [47], is recommended.

4 Description of the Aniketos components

This section describes the individual components of the Aniketos architecture splitting them according to the categorisation of D1.2.

For each component, the following aspects are presented, where this is applicable:

- The general description/specification of the components
- The limitations and restrictions of each component with respect to H/W, S/W and Operating System specification
- The dependencies and/or required interfaces, as well as a detailed description of the data model for the envisaged interfaces between the Aniketos components
- Component-specific implementation aspects (which are summarised for the whole set of the Aniketos platform and Environment Components)

It must be noted that the descriptions provided in the rest of this section follow the presentation of the Aniketos components are reflected on section 2, but they are aligned with the Aniketos Deliverable D1.2. They present an initial approach to the interfaces specifications, which have to be implemented as a first step towards a conceptual prototype. In that respect, these specifications are subject to continuous changes and updates.

4.1 Aniketos Platform components

4.1.1 Socio-technical Security Modelling Tool

Description

The Socio-technical Security (s-t-s) Modelling Tool allows the service developer to specify the service security requirements following a two-step process:

- Identify, analyse and model the goals, the actors, the resources, the socio-technical relationships (i.e. commitments, trust, etc.) and the security needs of the organisation
- Derive and model the needed services and the security and trust requirements on them (create the Security Specification Model of the composed service)

The tool consists of a graphical modelling environment that supports the socio-technical security language developed within Aniketos. The Socio-technical Security Modelling Tool receives the service specifications in the order to model the service from the Service specification/planning mechanism of the Environment. The tool looks up for threats and countermeasures in the Threat repository module to describe in the service model both the threats associated to the service and the possible countermeasures to be implemented in the service.

Limitations and Restrictions

The s-t-s Modelling Tool raises no restrictions.

Interfaces

The s-t-s Modelling Tool exposes the following methods for the *IModelling* interface:

Method Name	<i>modelTrustAndSecurityRequirements</i>
Method Definition/Description	This method creates the model of the service that includes the socio-technical aspects and the trust and security requirements.
Method input attributes	

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceSpecification</i>	ServiceSpecification	Specification of a composed/component service, or the planning or template creation for service composition.
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>SecuritySpecificationModel</i>	SecuritySpecificationModel	Specification of the service including trust and security requirements.
Dependency with other components	Socio-technical Security Modelling Tool creates the <i>SecuritySpecificationModel</i> . The tool starts working when it receives the invocation from the Service Composition Framework in the Environment. The Socio-technical Security Modelling Tool is able to retrieve specific threats and countermeasures from the Threat Repository Module. This method can be seen as internal to the s-t-s Modelling Tool.	

Method Name	<i>exportSecuritySpecificationModel</i>	
Method Definition/Description	This method exports the model of a specific service	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>SecuritySpecificationModel</i>	SecuritySpecificationModel	Specification of the service including trust and security requirements.
Dependency with other components	Socio-technical Security Modelling Tool exports the <i>SecuritySpecificationModel</i> , which is the input to the Model Transformation module.	

4.1.2 Model Transformation Module

Description

The Model Transformation Module transforms the service model described through the s-t-s modelling language (*SecuritySpecificationModel*) into the different formats needed from the other components of the platform, as follows:

- It transforms the *SecuritySpecificationModel* to the *AniketosCompliantSpecification* (which is a specialised service specification for deploying services being extended with security and trust characteristics, containing information about the service composition plan and the security requirements). The *AniketosCompliantSpecification* is then used by the Service Composition Framework component, the Contract Manager module and the service consumers.

- It takes the *AniketosCompliantSpecification* and generates the *SecurityDescriptor*, which provides information about security properties for a registered service and is exploited by the Aniketos Marketplace module.
- It generates the *ContractTemplate* (a template used to create security contract at design time or at runtime) from the *AniketosCompliantSpecification* that is used by the Contract Manager module.
- It binds the service to its threats through the *Binding:Threat-Service* method that invokes the *bindServiceToThreatMonitoring* method of the Service Threat Monitoring module, which in turn registers a service with the Notification module for alerts.

Limitations and Restrictions

The Model Transformation Module will be based on Java, as it is developed as an OSGI bundle.

Interfaces

The Model Transformation Module exposes the following four methods for the *ModelTransformationModuleService*:

Method Name	<i>transformModel</i>	
Method Definition/Description	This method transforms the <i>SecuritySpecificationModel</i> of the service into an <i>AniketosCompliantSpecification</i>	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>SecuritySpecificationModel</i>	SecuritySpecificationModel	Service socio-technical model including security information
<i>ServiceSpecification</i>	ServiceSpecification	It contains information about service composition plan
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>AniketosCompliantSpecification</i>	AniketosCompliantSpecification	Specialised 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
Dependency with other components	The Model Transformation Module receives the <i>SecuritySpecificationModel</i> from Socio-technical Security Modelling tool. The produced <i>AniketosCompliantSpecification</i> is exploited by the Contract Manager module and Service Composition Framework.	

Method Name	<i>generateSecurityDescriptor</i>	
Method Definition/Description	This method generates the security descriptors for a specific service, based on the <i>AniketosCompliantSpecification</i>	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>AniketosCompliantSpecification</i>	AniketosCompliantSpecification	Specialised 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
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>SecurityDescriptor</i>	SecurityDescriptor	provides information about security properties for a registered service
Dependency with other components	Security Descriptor is needed by the Marketplace module and Service composition framework.	

Method Name	<i>generateContractTemplate</i>	
Method Definition/Description	This method generates the contract template for a specific service, based on the <i>AniketosCompliantSpecification</i>	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>AniketosCompliantSpecification</i>	AniketosCompliantSpecification	Specialised 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
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ContractTemplate</i>	ContractTemplate	A template used to create security contract at design time or at runtime
Dependency with other components	<i>ContractTemplate</i> method is to interact with the Contract Manager Module	

Method Name	<i>bindServiceThreat</i>	
Method Definition/Description	This method links the service with the threats associated to it that are documented in the <i>AniketosCompliantSpecification</i> .	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>AniketosCompliantSpecification</i>	AniketosCompliantSpecification	Specialised 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
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>

<i>bindOperationResult</i>	boolean	Information about binding operation. If the binding operation has finished successfully is true; otherwise it's false.
<i>counterMID</i>	String	Optional: The unique identifier of monitoring control to which the service subscribes
<i>counterMDesc</i>	String	Optional: Description of the countermeasure, either in plain text or as a pattern.
Dependency with other components	The <i>bindServiceThreat()</i> method is used by the Notification Module and invokes the <i>bindServiceToThreat()</i> method of the Service Threat Monitoring module (see Section 4.1.9). This method is mainly exploited at design time to bind relevant threats to the specification. When doing the transformation, this binding needs to be stored between the threat and the actual running service instances.	

4.1.3 Trustworthiness Component

Description

The Trustworthiness Component is a physical component, which integrates the functionalities of two logical modules, as presented in D1.2, namely the Trustworthiness Prediction and Trustworthiness Monitoring modules.

The main functionalities of this component are:

- Provide a trust level to represent service trustworthiness
- Compute the trustworthiness level of a composite service
- Store trust values
- Represent the identity of contractual parties
- Recalculate trustworthiness when relevant events occur
- Respond to changes in trust levels (e.g. lower trustworthiness)

Limitations and Restrictions

The Trustworthiness Component raises no limitations and restrictions

Interfaces

The Trustworthiness Component exposes the following three methods for the *TrustworthinessService* interface:

Method Name	<i>getTrustworthinessPrediction</i>	
Method Definition/Description	This method returns the trustworthiness of a service (either complex or atomic)	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
Method output attributes		

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>Trustworthiness</i>	Real	The trustworthiness value of the specific service
Dependency with other components	This method is invoked by the Verification Module, the Contract Manager Module and the Marketplace in order to receive the trustworthiness value of the requested (complex or atomic) service.	

Method Name	<i>receiveNotification</i>	
Method Definition/Description	This method receives notifications about possible changes in trust values.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>notification</i>	String	A string detailing the type of notification
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method is triggered by the Notification module	

Method Name	<i>receiveQoSupdate</i>	
Method Definition/Description	This method receives updates with respect to quality of service metrics for a specific service	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>metrics</i>	collection	The identification of service specification in the Marketplace
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>qos_update</i>	List	The list of updated QoS parameters
Dependency with other components	This method is invoked by the Service Runtime Environment	

Method Name	<i>receiveReputationUpdate</i>	
Method Definition/Description	This method receives receives updated reputation	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>

<i>reputation</i>	double	The value of the received reputation
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method is invoked by the Service Runtime Environment	

The Trustworthiness Component exposes the following two methods for the *MonitoringServiceAccess* interface:

Method Name	<i>sendEventToMonitor</i>	
Method Definition/Description	This method is used to send event to be monitored from the Trustworthiness Component to the monitoring system that will check the rule.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>monitorAddress</i>	String	Address of the monitoring system
<i>userName</i>	String	Credential information
<i>password</i>	String	Credential information
<i>EventInput</i>	String	Event to be sent
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>monitorResult</i>	Boolean	The result on whether the event means any violation
Dependency with other components	This method enables the Trustworthiness Component to communicate with the external environment Service Monitoring Component	

Method Name	<i>deriveMonitoringRules</i>	
Method Definition/Description	This method receives security contract (or a part of the contract related to trust properties) and derives monitoring rules.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>Security contract</i>	String	The identification of service specification in the Marketplace
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>rules</i>	List	The list of rules
Dependency with other components	This method is triggered by the Service Monitoring Component and the Environment components, which provide signed contracts	

4.1.4 Verification module

The Verification Module can be seen as one physical component, which integrates the functionalities of two logical modules, namely the Contract Negotiation Module and Security Verification Module. The Verification Module is split into the following three sub-modules:

- Contract Manager Module (CMM)
- Property Verification Module (PVM)
- Composition Security Validation Module (CSVM)

In the following sections, these sub-modules and their interfaces are described separately.

4.1.4.1 Contract Manager Module

Description

The main functionalities of the CMM are:

- It manages the overall security checking process
- It checks the compliance of agreement template from a service provider with the consumer's security policy
- It supports the runtime usage of security contracts by:
 - Deriving the monitoring rules from security contracts
 - Updating the status of contracts based on relevant events
 - Responding to contract violations identified in the environment monitoring service

The CMM is a physical component, which implements part of the functionalities allocated to the logical component of the Security Verification Module, which is defined in D1.2.

Limitations and Restrictions

The CMM raises no limitations and restrictions

Interfaces

The CMM exposes the following four methods for the *ContractManagerService* interface:

Method Name	<i>AnalyseSecureComposition</i>	
Method Definition/Description	This method manages all security and trust checks. In fact, only mapping of service provider agreement template and service consumer security policies is performed by the module itself. Check of security and trust properties of provider are performed by Property Verification Module, the Trustworthiness Component and the Threat response recommendation module (all are invoked by the CMM)	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>AgreementTemplate</i>	String	Agreement template of service provider
<i>ConsumerPolicies</i>	String	Service consumer security policies
<i>SecureCompositionPlan</i>	String	Composition Plan for a complex service

Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>CheckResults</i>	String	Results of all check
Dependency with other components	This method is invoked by the Secure Composition Planner Module	

Method Name	<i>getTrustworthiness</i>	
Method Definition/Description	This method serves with the need for getting the trustworthiness prediction specification for a specific service.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>TrustworthinessPredictionResponse</i>	Real	The return value defining the trustworthiness of the requested service
Dependency with other components	Through this method, the CMM acts as a proxy between the Marketplace and the Trustworthiness component.	

Method Name	<i>determineSecurityProperties</i>	
Method Definition/Description	This method serves the determination whether the security properties are compliant with the specifications and the contracts	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>AniketosCompliantSpecification</i>	AniketosCompliantSpecification	Specialised 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
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>property</i>	PropertyID	An identifier that specifies the required property
Dependency with other components	Through this method, the CMM acts as a proxy between the Service Runtime Environment and the SPDM	

Method Name	<i>checkCountermeasuresImplementation</i>	
Method Definition/Description	This method serves the requirement for checking at runtime the compliance with the countermeasures of specific threats.	

Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>threatDesc</i>	List(String)	List with the description of the threats in plain text
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>(counterMID, nonComplianceTwSpecDesc)</i>	List (Integer, String)	A list of non-compliant countermeasures, i.e. countermeasures which are not implemented by the selected services. If the list is null, then the service is compliant to its specification. <i>CounterMID</i> is the unique identifier for the countermeasure not implemented. <i>nonComplianceTwSpecDesc</i> is the description of the non-compliance for a given <i>CounterMID</i>
Dependency with other components	Through this method, the CMM acts as a proxy between the Service Runtime Environment and the Threat Response Recommendation Module	

4.1.4.2 Property Verification Module

Description

The Property Verification Module (PVM) analyses a service implementation (e.g., based on its source code) for compliance with required security properties (e.g., absence of certain vulnerabilities, enforcement of access control, ensuring data privacy) as expressed in the contract.

Limitations and Restrictions

The PVM raises the following limitations and restrictions:

- The H/W limitations and requirements still have to be discussed and decided, but the formal analysis may require significant CPU and memory resources
- The S/W limitations and requirements still have to be discussed and decided, but Scala (on the Java platform) or F# (on mono/.net platform) are considered
- The PVM is planned to be platform independent (i.e., support both Windows and Linux on an x86 or AMD64 architecture), but it still has to be discussed and decided

Interfaces

The PVM exposes the following method for the *PropertyVerificationService* interface:

Method Name	<i>verifyTechnicalTrustProperties</i>	
Method Definition/Description	This method checks if a service implementation fulfils a certain security/trust property	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>AgreementTemplate</i>	String	Agreement template of service provider
<i>ServiceImplementation</i>	Class/URL	Archive of the implementation (alternatively, URL to the archive)
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>Verification Result</i>	Boolean	True or false
<i>Explanation</i>	Class/String	Explanation of the verification result
Dependency with other components	This is triggered by the CMM to provide responses to other components on the property verification results.	

4.1.4.3 Composition Security Validation Module

Description

The Composition Security Validation Module (CSVM) verifies the service compliance to agreement templates both at design-time and at runtime. During the design-time, this module uses inputs from the Trustworthiness Component (and more specifically the trustworthiness prediction functionality), the Security Property Determination Module (SPDM), the Threat response recommendation module and the Secure Composition Planner Module (through the CMM). Its output is returned to the CMM. At runtime, the CSVM is triggered through the CMM to verify the result of a secure re-composition.

Limitations and Restrictions

The CSVM raises the following limitations and restrictions:

- The H/W limitations and requirements still have to be discussed and decided, but the formal analysis may require significant CPU and memory resources
- The S/W limitations and requirements still have to be discussed and decided, but Scala (on the Java platform) or F# (on mono/.net platform) are considered
- The PVM is planned to be platform independent (i.e., support both Windows and Linux on an x86 or AMD64 architecture), but it still has to be discussed and decided

Interfaces

The CSVM exposes the following method for the *CompositionSecurityValidationService* interface:

Method Name	<i>VerifyCompositionCompliance</i>
Method Definition/Description	This method checks if the security requirements described in the agreement template of a service match the secure composition plan
Method input attributes	

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>Composition plan</i>	Class	This is the secure composition plan which composes the services
<i>Agreement Template</i>	Class	These are the specifications for a composite service the provider offers
<i>Set of Policies</i>	Class	Set of requirements of the user to a single service
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>Verification Result</i>	Boolean	True or false
<i>Explanation</i>	Class/String	Explanation of the verification result
Dependency with other components	This method is triggered by the CMM to provide responses to other components on the composition verification results.	

4.1.5 Security Property Determination Module

Description

The Security Property Determination Module (SPDM) is responsible for managing the security properties associated with a service (whether the service is atomic or itself a composite of subservices). It interacts with the Verification Module (and the CMM) and the properties associated with services in the Marketplace. Security properties can be absolute binary values, or ranges that represent predicted values.

The dependency direction between security property determination and security verification is also discussed in this document (i.e. which module serves which), which may result in an exchange of functionality from one module to another as appropriate.

The primary purpose of the Security Property Determination Module is to provide security state for WP3 and by implication, to the Aniketos platform. To this end its functions include the following:

- Retrieval of security properties for services from the marketplace;
- Determination of security properties of composed services;
- Retrieval of trust prediction (as an interface to the trust values from the Trustworthiness Prediction Module in case trust requirements appear in contracts);
- Predicting access control performance for composed services;
- Labelling services with relevant security properties for composition;
- Making determinations as to what the overall security properties for a composed service are;
- Reconciling any differences (a) between a given composed service and its component services and (b) between claimed values of inferred properties (such as trust and access control performance) and the corresponding accepted values of these inferred properties;
- Storage and maintenance of individual service security properties (necessary for Composition Planning).

Note the following exclusions, as they occur elsewhere in WP3 and WP4.

- The module does not perform any monitoring;
- The module does not perform any verification.

The initial documentation D1.2 [1] describes an interface that is capable of determining security properties for both a single service and a service composition; in fact we handle these in the same way, since a single service may itself simply be a composition of other services.

The module API offers getter and setter methods for security properties. For each of these a single property is required, specified by a *property identifier*, which alters the interface accordingly. The module also offers predictive capabilities of its own, notably with access control performance prediction.

There are two notable areas of functionality that are described below.

- **Property Verification** – Aniketos requires that a given service’s stated security property (Contract/Template Agreement) can be verified against its actual functionality. It is expected that this functionality will be handled by the verification module (see below), such that the module can be asked, “does service x fulfil property y as stated?”

The verification module will conduct formal analysis, behaviour observation, or any appropriate verification technique and return a result, signed and authenticated that is stored alongside the property in the Marketplace. The property determination module will check to see if the property has a current, signed certificate, and if so, simply provide the requested property from the Marketplace. If not, the property determination module will request that the verification module verifies the property and return the result.

- **Composed Property Aggregation / Agreement / Calculation** – The property determination module need only check the Marketplace (as above) for a single service. However, for a composed service, the property must be calculated based on the properties of the subservices of which it’s comprised. It is the responsibility of the Property Determination Module to iterate through the subservices in order to determine the properties of the overall composition. This process may therefore involve re-entrant recursive calls to the Property Determination module. Two illustrative examples are as follows.

- Available Encryption property for a Composition containing two services A and B. A has [AES-128, AES-256, Twofish], while B has [AES-128, AES-256, RC5]. In this case, the composed property is simply the intersection of all the composed service’s property values [AES-128, AES-256].
- Availability property for a Composition containing two services A and B. A has availability = 90%, whereas B has availability = 75%. The composition property is 75%, as it is in this case the lowest value / weakest link.

There is a dependency between this module and the security verification module, required so that a stated security property can be verified as an actual security property. However, it is unclear as to whether the verification module expects this functionality of the property determination module, or vice versa. This document assumes that this module obtains this verification functionality from the verification module, but this requires clarification.

However, there are some areas where properties may be determined directly by the Security Property Determination Module, as is the case for Access Control Performance Prediction. STACS (please refer to the Aniketos Deliverable D1.1) predicts access control performance for a given composition and should be part of the pre-processing in the Security Property Determination Module. Services and compositions should be tagged with the results of these predictions (along with trust from WP2) as security properties.

Prediction is done by modelling access request profiles for two or more composed services. The response times of access control requests depend on the composed request profiles and how the service components interact with (hence access) each other. These interactions can be defined by the composition planner, e.g. by means of a dependency graph. Access request profiles can be defined by analysing the typical use cases. STACS can then analyse, using discrete event simulation, based on the multiple queues representing the access control mechanisms of interdependent services in the

composition. With respect to contracts, access control performance can be modelled as a particular type of queue and the necessary parameters should be given in the marketplace as part of the contract.

There remains an open question on the naming techniques with respect to how services and their providers will be referenced. We believe this falls within the scope for Aniketos and should not be left up to the marketplace or environment as it has immediate impact on how we implement modules and define APIs. It will also have a strong influence on how WPs interact. UDDI has been mentioned previously, but agreement has yet to be made.

Limitations and Restrictions

The Security Property Determination Module raises no limitations and restrictions

Interfaces

The Security Property Determination Module exposes the following five methods for the *IService* interface:

Method Name	<i>getService</i>	
Method Definition/Description	This method receives the specified service from the relevant repository.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>serviceID</i>	ServiceID	The ID of the specified service
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>Service</i>	IService	The specified service
Dependency with other components	The SPDM uses this method to interact with the Marketplace and get services	

Method Name	<i>addService</i>	
Method Definition/Description	This method adds the specified service to the relevant repository.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>Service</i>	IService	The specified service
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	Service ID	The MarketPlace should return the ID of the specified service.
Dependency with other components	The SPDM uses this method to interact with the Marketplace and add services to it	

Method Name	<i>addProperty</i>	
Method Definition/Description	This method is used to attach properties to a service specification	

Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>property</i>	PropertyID	An identifier that specifies the required property
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	The SPDM interacts with the Marketplace through this method	

Method Name	<i>getValue</i>	
Method Definition/Description	This method gets the value of a security property	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>property</i>	PropertyID	An identifier that specifies the property
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>value</i>	String	The value of the verified property to be stored.
Dependency with other components	The Security Property Determination Module interacts with the Marketplace through this method	

Method Name	<i>setValue</i>	
Method Definition/Description	This method sets the value of a security property	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>property</i>	PropertyID	An identifier that specifies the property
<i>value</i>	String	The value of the verified property to be stored.
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		

Dependency with other components	The Security Property Determination Module interacts with the Marketplace through this method
---	---

The Security Property Determination Module exposes the following five methods for the *IProperty* interface

Method Name	<i>getProperty</i>	
Method Definition/Description	This method is used to get properties of a service specification	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>propertyID</i>	String	An identifier that specifies the required property
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>value</i>	String	The value of the verified property to be stored.
<i>freshness</i>	Date	The date of the value attached to the property
Dependency with other components	The SPDM interacts with the Marketplace and the CMM through this method	

4.1.6 Secure Composition Planner Module

Description

The Secure Composition Planner Module (SCPM) creates one or more suggestions based on a given composition plan's security features. It makes use of the Trustworthiness Component (and Trustworthiness Prediction functionality routed via Security Property Determination Module) and the SPDM itself to accomplish this. The following points are assumed:

- The SCPM receives composition plans from the environment (service specification/planning mechanism) which already comply with the functional aspects of the consumer's needs (user needs). The SCPM then analyses the service compositions based on trustworthiness and security properties.
- The SCPM uses trustworthiness indirectly from the Trustworthiness Prediction Module and security properties from the SPDM. The SPDM determines the security properties of the composed services and selects those that comply with the consumer security policy.

Three usage scenarios are considered that demonstrate the functionality of this module as follows.

1. The first scenario presents a simple situation where the SCPM is required to select only those composition plans that fulfil the consumer's security requirements from a given set of functionally-correct composition plans. This behaviour is referred to as *selectSecureCompositions*, and requires an input set of functionally-correct composition plans, along with the consumer's stated security policy. It returns only those functionally-correct compositions that adhere to the consumer's security policy, checking the consumer's policy against each service's stated properties for potential contract fulfilment – using trustworthiness where applicable. This approach suffers from the problem that the return set may be empty (*i.e.* no composition, as-is, may fulfil the security requirements) and does not distinguish between return sets (*i.e.* they are all

either compatible or not, no preference is given to any security heuristic). However, one benefit of this approach is that it could be implemented simply by passing each composition to the Security Verification Module (SVM), and returning compositions based purely on the results that this provides.

2. The second scenario extends the first situation to allow the resulting set to be annotated to describe a preference or ordering of the successfully return composition set. This behaviour is referred to as `orderSecureCompositions` and requires the input data from `selectSecureCompositions` with the addition of a statement of ordering/preference. This statement may be as simple as specifying a type from an enumeration of available orderings that have globally-understood meaning across the Aniketos platform. Alternatively this may interpret a policy statement as an indication of preference. The resulting set may simply be ordered by preference, or in the case of multiple weighted preferences, may return a scoring matrix indicating how each composition is ordered with respect to each preference.
3. The third scenario extends these scenarios to allow the SCPM to analyse a functionally-complete composition plan (as above), and if it finds it to be non-compliant with the consumer's security policy, rather than just omit it from the return set, it can suggest alterations that make it compliant. This behaviour is referred to as `suggestSecureCompositions`. The level of functionality expected from the suggestion process is still under discussion, but some possibilities include:
 - a. Alternative determination via Marketplace – Marketplace is queried for alternative services/compositions with specified security properties.
 - b. Use of techniques that propose alterations to services or compositions to include the additional security properties, for example using evolutionary processes or pre-defined patterns [48].

Regarding the interaction of this module and the relationship with other components, the following should be noted:

- The SCPM generates (or selects) one or more composition plans based on trustworthiness and the specified security properties. One or more of the specified properties could be related to each composition plan and used to order the composition plans in a number of ways. For example, this could be based on the most secure composition plans or the most trustworthy composition plans.
- The Security Property Determination Module is responsible for providing properties associated with the composed services.
- The Security Verification Module verifies that a given composition plan is sound according to the given security properties.
- The Threat Response Recommendation Module (from WP4) could provide input to the SCPM in terms of risks, vulnerabilities and threat mitigation. The Threat Response Recommendation Module analyses the composition plans from the SCPM for risks and vulnerabilities. The Threat Response Recommendation Module queries the Threat Repository to find a good pattern for the solution. The recommendations from the Threat Response Recommendation Module could be introduced as additional security properties for the required composition, or as an input to the ordering. The SCPM considers these additional security properties recommended by the Threat Response Recommendation Module in its process of generating secure composition plans.

Limitations and Restrictions

The Secure Composition Planner Module raises no limitations and restrictions

Interfaces

The Secure Composition Planner Module exposes the following three methods for the *CompositionPlannerInterface* interface:

Method Name	
	<i>selectSecureCompositions</i>

Method Definition/Description	This method selects those compositions from the specified list that adhere to the specified user policy	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>consumerPolicy</i>	Policy	A specification of the consumer's security policy
<i>functionalCompositions</i>	List< CompositionPlan >	A set of alternative service compositions that are functionally correct
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceCompositions</i>	List< CompositionPlan >	A set containing service compositions (a subset of functional Compositions) that are functionally correct and consistent with the user's security policy.
Dependency with other components	-	

Method Name	<i>orderSecureCompositions</i>	
Method Definition/Description	This method selects those compositions from the specified list that adhere to the specified user policy, and orders those that do according to the specified Ranking Criteria	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>consumerPolicy</i>	Policy	A specification of the consumer's security policy
<i>secureCompositions</i>	List< CompositionPlan >	A set of alternative service compositions that are functionally correct
<i>order</i>	OrderCriteria	An indication of how the resulting set should be ordered
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceCompositions</i>	List< CompositionPlan >	An ordered set of service compositions (a subset of functional Compositions) that are functionally correct and consistent with the user's security policy, and ordered according to the specified rank
<i>ScoringMatrix</i>	Array	A structure describing how each "secure" composition scores according to the ranking criteria/criterion.

Dependency with other components	-	
Method Name	<i>suggestSecureComposition</i>	
Method Definition/Description	This method generates a set of new composition plans as suggestions that adhere to the specified user policy	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>consumerPolicy</i>	Policy	A specification of the consumer's security policy
<i>functionalComposition</i>	CompositionPlan	A service composition that is functionally correct
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
ServiceComposition	List< CompositionPlan >	A set (possibly empty) containing new service compositions that are consistent with the user's security policy. This may include service compositions that differ from the input composition in terms of security properties (and potentially functional properties too)
Dependency with other components	This method enables the Secure Composition Planner Module to interact with the Service Composition Framework, the Service Runtime Environment, the Security Property Determination Module and the Trustworthiness Component.	

4.1.7 Security Policy Monitoring Module

Description

The Security Policy Monitoring Module facilitates for the following main functionalities:

- It supports runtime usage of security contracts
- It derives monitoring rules from security contracts
- It updates the contracts status based on relevant events
- It responds to contract violations

Limitations and Restrictions

The Security Policy Monitoring Module raises no limitations and restrictions.

Interfaces

The Security Policy Monitoring Module exposes the following two methods for the *ISecurityPolicyMonitoring* interface:

Method Name	<i>getContract</i>
--------------------	--------------------

Method Definition/Description	This method gets agreed Contract and derives monitoring rules from this contract	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>Contract</i>	String	Agreed Contract
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method is invoked by the Aniketos Environment components to get the contract attributes/rules	

Method Name	<i>getRealData</i>	
Method Definition/Description	This method gets the real data required for monitoring from the Service Runtime Environment. The real data are then used for monitoring checks.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>RealData</i>	String	Real data required for monitoring
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method is invoked by Service Runtime Environment.	

4.1.8 Threat Response Recommendation Module

Description

The Threat Response Recommendation Module works mainly during the validation phase. When it is invoked by the CMM, it checks for the compliance of a service towards the contract, and recommends for re-composition or reconfiguration if needed.

At design-time, during service validation, the CMM performs a verification of the secure composition sent by the Secure Composition Planner Module to the Service Composition Framework of the Environment. The Threat Response Recommendation Module is invoked by the CMM (which acts as proxy) at that stage, before the contract negotiation. The only possible checking at that stage is to verify that the selected services as composed by the Secure Composition Planner Module comply with the specification.

At run-time, during the service validation, a *validateService()* is sent by the (re) composition agent to the Service Runtime Environment, which is forwarded as an *analyseSecureComposition()* to the CMM for service validation. Still during run-time, any change or threat triggers an *analyseSecureComposition()* sent to the CMM for a service validation. The Threat Response Recommendation Module is invoked by the CMM.

At run-time, for both service validation and service re-composition, the CMM checks the compliance of the composed service towards the contract. Regarding the threats, it means that the running service, as known through the monitoring controls, should comply with the patterns, the policies and the

monitoring controls stored in the contract. The Threat Response Recommendation Module gets the Service ID from the CMM, collects all information corresponding to this ID from the logs, gets the contracts from the Marketplace, and checks then the compliance of the service to the contract.

There are two main functionalities handled by the Threat Response Recommendation Module:

- Check the countermeasures of selected services towards the secure specification. These functionalities enable the users of the community to add, update, or delete the threats and associated countermeasures in the repository.
- Check the actual countermeasures of the running service at run-time towards the contract. These functionalities enable the users of the community at design-time and the Threat response recommendation module at run-time to get the threats and associated countermeasures in the repository.

Limitations and Restrictions

The Threat Response Recommendation Module raises no limitations and restrictions.

Interfaces

The Threat Response Recommendation Module exposes the following two methods for the *IThreatResponseRecomm* interface:

Method Name	<i>checkCountermeasuresImplemTwSpec</i>	
Method Definition/Description	This method checks the compliance of a service to the countermeasures specified in the Security specification. At that stage, only patterns and some of the security policies can be checked, i.e. countermeasures where counterMType is set to pattern or policy.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>(counterMID, nonComplianceTwSpecDesc)</i>	List (Integer, String)	A list of non-compliant countermeasures, i.e. countermeasures which are not implemented by the selected services. If the list is null, then the service is compliant to its specification. <i>CounterMID</i> is the unique identifier for the countermeasure not implemented. <i>nonComplianceTwSpecDesc</i> is the description of the non-compliance for a given <i>CounterMID</i>
Dependency with other components	This method serves the CMM. It needs to send requests to the Marketplace for service discovery and a request to the Threat	

	Repository Module for getting the countermeasures.	
Method Name	<i>checkCountermeasuresImplemTwContract</i>	
Method Definition/Description	This method checks the compliance of a service to the countermeasures specified in the Security specification. At that stage, all the patterns, security policies and monitoring controls can be checked	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>(counterMID, nonComplianceTwSpecDesc)</i>	List (Integer, String)	A list of non-compliant countermeasures, i.e. countermeasures which are not implemented by the selected services. If the list is null, then the service is compliant to its specification. <i>CounterMID</i> is the unique identifier for the countermeasure not implemented. <i>nonComplianceTwSpecDesc</i> is a description of the non-compliance for a given <i>CounterMID</i> .
Dependency with other components	This method serves the CMM. It needs to send requests to the Marketplace for service discovery and for getting the service descriptors. It also needs a request to the Service Threat Monitoring Module for getting the service logs and a request to the Threat Repository Module for getting the countermeasures.	

4.1.9 Service Threat Monitoring Module

Description

The Service Threat Monitoring Module is invoked at deployment-time when a new service is deployed and registers to the monitoring service according the specifications reported in the contract.

The Service Threat Monitoring Module acts mostly as a daemon at run-time in charge of gathering monitoring controls for the deployed services, to store the gathered information in a log, and to send alerts to the notification module in case:

- The threat level has changed,
- The contract has been violated with respect to the implementation of the requested countermeasures.

Note that only the controls related to the threat countermeasures are monitored by the Service Threat Monitoring Module. The security properties and trustworthiness have independent monitoring modules able to send notifications in case of a change in the security properties resp. the trust level, and in case of a contract violation.

The Service Threat Monitoring Module interacts with the following components:

- The Model Transformation Module,
- The Marketplace, directly or through the Model Transformation Module,
- The Service Monitoring Module of the Environment, and any source of monitoring and events,
- The Notification Module

At deployment-time, the new composed service registers for monitoring. The Service Composition Framework sends a *registerForAlert(Registration)* to the Notification Module which sends back a *bindServiceThreat(Service, AniketosCompliantSpecification)* to the Model Transformation Module. The Model Transformation Module in turn extracts the monitoring controls from the *AniketosCompliantSpecification* and sends a *bindServiceToThreatMonitoring(Service, counterMID, counterMDesc)* to the Service Threat Monitoring Module. A symmetric *unBind* request shall be sent when the service is removed from the Aniketos Marketplace, or when it is modified.

At run-time, the Service Threat Monitoring Module receives events from the Service Monitoring Module of the Environment, based on the monitoring of a composing service. The Service Threat Monitoring Module then diagnoses for a change in the threat level or for a contract violation of a Service as registered in the Marketplace, and sends back an alert to the Notification module.

There are four main functionalities handled by the Service Threat Monitoring Module:

- Registering (bind and unbind) the list of the services to be monitored as well as the mapping of the monitoring control towards the composing services and the monitoring agents.
- Logging events and performing the diagnosis of a change of threat level or of a contract violation
- Sending the requested logs to the Service Threat Recommendation Module
- Sending alerts to the Notification Module

Limitations and Restrictions

The Service Threat Monitoring Module raises no limitations and restrictions.

Interfaces

The Service Threat Monitoring Module exposes the following two methods for the *IThreatMonitoring* interface:

Method Name	<i>bindServiceToThreatMonitoring</i>	
Method Definition/Description	This method registers to the monitoring module a Service with the associated monitoring controls for the service	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>counterMID</i>	String	The unique identifier of monitoring control to which the service subscribes.
<i>counterMDesc</i>	String	Description of the countermeasure, either in plain text or as a pattern.
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace

Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>eventType</i>	eventType	The event type
<i>momSource</i>	String	The monitoring source
<i>composingService</i>	List	The list of the composing services
Dependency with other components	This method serves the Model Transformation Module, i.e. receives the command to register a service for it to be monitored. The Model Transformation Module gets a <i>registerForAlert(Registration)</i> request from the Notification Module and transforms it into a <i>bindServiceToThreat(Service, Threat)</i> . It needs a call to the Marketplace or the Model Transformation Module in order to be able to decompose the request on the Service in elementary monitoring constraints on the <i>composingServices</i> . It also needs to register towards the monitoring sources of the events, i.e. the Service Monitoring Module of the Environment or any other Aniketos monitoring module.	

Method Name	<i>unBindServiceToThreatMonitoring</i>	
Method Definition/Description	This method removes from the monitoring module a Service with the associated monitoring controls for the service	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
eventType	eventType	The event type
momSource	String	The monitoring source
composingService	List	The list of the composing services
Dependency with other components	This method serves the Aniketos Marketplace or the Model Transformation Module, i.e. receives the command to remove the service from the list of services to be monitored.	

Method Name	<i>getServiceLog</i>	
Method Definition/Description	This method sends a selection of the event logs related to a given Service, optionally a given countermeasure and starting from a given past date.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>counterMID</i>	String	<i>Optional</i> The monitoring control ID of

		the event which should be sent as defined in the specifications and the threat repository, corresponding to the event.
<i>fromDate, fromTime</i>	String, String	The Date and time from which all event regarding the Service and possibly the countermeasure should be sent.
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
logfile	(composingService, monSource, counterMID, eventType, eventDesc, date, time)	A list of logs where composingService is the composingService counterMID the countermeasure, eventType the type of event, eventDesc the description of the event, date and time the date and time of the event concerned by the log.
Dependency with other components	This method serves the Service Threat Recommendation Module	

The Service Threat Monitoring Module exposes the following method for the *MonitoringServiceAccess* interface:

Method Name	<i>threatEvent</i>	
Method Definition/Description	This method receives an event from the Service Monitoring Module of the Environment and logs it.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>composingService</i>	String	The service on which the event has been detected.
<i>serviceProvider</i>	String	The service providers of the service on which the event has been detected.
<i>monSource</i>	String	Monitoring source element which sent the Event
<i>counterMID</i>	String	<i>Optional</i> The monitoring control ID as defined in the specifications and the threat repository, corresponding to the event.
<i>eventType</i>	String	An ID code reported by the monitoring source
<i>eventDesc</i>	String	More contextual information reported by the source (for example, ContextChange, ServiceChange)

<i>date</i>	String	Date of the event
<i>time</i>	String	Time of the event
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method serves the Service monitoring module of the Environment, i.e. receives the event and treats it. All the events received are logged and the output of the method might trigger an alert.	

The Service Threat Monitoring Module exposes the following method for the *IAAlert* interface:

Method Name	<i>alert</i>	
Method Definition/Description	This method sends an alert to the Notification module after diagnosis of a change or threat.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>composingService</i>	String	The service on which the event has been detected.
<i>serviceProvider</i>	String	The service providers of the service on which the event has been detected.
<i>monSource</i>	String	Monitoring source element which sent the Event
<i>counterMID</i>	String	<i>Optional</i> The monitoring control ID as defined in the specifications and the threat repository, corresponding to the event.
<i>eventType</i>	String	An ID code reported by the monitoring source
<i>eventDesc</i>	String	More contextual information reported by the source
<i>date</i>	String	Date of the event
<i>time</i>	String	Time of the event
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>service</i>	Service	the service on which the event is related as referenced in the MarketPlace
<i>counterMID</i>	String	<i>Optional</i> The monitoring control ID as defined in the specifications and the threat repository, corresponding to the event.
<i>alertType</i>	String	Any type of alert to be sent to

		the Notification module ¹ : <ul style="list-style-type: none"> • ServiceChange • ContextChange • ThreatLevelChange • ContractViolation
<i>alertvalue</i>	String	Value of monitored parameter
<i>alertDesc</i>	String	Description of the Alert
Dependency with other components	This method serves the Service monitoring module of the Environment, i.e. receives the event and treats it.	

4.1.10 Notification module

Description

As indicated in D1.2, 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. 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.

The Notification Module interacts with the following components:

- The Service Monitoring Module of the Environment, the Service Threat Monitoring Module, the Security Policy Monitoring Module, the Trustworthiness Component, and any source of monitoring and events,
- The CMM,
- The Model Transformation Module,
- Any component or service which has registered for alerts.

At run-time, each time a monitoring module from the Aniketos environment (i.e. the Service Threat Monitoring Module, the Security Policy Monitoring Module or the Trustworthiness Component) diagnoses a change or a contract violation, it sends an alert to the Notification Module. The Notification Module sends then:

- Either the alert in a publish/subscribe mode to the components or services which have subscribe to it according a given threshold. This goes along with a registration interface to the publish/subscribe mechanism. This is the *IAlert interface*.
- Or an *analyseSecureComposition(SecurityVerificationRequest)* to the CMM. This is the *ISecurityVerification interface*.

After a service is re-adapted by means of the Model Transformation Module, which generates a *generateSecurityDescriptor(AniketosCompliantSpecification): SecurityDescriptor*, the Notification Module will notify the runtime environment about changes related to trust and security properties as well as threat situation. This is the *INotification interface*.

There are three main functionalities handled by the Notification Module:

- Receive alerts, send alerts in a publish/subscribe mode, and register/unregister the services or module to that service,
- Notify the Service run-time environment for a reconfiguration or a re-composition

¹ Note that the Service Threat Monitoring module will only send these types of alerts. The other types are taken in charge by other Aniketos platform components

- Send verification requests to the CMM and/or SPDM when alerts are received.

Limitations and Restrictions

The Notification Module raises no limitations and restrictions.

Interfaces

The Notification Module exposes the two interfaces.

For the **IAAlert** interface the following four methods are exposed:

Method Name	<i>receivedAlert</i>	
Method Definition/Description	This method receives an alert from the Service Threat Monitoring Module (and possibly the Trustworthiness Module and the Security Policy Monitoring Module) in a Request/Reply mode. As a response it publishes an alert in a Publish/Subscribe mode	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>counterMID</i>	String	The monitoring control ID as defined in the specifications and the threat repository, corresponding to the event.
<i>alertType</i>	String	Any type of alert to be sent to the Notification module: <ul style="list-style-type: none"> • ServiceChange • ContextChange • ThreatLevelChange • SecurityPropertyChange • TrustLevelChange • ContractChange • ContractViolation
<i>AlertValue</i>	String	A value of monitored parameter
<i>AlertDesc</i>	String	A description of the Alert
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	Receives alerts from Service threat monitoring module (and possibly the Trustworthiness Component and the Security policy monitoring module)	

Method Name	<i>publishedAlert</i>	
Method Definition/Description	This method publishes an alert in a Publish/Subscribe mode	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Method output attributes		

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>alertChannelID</i>	String	Instead of sending the list of the subscribers in the head of the message, the publisher and the subscriber will share a channel ID for the type of alert (alertChannelID) of the same type of priorities. If there are two alertThresholds for the same alertType, two alertChannelIDs are needed.
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>counterMID</i>	String	The monitoring control ID as defined in the specifications and the threat repository, corresponding to the event.
<i>alertType</i>	String	Any type of alert to be sent to the Notification module: <ul style="list-style-type: none"> • ServiceChange • ContextChange • ThreatLevelChange • SecurityPropertyChange • TrustLevelChange • ContractChange • ContractViolation
<i>AlertValue</i>	String	A value of monitored parameter
<i>AlertDesc</i>	String	A description of the Alert
Dependency with other components	Publishes alerts to any module or service which has subscribed to the alert with a given threshold	

Method Name	<i>registerForAlert</i>	
Method Definition/Description	This method registers a service to receive alerts in a publish/subscribe mode	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>alertType</i>	ServiceChange ContextChange TrustLevelChange ThreatLevelChange SecurityPropertyChange ContractChange ContractViolation	Any type of alert to be sent to the Notification module
<i>alertThreshold</i>	String	The threshold above which the alert should be sent.
Method output attributes		

<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>alertChannelID</i>	alertChannelID	Instead of sending the list of the subscribers in the head of the message, the publisher and the subscriber will share a channel ID for the type of alert (alertChannelID) of the same type of priorities. If there are two alertThresholds for the same alertType, two alertChannelIDs are needed.
Dependency with other components	When the Service Composition Framework or the Model Transformation Module sends a <i>registerForAlert</i> request, the Notification Module should also send a <i>bindServiceToThreat</i> to the Service Threat Monitoring Module.	

Method Name	<i>unRegisterForAlert</i>	
Method Definition/Description	This method unregisters a service from the publish/subscribe service for the alerts	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>alertType</i>	ServiceChange ContextChange TrustLevelChange ThreatLevelChange SecurityPropertyChange ContractChange ContractViolation	Any type of alert to be sent to the Notification module
<i>alertThreshold</i>	String	The threshold above which the alert should be sent.
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>alertChannelID</i>	alertChannelID	Instead of sending the list of the subscribers in the head of the message, the publisher and the subscriber will share a channel ID for the type of alert (alertChannelID) of the same type of priorities. If there are two alertThresholds for the same alertType, two alertChannelIDs are needed.
Dependency with other components	This method depends on the Aniketos Marketplace directly or through the Model Transformation Module for it to warn for Service unsubscription.	

For the **INotification** interface the following method is exposed:

Method Name	<i>notify</i>	
Method Definition/Description	When a new composition or configuration has been prepared by the Model transformation module, the notification module passes the information to the Run-time Environment	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>SecurityDescriptor</i>	String	The security descriptor of the service as computed by the Model transformation module.
Dependency with other components	This method sends a request to the Service Runtime Environment	

For the **ISecurityVerification** interface the following method is exposed:

Method Name	<i>analyseSecureComposition</i>	
Method Definition/Description	This method sends a request to the Verification Module for it to verify the compliance of the Service at run-time	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>SecurityVerificationRequest</i>	String	Includes the reference to the service. Triggers a verification by the Security verification module
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method enables interaction with the CMM	

4.1.11 Community Support Module

Description

The Community Support Module is a content repository giving support to all Aniketos stakeholders, such as service developers, service composers and end users, with material, including patterns and guidelines for developing trust and security properties in composite service engineering and establishing trust among end users, as well as demonstration material to enable them realising the use of the Aniketos platform into their service engineering practices.

Limitations and Restrictions

The Community Support Module raises the following limitations and restrictions:

- H/W limitations and requirements: None
- S/W limitations and requirements: PHP, MySQL, Drupal
- Operating System: any operating system capable of running PHP, MySQL and Drupal.

Interfaces

The Community Support Module exposes the following three methods for the *CommunitySupportService* interface:

Method Name	<i>integrateTool</i>	
Method Definition/Description	This method integrates a software tool into to Community Support Module. Tools will be used by developers.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>tool</i>	Tool	Software and reference material for a tool to be used by developers
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method enables the Aniketos stakeholders to interact with the Community Support Module through the Marketplace	

Method Name	<i>maintainSoftwareAndService</i>	
Method Definition/Description	This method updates the software or reference information for a tool in the Community Support Module.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>tool</i>	Tool	Software and reference material for a tool to be used by developers
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method enables the Aniketos stakeholders to interact with the Community Support Module through the Marketplace	

Method Name	<i>provideTrustworthiness</i>
--------------------	-------------------------------

Method Definition/Description	This method provides trustworthiness attributes to a service.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
<i>trustworthiness</i>	Trustworthiness	Trustworthiness attributes
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method enables the Aniketos stakeholders to interact with the Community Support Module through the Marketplace	

4.1.12 Threat Repository Module

Description

The Threat Repository Module is a database containing a list of threats and their associated countermeasures. According to the precisions on countermeasures proposed above, the countermeasures stored in the Threat Repository can be either security policies, or security patterns or monitoring controls.

The Threat Repository Module interacts with three other components:

- The Community Support Module
- The Socio-technical Security Modelling Tool
- The Threat Response Recommendation Module

The Threat Repository Module is used by the community to dynamically update the repository of threats and countermeasures.

It is involved in the specification of a composed service, where the Socio-technical Security Modelling Tool sends a *getCountermeasures(ServiceSpecification, null, null): (ThreatIDs, counterMID, counterMType, counterMDesc)* request for a given service specification. The Threat Repository Module gets the request and sends back adapted answers to the Socio-technical Security Modelling Tool for it to build an Aniketos compliant specification.

The Threat Repository Module is also called at run-time by the Threat Response Recommendation Module in order to provide the responses to the threats, that is to say to find the countermeasures to the threats.

There are two main functionalities handled by the Threat Repository Module:

- Set the information stored in the repository. These functionalities enable the users of the community to add, update, or delete the threats and associated countermeasures in the repository.
- Get the information stored in the repository. These functionalities enable the users of the community at design-time and the Threat Response Recommendation Module at run-time to get the threats and associated countermeasures in the repository.

Limitations and Restrictions

The Threat Repository Module raises no limitations and restrictions, but reserves dependencies to the external tool used to provide to envisaged functionalities.

Interfaces

The Threat Repository Module exposes the *ThreatRepositoryService* interface through the following seven methods:

Method Name	<i>addThreat</i>	
Method Definition/Description	This method is used to add the threats in the Threat Repository	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceSpecifications</i>	ServiceSpecification	Optional: ServiceSpecifications to which the threat applies
<i>SecuritySpecifications</i>	SecuritySpecification	Optional: SecuritySpecifications to which the threat applies
<i>threatDesc</i>	String	Description of the threat in plain text
<i>counterMIDs</i>	counterMID	Optional: Countermeasures that can apply to the threat
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>threatID</i>	String	A unique identifier for the threat
Dependency with other components	It serves the Community Support Module to enable Aniketos stakeholders to add threats to the Repository	

Method Name	<i>updateThreat</i>	
Method Definition/Description	This method is used to update the threats in the Threat Repository	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>threatID</i>	String	A unique identifier for the threat
<i>ServiceSpecifications</i>	ServiceSpecification	Optional: ServiceSpecifications to which the threat applies
<i>SecuritySpecifications</i>	SecuritySpecification	Optional: SecuritySpecifications to which the threat applies
<i>threatDesc</i>	String	Description of the threat in plain text
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	It serves the Community Support Module to enable Aniketos stakeholders to update threats on the Repository	

Method Name	<i>deleteThreat</i>
--------------------	---------------------

Method Definition/Description	This method is used to delete the threats in the repository	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
ThreatID	String	A unique identifier for the threat
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	It serves the Community Support Module to enable Aniketos stakeholders to delete threats from the Repository	

Method Name	<i>getThreats</i>	
Method Definition/Description	This method gets the threats in the repository	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceSpecifications</i>	ServiceSpecification	Optional: ServiceSpecifications to which the threat applies
<i>SecuritySpecifications</i>	SecuritySpecification	Optional: SecuritySpecifications to which the threat applies
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>threatID</i>	threatID	The unique identifier for the threat
<i>threatDesc</i>	String	A description of the threat in plain text
<i>counterMIDs</i>	String	Optionally the countermeasures that can apply to the threat
Dependency with other components	It serves the Community Support Module to enable Aniketos stakeholders to get threats to the Repository. It also enables the Socio-technical security modelling tool to look-up threats on the Repository	

Method Name	<i>addCountermeasure</i>	
Method Definition/Description	This method is used to add a countermeasure in the Threat Repository	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>relatedThreats</i>	threatID	Threats to which the countermeasure applies
<i>counterMType</i>	Policy, pattern, monitoring control	The type of the countermeasure
<i>counterMDesc</i>	string	Description of the countermeasure, either in plain

		text or as a pattern
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
counterMID	String	A unique identifier for the countermeasure
Dependency with other components	It serves the Community Support Module	

Method Name	<i>updateCountermeasure</i>	
Method Definition/Description	This method is used to update a countermeasure in the Threat Repository	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>counterMID</i>	String	A unique identifier for the countermeasure
<i>relatedThreats</i>	ThreatID	Threats to which the countermeasure applies
<i>counterMType</i>	Policy, pattern, monitoring control	The type of the countermeasure according to the proposed classification in section 5.2.
<i>counterMDesc</i>	String	Description of the countermeasure, either in plain text or as a pattern
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	It serves the Community Support Module	

Method Name	<i>deleteCountermeasure</i>	
Method Definition/Description	This method is used to delete a countermeasure from the Threat Repository	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
counterMID	String	A unique identifier for the countermeasure
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method serves the interaction with the Community Support Module	

Method Name	<i>getCountermeasures</i>
--------------------	---------------------------

Method Definition/Description	This method is used to get from the Threat Repository the list of the countermeasures and their attributes which harden a given <i>ServiceSpecification</i> , a given <i>SecuritySpecification</i> or a given Threat	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceSpecification</i>	ServiceSpecification	Optional: a ServiceSpecification to be hardened
<i>SecuritySpecification</i>	SecuritySpecification	Optional: a SecuritySpecification to be hardened
<i>ThreatID</i>	String	Optional: a Threat to be hardened
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>countermeasures</i>	(ThreatIDs, counterMID, counterMType, counterMDesc)	The countermeasures which correspond to a ServiceSpecification, or a SecuritySpecification, or a ThreatID, as well as the full attributes of the countermeasures and the ThreatIDs they counter.
Dependency with other components	This method serves the interaction with the: <ul style="list-style-type: none"> • Community Support Module • Threat Response Recommendation Module • Socio-technical Security Modelling Tool • Service Threat Monitoring Module 	

4.1.13 Marketplace

Description

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.

The Marketplace includes the logical Environment component (Service Registry) and mechanism (Service Discovery Mechanism) that have been specified in D1.2.

The Marketplace will consist of two sub-modules:

- A web service that will allow remote clients to register and discover services
- A web platform, based on Drupal CMS, which will present a friendly User Interface for accessing Marketplace services. This web platform will also be the container of other Aniketos components, such as Community Support Module and Training Material Module

Limitations and Restrictions

The Web Service sub-module of the Marketplace raises the following limitations and restrictions:

- H/W limitations and requirements: None
- S/W limitations and requirements: OSGi bundles component
- Operating System: any operating system capable of running JVM and an OSGi container

The Web Platform sub-module of the Marketplace raises the following limitations and restrictions:

- H/W limitations and requirements: None
- S/W limitations and requirements: PHP, MySQL, Drupal
- Operating System: any operating system capable of running PHP, MySQL and Drupal

Interfaces

The Marketplace exposes the following three methods for the *IMarketplace* interface:

Method Name	<i>announceService</i>	
Method Definition/Description	This method registers a service in the Marketplace	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>marketPlaceAnnouncement</i>	MarketPlaceAnnouncement	
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method enables the service providers to publish their services into the Marketplace	

Method Name	<i>discoverService</i>	
Method Definition/Description	This method discovers services that match specific criteria	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>discoveryRequest</i>	DiscoveryRequest	A class containing the request for service discovery
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>Service</i>	List	List of services that match the criteria
Dependency with other components	This method enables the Marketplace to offer service discovery functionality to all the other Aniketos Platform and Environment components	

Method Name	<i>getSecurityDescriptor</i>
Method	This method gets the security descriptor for a specific service

Definition/Description		
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	String	The identification number of a specific service in the Markeyplace
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>SecurityDescriptor</i>	SecurityDescriptor	The service security descriptors
Dependency with other components	This method enables the Marketplace to provide the security descriptors of a specific service to all the other Aniketos Platform components	

4.1.14 Training Material Module

Description

The Training Material Module contains training and individual learning materials that enable the uptake of Aniketos practices and results and the development and delivery of secure and trustworthy services.

Limitations and Restrictions

The Training Material Module raises the following limitations and restrictions:

- H/W limitations and requirements: None
- S/W limitations and requirements: PHP, MySQL, Drupal
- Operating System: any operating system capable of running PHP, MySQL and Drupal

Interfaces

The Training Material Module exposes the following three methods for the *TrainingMaterialService* interface:

Method Name	<i>addReferenceMaterial</i>	
Method Definition/Description	This method is used to add a reference material item to the Training Material Module	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>referenceMaterial</i>	ReferenceMaterial	It contains the reference material to be added to the Training Material Module
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method enables the Aniketos stakeholders to interact with the Training Material Module	

Method Name	<i>getReferenceMaterial</i>	
Method Definition/Description	This method is used to get a reference material item from the Training Material Module	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>name</i>	String	Name of reference material item
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>referenceMaterial</i>	ReferenceMaterial	It contains the reference material to be added to the Training Material Module
Dependency with other components	This method enables the Aniketos stakeholders to interact with the Training Material Module	

Method Name	<i>removeReferenceMaterial</i>	
Method Definition/Description	This method is used to remove reference material item from the Training Material Module	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>name</i>	String	Name of reference material item
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	This method enables the Aniketos stakeholders to interact with the Training Material Module	

4.2 Environment components

This section describes the Aniketos Environment components, which, although they do not constitute part of the Aniketos research work, they are necessary to realise the Aniketos platform components' functionalities.

4.2.1 Service Composition Framework

Description

The Service Composition Framework enables service composition, which entails functionalities like service specification, service validation, service discovery, assembly and deployment of services. In order to offer these functionalities, this component mainly uses the interfaces provided by three logical components belonging to the Environment as depicted in D1.2: Service specification/planning mechanism, Service validation mechanism and Service discovery mechanism (in cooperation with the Marketplace).

Limitations and Restrictions

The Service Composition Framework raises limitations and restrictions, which are implied by the external technologies adopted to provide the envisaged functionalities.

Interfaces

The Service Composition Framework exposes the following four methods for the *ServiceCompositionFrameworkInterface* interface:

Method Name	<i>specifyService</i>	
Method Definition/Description	This method creates the service specification of a composite service	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceSpecification</i>	ServiceSpecification	Specification of a composed/component service, or the planning or template creation for service composition.
Dependency with other components	In order to obtain the <i>ServiceSpecification</i> , the Service Composition Framework invokes Service specification/planning mechanism.	

Method Name	<i>discoverAndSelectService</i>	
Method Definition/Description	This method is used to discover services based on the trustworthiness and security properties and select the set of service components that makes up the composite service. This method returns a composition plan.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>serviceQuery</i>	ServiceQuery	A query specifying trustworthiness and security properties to make the selection of service components
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>compositionPlan</i>	Composition Plan	
Dependency with other components	This method enables the Aniketos developers to interact with the Service Composition Framework	

Method Name	<i>validateService</i>
--------------------	------------------------

Method Definition/Description	The method is used to validate the functional properties of the composed service in terms of what it is supposed to do	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>populatedCompositionPlan</i>	PopulatedCompositionPlan	It is a composition plan
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>SecurityVerificationResult</i>	String	
Dependency with other components	This method enables the Aniketos developers to interact with the Service Composition Framework	

Method Name	<i>AssembleAndDeployService</i>	
Method Definition/Description	This method creates a runnable composite service instance based on the service specification, and deploys the service to make it available	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceSpecification</i>	ServiceSpecification	Specification of a composed/component service, or the planning or template creation for service composition.
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
Dependency with other components	This method enables the Aniketos developers to interact with the Service Composition Framework	

4.2.2 Service Runtime Environment

This component is a runtime environment for the execution of services. It will be composed by the following mechanisms:

- The Service Discovery Mechanism, which is used for service discovery, both for service components with and without Aniketos support
- The Service Execution Mechanism, which is a mechanism for loading and executing a service (part of the runtime environment)
- The Service Recomposition Mechanism, which is a mechanism to recompose services at runtime (part of the runtime environment)
- The Service Adaptation Mechanism, which is a mechanism to adapt a service at runtime (part of the runtime environment), such as changing security properties like the length of an encryption key

- The Service Monitoring Module: 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)
- The Context sensor: functionality or service that at runtime can be used to detect context changes that can trigger Aniketos activities (e.g. change of user location)

This component is not the main part of the Aniketos development, but it is necessary to demonstrate how existing techniques and tools in those areas, such as Music (provided by SINTEF) and PRRS (provided by ATOS), can be integrated with the Aniketos platform with respect to the following aspects:

- Security and trust in recomposition/adaptation of composite services.
- Monitoring and evaluation of trustworthiness and security violations of service contracts, also considering contextual information such as change in operation conditions and users' behaviour.
- Runtime validation of secure service behaviour.

The Service Runtime Environment (SRE) is involved in:

- The Aniketos runtime service provision process: a Service Provider wants to deploy a service with a specific service specification
- The Aniketos runtime service validation process: a Service Provider or a recomposition agent will use this component to validate a service. The SRE will send an *analyseSecureComposition* request to the Aniketos Verification Module
- The Aniketos runtime recomposition process: a recomposition agent will use this component to discover and select a Service as well as negotiate a contract and assemble a new compose service. The SRE will communicate with the Marketplace for the discovery mechanism. It will also send a *suggestSecureComposition* request to the Secure Composition Planner Module and a request for contract negotiation to the Verification Module and specifically the Contract Manager Module.
- The Aniketos runtime reconfiguration process: in this case, a recomposition agent will send a request to adapt a service and the SRE will perform the adaptation and send an *analyseSecureComposition* request to the Verification Module. Optionally, the SRE will send a contract negotiation or update request to the Verification Module and specifically the Contract Manager Module.
- The Aniketos runtime monitoring process: the SRE will invoke the Service Monitoring Module to include a service in the monitoring process. The Service Monitoring Module will send notification of threat events to the Service Threat Monitoring Module and will send alerts of context changes to the Notification Module.

As said, two existing solutions can contribute to the above mentioned functionalities of SRE, which are presented in the following sections.

4.2.2.1 PRRS

Description

The platform for runtime re-configurability of security (PRRS) provides run-time management of Security and Dependability (S&D) solutions and monitoring of the system context. The objective of the Aniketos Marketplace will be to provide service consumers with S&DServices (Security and Dependability Services), i.e. services that assure a specific requirements on trust and security properties. Consequently, Service Providers should include in the marketplace only S&DServices with their offered specifications as service descriptors.

The security and dependability in services is what can be assured through the PRRS or SRF (*Serenity Runtime Framework*)². With that goal, service developers will have to deploy their services as *SRF-aware applications*. In other words, they will design services whose security requirements will be implemented (in a context-aware way) thanks to the S&DServices that PRRS can provide during runtime. It is up to the SRF-aware service to initiate a request to the PRRS to obtain a S&DService to satisfy its own security requirements. Once the S&DService has been deployed and is in use by the SRF-aware service, the S&DService can be monitored at runtime by the PRRS platform.

At design-time:

- When a developer wants to create an S&DService, it will have to identify the security requirements that this S&DService will satisfy. This step will be performed with the help of the Socio-technical security modeling tool from the security needs.
- The PRRS stores S&DServices in a database called S&DLibrary. In essence, it is a repository with a collection of S&DServices and S&DSolutions that satisfy different possible security requirements. Each solution will have associated³
 - The security properties that it provides (e.g. “userAuthentication”)
 - The interfaces provided with that solution (e.g. an interface with a simple way to authenticate users through a function called “authentication” that results in a boolean value indicating if it was successfully or not).
 - The event capturer and event format that the S&DService needs to generate events in order that security requirement can be monitored.
 - The monitoring rules for that type of events.
 - The context conditions (field “preconditions”) under which a S&DService or S&DSolution is applicable or not.

It should be noted that MUSIC also takes into account the changes in properties and context at runtime to select a service or another and the way of interaction is under study in Aniketos to improve the reconfiguration mechanism.

The Service Composition Framework and/or Service Runtime Environment could provide an interface to developers to show the different security requirements or properties, which S&DSolution can be provided by the PRRS. This will be evaluated in a later phase.

It should also be noted that the PRRS provides a support library API and documentation needed to help developers to build SRF-aware applications to deal with all the SRF and Executable Component communication issues. It only needs to be imported by the client application and used as needed.

- Once a new S&DService is ready, a S&D Service Provider will be able to include it in the Aniketos Marketplace indicating the Aniketos security attributes that this S&DService can offer.

² We are going to use both terms indistinctly. SRF (Serenity Runtime Framework) is the name of the PRRS (Platform for Run-time re-configurability of security) from the latest version.

³ These S&DSolutions have to be implemented by S&D experts and stored in the S&DLibrary before being accessible to developers

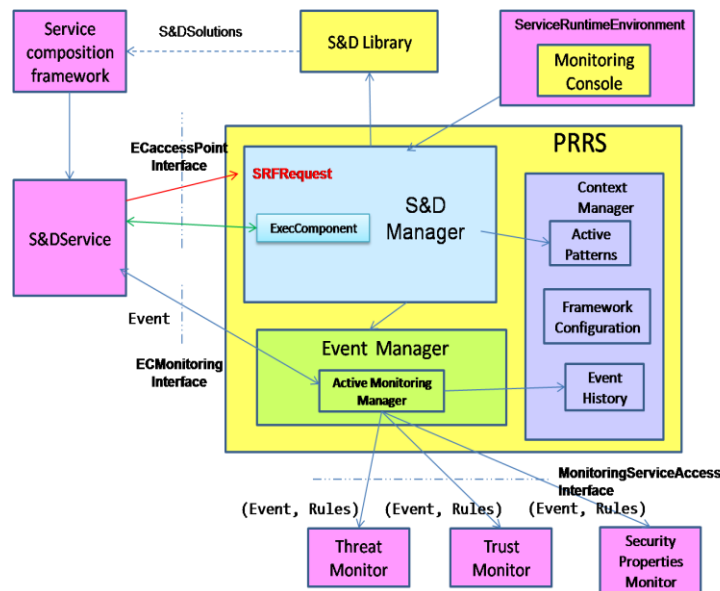


Figure 20: The logical architecture of PRRS

At run-time:

The PRRS is implemented as a service running in a device, on top of the operating system, and listening to requests from SRF-aware applications.

- When a S&DService, that has been selected through the discovery mechanism provided by the Service runtime environment, is activated (e.g. the service is instantiated and starts working), it will create a connection with the SRF by means of the SRF_AP_AccessPoint class provided in the support API and through the interface ECaccessPoint will send requests to the PRRS in order to fulfill their security requirements.

NOTE: According to figure 45 in D1.2, a method serviceMonitor() would be required and used to register from the Service Runtime Environment new composed services in the Monitoring Service. By the moment PRRS only register services (SRF-aware applications) at runtime when they request a pattern. A possible improvement for Aniketos could be to add an interface to register S&DServices in the Service Monitoring.

- These requests are processed by the SRF Manager that will check the S&DLibrary and ContextManager databases to automatically select the S&DService or S&DSolution that better complies with the security properties requested taking into account the current context. The SRF answers with an instance of the Executable Component handler (EChandler) associated to the selected solution and ready to be used by the application.

This EChandler (it may be a web-service reference to the S&DService or it can be an implemented JAR file⁴) will be the responsible:

- To provide the methods or operations that are called by the SRF-aware application (the interfaces known at design-time and included in the application)
- To provide an event capturer for the SRF-aware application that will be used for the generation of events to be sent to the PRRS to ensure the security property.

⁴ The implementation of these S&DSolutions should be provided by Security Experts with their tests performed and certifications to assure they provide Security and Dependability. This has been done prior to storing those S&DServices and S&DSolutions in the S&DLibrary of the PRRS.

- To provide a reaction listener method in order the SRF-aware application can receive notifications from the PRRS (e.g. in case of a violation in the security property)
- Once the ECHandler is created the process of the Service Monitoring provided by the PRRS will start:
 - The executable component will be registered in the ContextManager with the active pattern (security solutions selected by the PRRS). This register could be used in Aniketos Notification module or Threat repository module for the register of active services with their associated security solutions (countermeasures) but this is still under evaluation.
 - The executable component will generate/collect events in the Event Capturer and will send them to the PRRS to be monitored. PRRS offers the possibility of receiving events by socket or through a web service. These Event Capturers could be any S&DService (SRF-aware application) running on any external component (e.g. any component from Aniketos architecture) which triggers an Event to the interface provided by the PRRS.
 - The PRRS will send the events received and the monitoring rule associated (defined in the S&DSolution) to the monitoring systems that will check it as also defined in the S&DSolution. All the Aniketos platform components dealing with threat monitoring, trust monitoring and security properties monitoring need to have access to the *MonitoringServiceAccess* interface of PRRS in this context.
 - The PRRS will check periodically the different monitoring services in order to detect rule violations by the events. In that case, the event will be stored in the ContextManager database and a notification will be sent to the S&DService that was registered and is the source of that event. So, that application will be able to react and e.g. to invoke again the security solution because other implementation could be more suitable. A possible improvement for Aniketos would be to send this notification from PRRS to the Notification module or even include in PRRS a mechanism to register components that have to be notified. This will be evaluated in a later phase.
- The PRRS provides a Console of monitoring for the Service runtime environment. This console will show:
 - The monitoring services available
 - The patterns that are active
 - The events that have violated some rule

It is a matter of future discussions on whether this Console should be integrated in the Service Runtime Environment with the GUI provided by MUSIC.

Limitations/Requirements:

The PRRS raises the following limitations and restrictions:

- H/W limitations and requirements: None
- S/W limitations and requirements for the Service Monitoring Module:
 - Developed in Java, making use of client-server architecture
 - MySQL
- Operating System for the Service Monitoring Module: Windows

Interfaces

The PRRS for the Service Monitoring Module implements the following interfaces

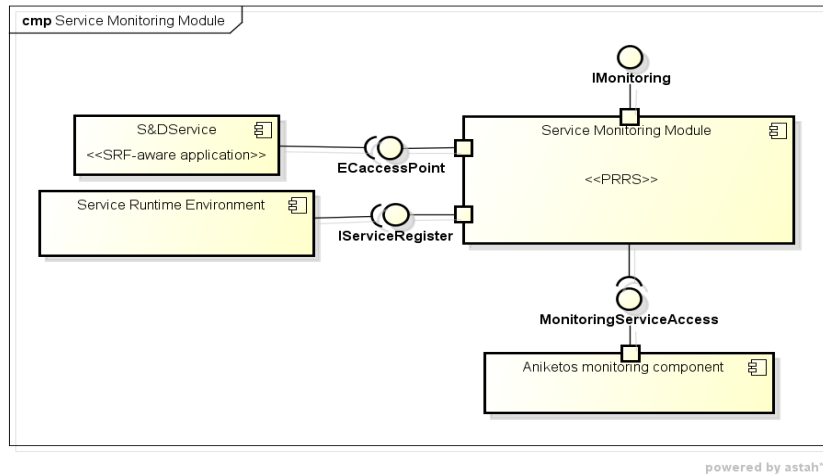


Figure 21: The interfaces of the Service Monitoring Module

For the **IMonitoring** interface, the following method is exposed:

Method Name	<i>sendEvent</i>	
Method Definition/Description	This method is used by applications to send events to be monitored	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>Event</i>	String	String form of the XML representation of an Event
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	S&DServices and external components that trigger events to be monitored.	

For the **IServiceRegister** interface, the following method is exposed:

Method Name	<i>serviceMonitor</i>	
Method Definition/Description	This method is used by the Service Runtime Environment to register new Services in the monitoring service NOTE: Individual services developed like SRF-aware applications are automatically registered in the PRRS monitoring service when they request some S&Dsolution at runtime	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>ServiceID</i>	ServiceID	The identification of service specification in the Marketplace
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other	-	

components	
-------------------	--

For the **ECAccessPoint**⁵ interface, the following three methods are exposed:

Method Name	<i>requestSolution</i>	
Method Definition/Description	This method is used by applications to request an S&Dsolution to the PRRS. An executable component handler will be returned	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>solution</i>	String	String with a pattern to search for the solution
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>EC handler</i>		
Dependency with other components	S&DServices	

Method Name	<i>callOperation</i>	
Method Definition/Description	This method is used to call an operation provided by an executable component. It returns a String of bytes	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>operation</i>	String	Operation provided in the S&Dsolution that is invoked.
<i>arguments</i>	Map	List of arguments for the operation
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
	Bytes[]	
Dependency with other components	S&DServices	

Method Name	<i>callOperationObjec</i>	
Method Definition/Description	This method is used to call an operation provided by an executable component. It returns a serializable object	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>operation</i>	String	Operation provided in the

⁵ To be used by SRF-aware applications by means of the SRF_AP_AccessPoint and SerenityExecutableComponent_AP classes provided by the PRRS API.

		S&Dsolution that is invoked.
<i>arguments</i>	Map	List of arguments for the operation
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
	Object	
Dependency with other components	S&DServices	

For the **MonitoringServiceAccess** interface, the following four methods are exposed:

Method Name	<i>sendMonitoringRules</i>	
Method Definition/Description	This method is used to send monitoring rules and events from the PRRS to the monitoring system that will check the rule	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>monitorAddress</i>	String	Address of the monitoring system
<i>userName</i>	String	Credentials
<i>password</i>	String	Credentials
<i>ruleInput</i>	String	Monitoring rule to be sent
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>resultMonitoring</i>	Boolean	The result of sending the monitoring rules
Dependency with other components	-	

Method Name	<i>sendEventToMonitor</i>	
Method Definition/Description	This method is used to send event to be monitored from the PRRS to the monitoring system that will check the rule.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>monitorAddress</i>	String	Address of the monitoring system
<i>userName</i>	String	Credentials
<i>password</i>	String	Credentials
<i>eventInput</i>	String	Event to be sent
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>resultMonitoring</i>	Boolean	The result of sending the event

		for monitoring
Dependency with other components	-	

Method Name	<i>checkMonitoringRule</i>	
Method Definition/Description	This method is used to check if a rule has been violated in the monitoring system. In that case a message is returned to the PRRS with the information related.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>monitorAddress</i>	String	Address of the monitoring system
<i>userName</i>	String	Credentials
<i>password</i>	String	Credentials
<i>ruleInput</i>	String	Rule to be checked
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>resultCheck</i>	String	The result after checking the rule
Dependency with other components	-	

Method Name	<i>unsubscribeMonitoringRules</i>	
Method Definition/Description	This method is used to unsubscribe a monitoring rule in a monitoring system	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>endPointUrl</i>	String	End point with the monitoring system
<i>userName</i>	String	Credentials
<i>password</i>	String	Credentials
<i>input</i>	String	Rule to be unsubscribed
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>resultUnsubscribe</i>	Boolean	The result from unsubscribing from monitoring rules
Dependency with other components	-	

4.2.2.2 MUSIC

Description

MUSIC framework can play two roles within Aniketos:

- It can be primarily used as an example of the service runtime environment, with functionalities like service re-composition mechanism, service execution mechanism, service adaptation mechanism and context sensors.
- It can also be used at design time as service composition framework and service planning mechanism.

Limitations/Requirements:

MUSIC raises the following limitations and restrictions:

- H/W limitations and requirements: None
- S/W limitations and requirements: Java 1.4 or Java ME CDC, OSGi
- Operating System: MUSIC can run over Windows, Linux, Windows Mobile and Android devices

Interfaces

MUSIC framework consists of the MUSIC studio for developing applications and the MUSIC middleware for running applications. It is possible for the MUSIC studio to interact with Aniketos at design time by looking up the threat response recommendations and incorporating them into the MUSIC UML-based models. However, this is not the focus of the current stage and will not be considered at this stage.

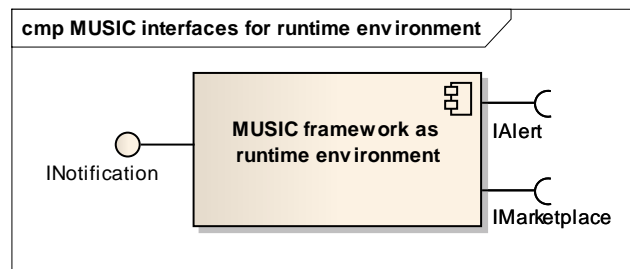


Figure 22: The runtime interfaces of MUSIC towards Aniketos

Focusing on the use of MUSIC middleware as the runtime environment for Aniketos the interfaces projected in Figure 22 can be considered. Thus, MUSIC uses the following interfaces provided by Aniketos platform:

- **IAlert:** this interface is used to register a service with the Notification Module when a service is deployed. The method used is *registerForAlert(Registration):void*.
- **IMarketplace:** this interface is used to publish a service in Aniketos Marketplace (method: *announceService(MarketplaceAnnouncement):void*) and to discover a service from the Marketplace (method: *discoverService(DiscoveryRequest):DiscoveryResponse*).

MUSIC offers the following interfaces:

- **INotification:** this interface is used to receive Aniketos notification at runtime. The method is *publishedAlert(): alertChannelID, Service, counterMID, alertType, alertValue, alertDesc*.

For the **INotification** interface, the following method is exposed:

Method Name	<i>alert</i>
Method Definition/Description	MUSIC runtime receives alert about changes of trust level, threat level and security properties of the registered services

Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>alert</i>	Alert	Contains information about the alert <i>type</i> (valid types: Trust level change, Threat level change and Security property change), <i>notifier</i> and <i>receiver</i> of the message as well as the <i>description</i> .
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
-		
Dependency with other components	Basically, it is the Aniketos Notification module that invokes this method to pass notifications to MUSIC middleware	

4.2.3 Identity Management Service

The Identity Management Service is a specific mechanism that is exploited to allocate trust to identifiable entities. It runs in the environment, but accessed by the Aniketos platform. The implementation of Identity Management (IdM) will be based either on internal method(s) or by exploitation of external Identity Provider (IdP) services.

In order to better focus on this service and the relevant mechanisms that are necessary for identity management, one should consider the structure of the target solution. The Aniketos platform is composed by three main support blocks: design-time, run-time and community. Design-time support is locally installed in Integrated Development Environments (IDE) of Service Developers. Run-time support is installed locally in the runtime environment of Service Providers. Community support is the unique centralized platform entity which will be used by all the actors involved, including Service Developers, Service Providers and End Users.

Limitations/Requirements:

The IdM raises limitations and restrictions, which are implied by the external components adopted to support the envisaged functionalities.

Interfaces

Through the *IdM* interface, the following methods are exposed:

Method Name	<i>verifyEntity</i>	
Method Definition/Description	This method is used to identify entities in Aniketos. Here entities are considered the service providers and consumers and the platform users. This method returns the result of the identification process.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>entityID</i>	EntityID	The identifier of the entity to be authenticated
<i>verificationToken</i>	VerificationToken	The proof of the identity claimed

Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>identityVerificationResult</i>	IdentityVerificationResult	The result of the identification process
Dependency with other components	This method makes it possible for the Aniketos platform components to interact with the Aniketos environment components in a trusted manner. In particular, interactions have been envisaged with the Contract Manager Module, the Marketplace and the Community Support Module.	

Method Name	<i>registerEntity</i>	
Method Definition/Description	This method is used to register entities in Aniketos. The method returns the result of the registration process.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>entityID</i>	EntityID	The identifier of the entity to be authenticated
<i>verificationToken</i>	VerificationToken	The proof of the identity claimed
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>identityRegistrationResult</i>	IdentityRegistrationResult	The result of the registration process.
Dependency with other components	This method makes it possible to register an entity as a trusted entity into the system.	

Method Name	<i>unregisterEntity</i>	
Method Definition/Description	This method is used to unregister entities in Aniketos. The method returns the result of the registration process.	
Method input attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>entityID</i>	EntityID	The identifier of the entity to be authenticated
<i>verificationToken</i>	VerificationToken	The proof of the identity claimed
Method output attributes		
<i>Name</i>	<i>Type</i>	<i>Description</i>
<i>identityUnRegistrationResult</i>	IdentityUnRegistrationResult	The result of the unregistration process.
Dependency with other components	This method makes it possible to unregister an entity previously registered into the system	

4.3 Summary of interfaces

This section makes a summary of the presented interface specifications of Sections 4.1 and 4.2. The scope is to provide the readers with a coherent and comprehensive roadmap to the dependencies between the components, while it serves as the connection point with the Aniketos Deliverable D1.2 [1] and the sequence (collaboration model view) diagrams (Figures 23-46) presented there.

Table 6 presents the relation of the Aniketos baseline implementation, performed in the context of D5.1, with respect to the design and the specification document D1.2. As it can be seen, D5.1 provides a superset of the defined methods and interfaces in D1.2, which shows the progress being performed with respect to the design phase. For the sake of the implementation, it is obvious that the specifications are subject to changes, which are planned for the forthcoming deliverable D1.5, which is due M33.

Table 6: Relation of D5.1 implementations to D1.2 specifications

Component	Interface	Method	Consumed by	Relation to D1.2 ⁶
Socio-technical Security Modelling Tool	<i>IModelling</i>	<i>modelTrustAndSecurityRequirements</i>	Service Composition Framework	<ul style="list-style-type: none"> It corresponds to Figure 23 for the interface definition Figure 38 shows the interaction of the Socio-technical modelling tool with the Service Composition Framework at the design time service specification process
		<i>exportSecuritySpecificationModel</i>	Model Transformation Module	
Model Transformation Module	<i>ModelTransformationModuleService</i>	<i>transformModel</i>	<ul style="list-style-type: none"> Contract Manager module Service Composition Framework 	<ul style="list-style-type: none"> It corresponds to Figure 23 for the interface definition Figure 38 shows the interaction of the Model Transformation Module with the Service Composition Framework at the design time service specification process Figure 41 shows the interaction of the Model Transformation Module with the Service Composition Framework and the Notification Module at the design time service assembly and deployment process Figure 42 shows the interaction of the Model Transformation Module with the Marketplace at the runtime service provision process
		<i>generateSecurityDescriptor</i>	<ul style="list-style-type: none"> Service Composition Framework Marketplace 	
		<i>generateContractTemplate</i>	Contract Manager Module	
		<i>bindServiceThreat</i>	Notification Module	

⁶ Contains references to the figures of D1.2

Component	Interface	Method	Consumed by	Relation to D1.2 ⁶
Trustworthiness Component	<i>TrustworthinessService</i>	<i>getTrustworthiness</i>	<ul style="list-style-type: none"> ▪ Marketplace ▪ Contract Manager Module (CMM) 	<ul style="list-style-type: none"> ▪ It corresponds to Figure 25 (<i>ITrustworthiness Prediction</i> interface) for the interface definition ▪ Figure 39 shows the interaction of the Trustworthiness Component (Trustworthiness Prediction Module) with the Marketplace at the design time service discovery and selection process ▪ Figure 43 shows the interaction of the Trustworthiness Component (Trustworthiness Prediction Module) with the CMM (Security Verification Module) at the runtime service validation process ▪ Figure 44 shows the interaction of the Trustworthiness Component (Trustworthiness Prediction Module) with the Marketplace at the runtime re-composition ▪ Figure 45 shows the interaction of the Trustworthiness Component (Trustworthiness Prediction Module) with the CMM (Security Verification Module) at the runtime reconfiguration ▪ Figure 46 shows the interaction of the Trustworthiness Component (Trustworthiness Prediction Module) with the Notification Module (through the <i>alert</i> method) at the runtime monitoring
		<i>receiveAlert</i>	Notification Module	
		<i>receiveQoSupdate</i>	-	
	<i>MonitoringServiceAccess</i>	<i>sendEventToMonitor</i>	Service Monitoring Component	
		<i>deriveMonitoringRules</i>	Service Monitoring Component	

Component	Interface	Method	Consumed by	Relation to D1.2 ⁶
Contract Manager Module	<i>ContractManagerService</i>	<i>AnalyseSecureComposition</i>	<ul style="list-style-type: none"> ▪ Secure Composition Planner Module ▪ Service Runtime Environment ▪ Service Composition Framework ▪ Notification Module 	<ul style="list-style-type: none"> ▪ It corresponds to Figure 27 (<i>ISecurityVerification</i> interface) for the interface definition ▪ Figure 40 shows the interaction of the Contract Manager Module with the Service Composition Framework at the design time service validation and contract establishment ▪ Figure 43 shows the interaction of the Contract Manager Module with the Service Runtime Environment at the runtime service validation process ▪ Figure 45 shows the interaction of the Contract Manager Module with the Service Runtime Environment at the runtime reconfiguration ▪ Figure 46 shows the interaction of the Contract Manager Module with the Notification Module at the runtime monitoring
		<i>checkCountermeasuresImplementation</i>	Service Runtime Environment	
		<i>getTrustworthiness</i>	Trustworthiness Component	
		<i>determineSecurityProperties</i>	Service Runtime Environment	
Property Verification Module	<i>PropertyVerificationService</i>	<i>verifyTechnicalTrustProperties</i>	Contract Manager Module	No association
Composition Security Validation Module	<i>CompositionSecurityValidationService</i>	<i>VerifyCompositionCompliance</i>	Contract Manager Module	No association
Security Property Determination Module	<i>IService</i>	<i>getService</i>	<ul style="list-style-type: none"> ▪ Marketplace ▪ Contract Manager Module (CMM) 	<ul style="list-style-type: none"> ▪ It corresponds to Figure 26 (<i>ISecurityProperty</i> interface) for the interface definition
		<i>addService</i>		
		<i>addProperty</i>		
		<i>getValue</i>		

Component	Interface	Method	Consumed by	Relation to D1.2 ⁶
		<i>setValue</i>		
	<i>IProperty</i>	<i>getProperty</i>		
Secure Composition Planner Module	<i>CompositionPlannerInterface</i>	<i>selectSecureCompositions</i>	<ul style="list-style-type: none"> ▪ Service Composition Framework ▪ Security Property Determination Module ▪ Trustworthiness Component ▪ Service Runtime Environment 	<ul style="list-style-type: none"> ▪ It corresponds to Figure 35 (<i>ICompositionPlanner</i> interface) for the interface definition ▪ Figure 39 shows the interaction of the Secure Composition Planner Module with the Service Composition Framework at the design time service discovery and selection process ▪ Figure 44 shows the interaction of the Secure Composition Planner Module with the Service Runtime Environment at the runtime re-composition
		<i>orderSecureCompositions</i>		
		<i>suggestSecureComposition</i>		
Security Policy Monitoring Module	<i>ISecurityPolicyMonitoring</i>	<i>getContract</i>	<ul style="list-style-type: none"> ▪ Aniketos Environment components ▪ Service Runtime Environment 	No association
		<i>getRealData</i>		
Threat Response Recommendation Module	<i>IThreatResponseRecommendation</i>	<i>checkCountermeasuresImplementationTwSpec</i>	Contract Manager Module	<ul style="list-style-type: none"> ▪ It corresponds to Figure 34 for the interface definition
		<i>checkCountermeasuresImplementationTwContract</i>		
Service Threat Monitoring Module	<i>IThreatMonitoring</i>	<i>bindServiceToThreatMonitoring</i>	<ul style="list-style-type: none"> ▪ Model Transformation Module ▪ Aniketos Marketplace ▪ Service Threat Recommendation Module 	<ul style="list-style-type: none"> ▪ It corresponds to Figure 29 for the interface definition (the <i>MonitoringServiceAccess</i> interface corresponds to the <i>ITheatEvent</i>) ▪ Figure 41 shows the interaction of the Service Threat Monitoring Module with the Model Transformation Module at the design time
		<i>unBindServiceToThreatMonitoring</i>		
		<i>getServiceLog</i>		

Component	Interface	Method	Consumed by	Relation to D1.2 ⁶
	<i>MonitoringServiceAccess</i>	<i>threatEvent</i>	Service Monitoring Component	<ul style="list-style-type: none"> Figure 46 shows the interaction of the Service Threat Monitoring Module with the Service Monitoring Component and the Notification Module at the runtime monitoring
	<i>IAlert</i>	<i>alert</i>	Notification Module	
Notification module	<i>IAlert</i>	<i>receivedAlert</i>	<ul style="list-style-type: none"> Service Threat Monitoring Module Service Monitoring Module Security Policy Monitoring Module Trustworthiness Component 	<ul style="list-style-type: none"> It corresponds to Figure 31 for the interface definition Figure 41 shows the interaction of the Notification module with the Service Composition Framework at the design time service assembly and deployment process Figure 46 shows the interaction of the Notification module with the Service Monitoring Module, the Service Threat Monitoring Module and the Security Policy Monitoring Module at the runtime monitoring (<i>alert</i> method corresponds to the <i>receivedAlert</i> here) Figure 47 shows the interaction of the Notification module with the Trustworthiness Component (Monitor Trustworthiness Module) and the Security Policy Monitoring Module at the end user processes
		<i>publishedAlert</i>	-	
		<i>registerForAlert</i>	<ul style="list-style-type: none"> Model Transformation Module Service Composition Framework 	
	<i>unRegisterForAlert</i>	<ul style="list-style-type: none"> Model Transformation Module Marketplace 		
	<i>INotification</i>	<i>notify</i>	Service Runtime Environment	

Component	Interface	Method	Consumed by	Relation to D1.2 ⁶
	<i>ISecurityVerification</i>	<i>analyseSecureComposition</i>	<ul style="list-style-type: none"> ▪ Contract Manager Module (CMM) ▪ Property Verification Module ▪ Composition Security Validation Module 	
Community Support Module	<i>CommunitySupportService</i>	<i>integrateTool</i>	Marketplace	<ul style="list-style-type: none"> ▪ It corresponds to Figure 37 (<i>ICommunitySupport</i> interface) for the interface definition ▪ Figure 48 shows the interaction of the Community Support Module with the Aniketos stakeholders at the processes for enriching the Aniketos platform at runtime
		<i>maintainSoftwareAndService</i>	Marketplace	
		<i>provideTrustworthiness</i>	Marketplace	
Threat Repository Module	<i>ThreatRepositoryService</i>	<i>addThreat</i>	Community Support Module	<ul style="list-style-type: none"> ▪ It corresponds to Figure 32 (<i>IThreatRepository</i> interface) for the interface definition ▪ Figure 38 shows the interaction of the Threat Repository Module with the Socio-technical Security Modelling Tool (the <i>lookupThreat</i> method corresponds to the <i>getThreats</i> method here) at the design time service specification process ▪ Figure 48 shows the interaction of the Threat Repository Module with the Aniketos contributors at the processes for enriching the Aniketos platform at runtime
		<i>updateThreat</i>	Community Support Module	
		<i>deleteThreat</i>	Community Support Module	
		<i>getThreats</i>	<ul style="list-style-type: none"> ▪ Community Support Module ▪ Socio-technical Security Modelling Tool 	
		<i>addCountermeasure</i>	Community Support Module	
		<i>updateCountermeasure</i>	Community Support Module	
		<i>deleteCountermeasure</i>	Community Support Module	

Component	Interface	Method	Consumed by	Relation to D1.2 ⁶
		<i>getCountermeasures</i>	<ul style="list-style-type: none"> ▪ Community Support Module ▪ Threat Response Recommendation Module ▪ Socio-technical Security Modelling Tool ▪ Service Threat Monitoring Module 	
Marketplace	<i>IMarketplace</i>	<i>announceService</i>	All components	<ul style="list-style-type: none"> ▪ It corresponds to Figure 33 for the interface definition ▪ Figure 39 shows the interaction of the Marketplace with the Service Providers and the Service Composition Framework at the design time service discovery and selection process ▪ Figure 41 shows the interaction of the Marketplace with the Service Providers at the design time service assembly and deployment process ▪ Figure 42 shows the interaction of the Marketplace with the Service Providers at the runtime service provision process ▪ Figure 44 shows the interaction of the Marketplace with the Service Runtime Environment at the runtime re-composition ▪ Figure 46 shows the interaction of the Marketplace with the Security Policy
		<i>discoverService</i>	All components	

Component	Interface	Method	Consumed by	Relation to D1.2 ⁶
		<i>getSecurityDescriptor</i>	All components	<ul style="list-style-type: none"> Monitoring Module at the runtime monitoring Figure 48 shows the interaction of the Marketplace with the Security Providers at the processes for enriching the Aniketos platform at runtime
Training Material Module	<i>TrainingMaterialService</i>	<i>addReferenceMaterial</i>	Aniketos stakeholders	<ul style="list-style-type: none"> It corresponds to Figure 36 (<i>ITraining</i> interface) for the interface definition Figure 48 shows the interaction of the Training Material Module with the Aniketos contributors at the processes for enriching the Aniketos platform at runtime
		<i>getReferenceMaterial</i>	Aniketos stakeholders	
		<i>removeReferenceMaterial</i>	Aniketos stakeholders	
Service Composition Framework	<i>ServiceCompositionFrameworkInterface</i>	<i>specifyService</i>	-	<ul style="list-style-type: none"> Figure 39 shows the interaction of the Service Composition Framework with the Service Developers at the design time service discovery and selection process Figure 40 shows the interaction of the Service Composition Framework with the Service Developers at the design time service validation and contract establishment Figure 41 shows the interaction of the Service Composition Framework with the Service Developers at the design time service assembly and deployment process
		<i>discoverAndSelectService</i>	-	
		<i>validateService</i>	-	
		<i>AssembleAndDeployService</i>	-	
Service Monitoring Component (PRRS)	<i>IMonitoring</i>	<i>sendEvent</i>	-	No association
	<i>IServiceRegister</i>	<i>serviceMonitor</i>	-	
	<i>ECAccessPoint</i>	<i>requestSolution</i>	-	
		<i>callOperation</i>	-	
		<i>callOperationObjec</i>	-	
<i>MonitoringServiceAcc</i>	<i>sendMonitoringRules</i>	-		

Component	Interface	Method	Consumed by	Relation to D1.2 ⁶
	<i>ess</i>	<i>sendEventToMonitor</i>	-	
		<i>checkMonitoringRule</i>	-	
		<i>unsubscribeMonitoringRules</i>	-	
Service Runtime Environment (MUSIC)	<i>INotification</i>	<i>alert</i>	Notification module	<ul style="list-style-type: none"> Figure 42 shows the interaction of the Service Runtime Environment with the Notification Module at the runtime service provision process
Identity Management Service	<i>IdM</i>	<i>verifyEntity</i>	-	No association
		<i>registerEntity</i>	-	
		<i>unregisterEntity</i>	-	

5 Design of the Aniketos Marketplace

The Aniketos 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. Through the Aniketos Marketplace, service providers will be able to upload their offered specifications as service descriptors, so that their services are made available for discovery.

The architecture of the Aniketos Marketplace, including the appropriate Service Registry is depicted in Figure 23.

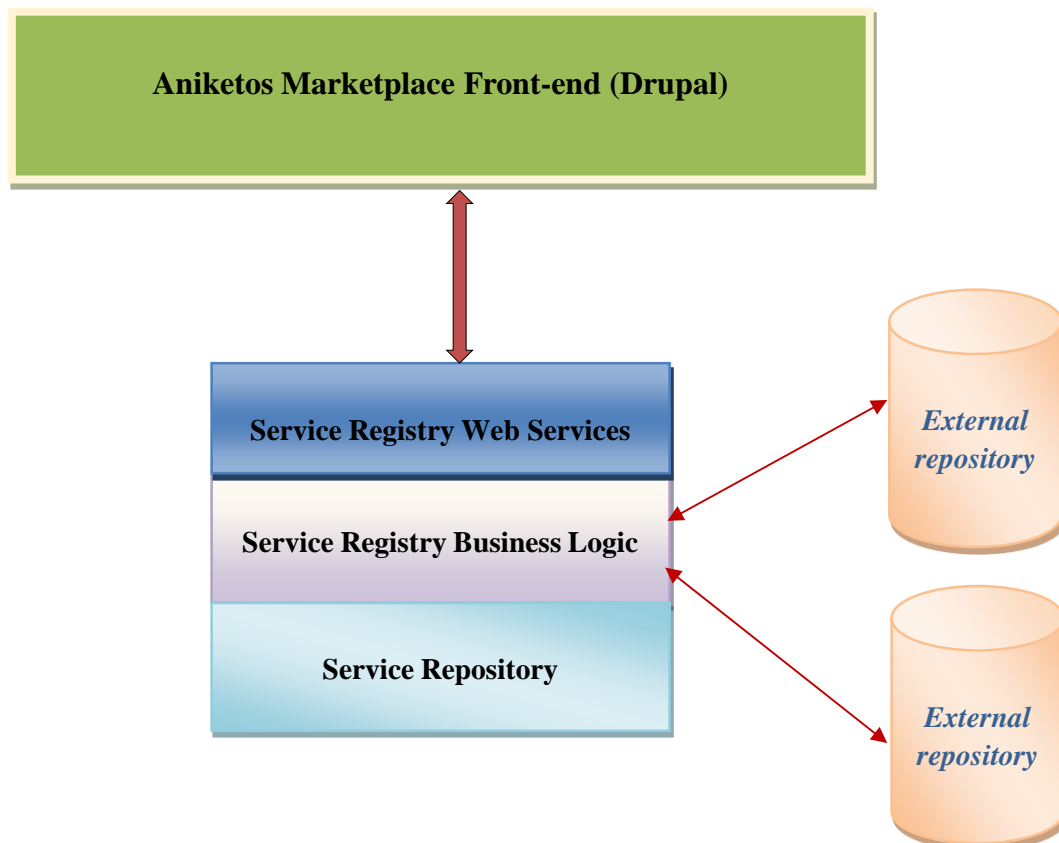


Figure 23: The Architecture of the Aniketos Marketplace

The Service Registry is designed on a three-layered architecture approach. The first layer (*Service Repository*) comprises the persistence layer, which maintains information about all deployed services. The intermediate layer is a thin business logic layer (*Service Registry Business Logic*) that enforces all the rules required by the service registry. The component can interface with the external world through a third layer (*Service Registry Web Services*).

The capability of the Marketplace to interact with external service registries will make the integration of existing systems into Aniketos Platform easier. In legacy systems, it may be difficult or impossible to move services from one repository to another. This won't be a problem for Aniketos adoption as the current repository could still be used and the registration to Aniketos Marketplace will store the security attributes needed by Aniketos Platform. Also the benefits of Aniketos Platform could be exploited, without affecting the current operation of a SOA system.

All services will be registered in the Aniketos internal service repository. This repository maintains information regarding the security attributes that are needed for a service to be part of the Aniketos execution system.

The internal repository, in its final form, will be a full implementation of a service repository system, providing every piece of information needed for publishing and discovering a service. However, it will

be also able to communicate with external service repositories to retrieve this information. Therefore for an existing service, a provider may choose to keep the repository that is already in use and publish in Aniketos repository only the Aniketos specific information.

A major client of the Service Registry Component is the Aniketos Marketplace front-end. This front-end offers the following functionalities:

- Users and roles management in a centralized way
- Graphical implementation of the Aniketos marketplace interface (*announceService*, *discoverService*, *getSecurityDescriptor*)
- Tight interaction with the Community Support Module

The Aniketos Marketplace front-end is being developed on top of the Drupal CMS system [49].

The Service Registry Web Services will be also consumed by many runtime and design components, as their service discovery interface is a main feature of the Aniketos Platform.

The Aniketos Marketplace offers interfaces for registering and discovering services. The following Figure 24 depicts an initial sketch of the service registration user interface.

Figure 24: Register Service User Interface

Upon service registration, the users of the Marketplace can search for the specific service and any other service, which has been registered as an Aniketos compliant service in the Marketplace. Figure 25 visualises how the user can provide search criteria and view the results in a tabular form. It should be clarified that this is the graphical interface for accessing the contents of the Marketplace, but the respective software interface for accessing the Marketplace has also been designed and is available in stub mode, enabling other Aniketos platform and environment components to interact with the Marketplace services.

Service Id:
Service Name:
Keywords:
Tags:
Provider:
Security Attributes:

- Security Attribute 1
- Security Attribute 2
- Security Attribute 3
- Security Attribute 3
- Security Attribute 3

Name	Url	Provider	Tags	Attributes
Service 1	http://www.aniketos.eu/marketplace/service1	Provider 1	test dummy	Security Attribute 1 Security Attribute 3
Service 2	http://www.aniketos.eu/marketplace/service2	Provider 1	test dummy	Security Attribute 1 Security Attribute 3
Service 3	http://www.aniketos.eu/marketplace/service3	Provider 2	test dummy	Security Attribute 1 Security Attribute 3
Service 4	http://www.aniketos.eu/marketplace/service4	Provider 2	test dummy	Security Attribute 1 Security Attribute 3
Service 5	http://www.aniketos.eu/marketplace/service5	Provider 2	test dummy	Security Attribute 1 Security Attribute 3
Service 6	http://www.aniketos.eu/marketplace/service6	Provider 3	test dummy	Security Attribute 1 Security Attribute 3
Service 7	http://www.aniketos.eu/marketplace/service7	Provider 4	test dummy	Security Attribute 1 Security Attribute 3

Figure 25: Searching for Services

6 Aniketos Baseline Implementation

6.1 Aniketos Platform at design-time

As per Section 2, at design-time, the Aniketos Platform is used from the service developer in the service creation and specification phase. The Platform facilitates the required functionalities for discovering existing services, selecting service components, validating services, establishing contracts, assembling service compositions and deploying services (please refer to Figure 26).

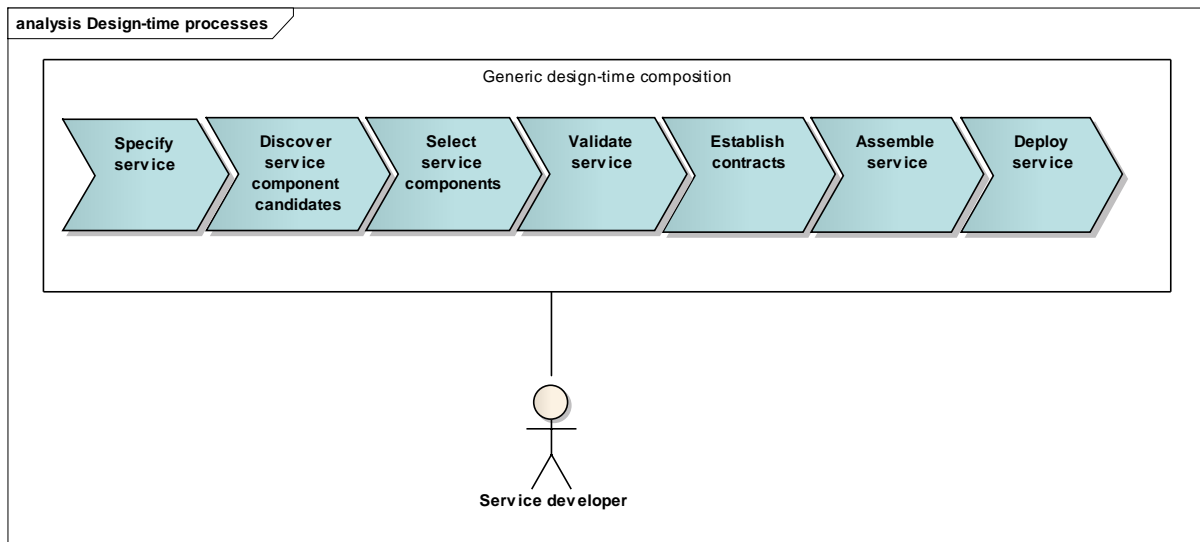


Figure 26: Typical processes related to design-time service composition (please refer to D1.2 [1])

As mentioned in Section 3.2, the Aniketos design-time Platform is based on the Eclipse IDE. Eclipse will integrate all existing and new components that are part of the Aniketos design-time Platform. The service developer will have access to all Aniketos design functionalities through a single IDE environment.

6.1.1 Design-time related Aniketos Components

Figure 27 depicts the design-time related components and their interactions with the runtime system.

Inside the “Eclipse IDE” box, the components (as per Figure 5) that will be fully or partially implemented as Eclipse plug-ins are presented. The blue components offer rich design capabilities, while the orange ones only act as mediators to their corresponding runtime components. The green components are part of the runtime system and provide services to the design time components.

Each design time component is implemented as one or more Eclipse plug-ins. For existing Eclipse plug-ins, a surrounding wrapper plug-in may need to be implemented, so that the original plug-in can be integrated into the Aniketos platform.

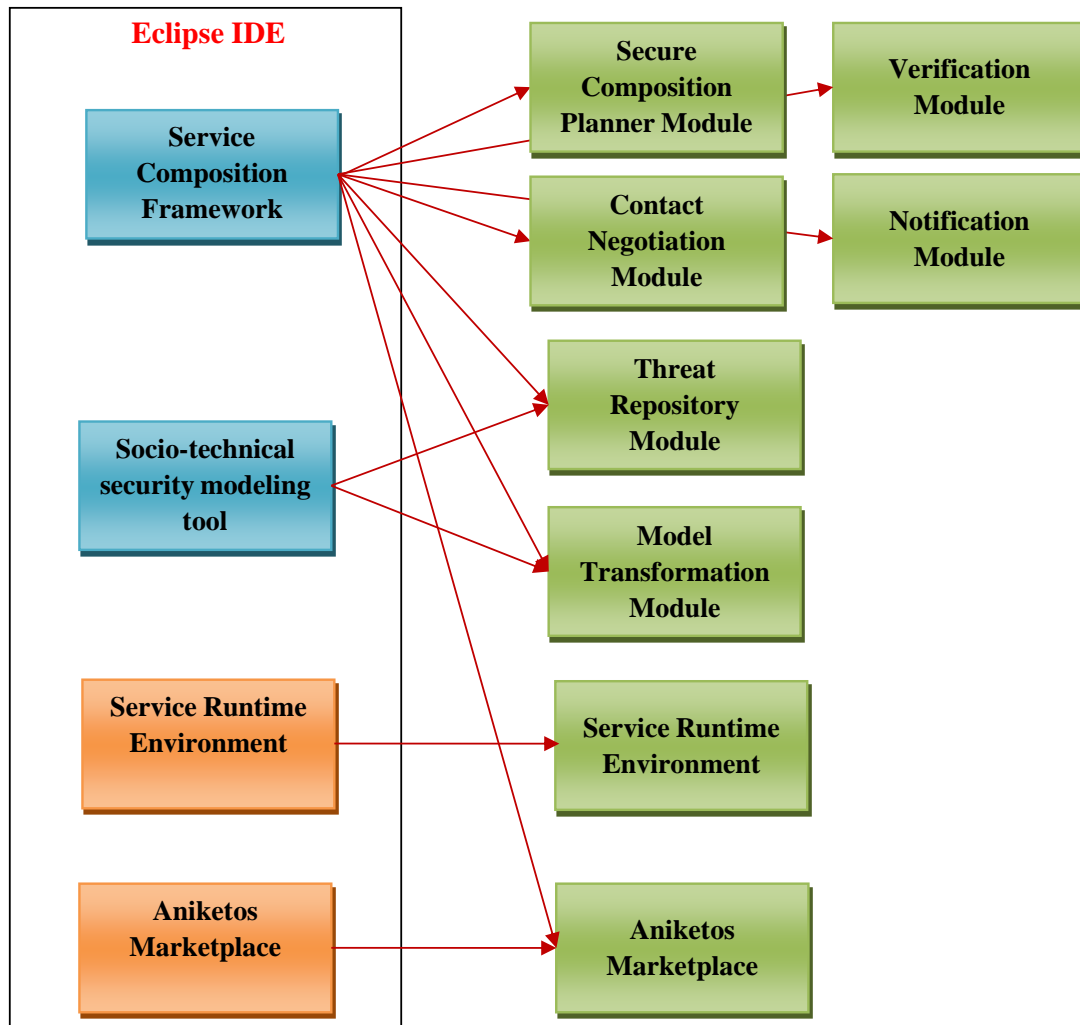


Figure 27: The design-time related Components

6.1.2 Template Eclipse Plug-in

For the platform basis implementation a template Eclipse plug-in has been developed. This template plug-in demonstrates how the features that are expected to be common in all design time components should be implemented. More specifically, it provides guidelines on the:

- Integration within Eclipse's graphical user interface
- Exchange of data between two Eclipse plug-ins
- Communication with runtime components through SOAP and REST web services

The Aniketos Eclipse plugin uses Eclipse plugin development environment (PDE) and OSGI technology. It is distributed as a set of Eclipse plugin and OSGI bundles. This dummy plugin demonstrates:

- The basics for creating an Eclipse plugin that contributes to the UI
- Packaging a plugin for installation within the Eclipse workbench
- Creating and registering an Eclipse View object
- Creating an OSGI-compatible Eclipse plugin that can locate and communicate with other OSGI bundles running in the same Equinox instance, comprising:
 - Using the Bundle's Activator to initialise OSGI services being made available to others
 - Using the Bundle's context to get a reference to OSGI services to be used
- Starting an Eclipse runtime environment that includes these plugins and bundles

The client plugin uses OSGI methods to locate and invoke a method on a separate bundle. Currently, it uses a default context (Equinox), and relies on the platform being correctly configured in order to locate, install, and start the relevant bundles. Full configuration details for a suitable Eclipse platform are included.

The Template Eclipse Plug-in is described in detail in Annex 8.2.

6.2 Runtime Platform

As per Section 2, the Aniketos Platform at runtime is mainly used from the service providers to manage and monitor security and trustworthiness in the execution of composite services. The Runtime Platform facilitates the required functionalities for monitoring service execution, validating services, reacting to changes in service provisioning, services re-composition and contracts' reconfiguration (please refer to Figure 28).

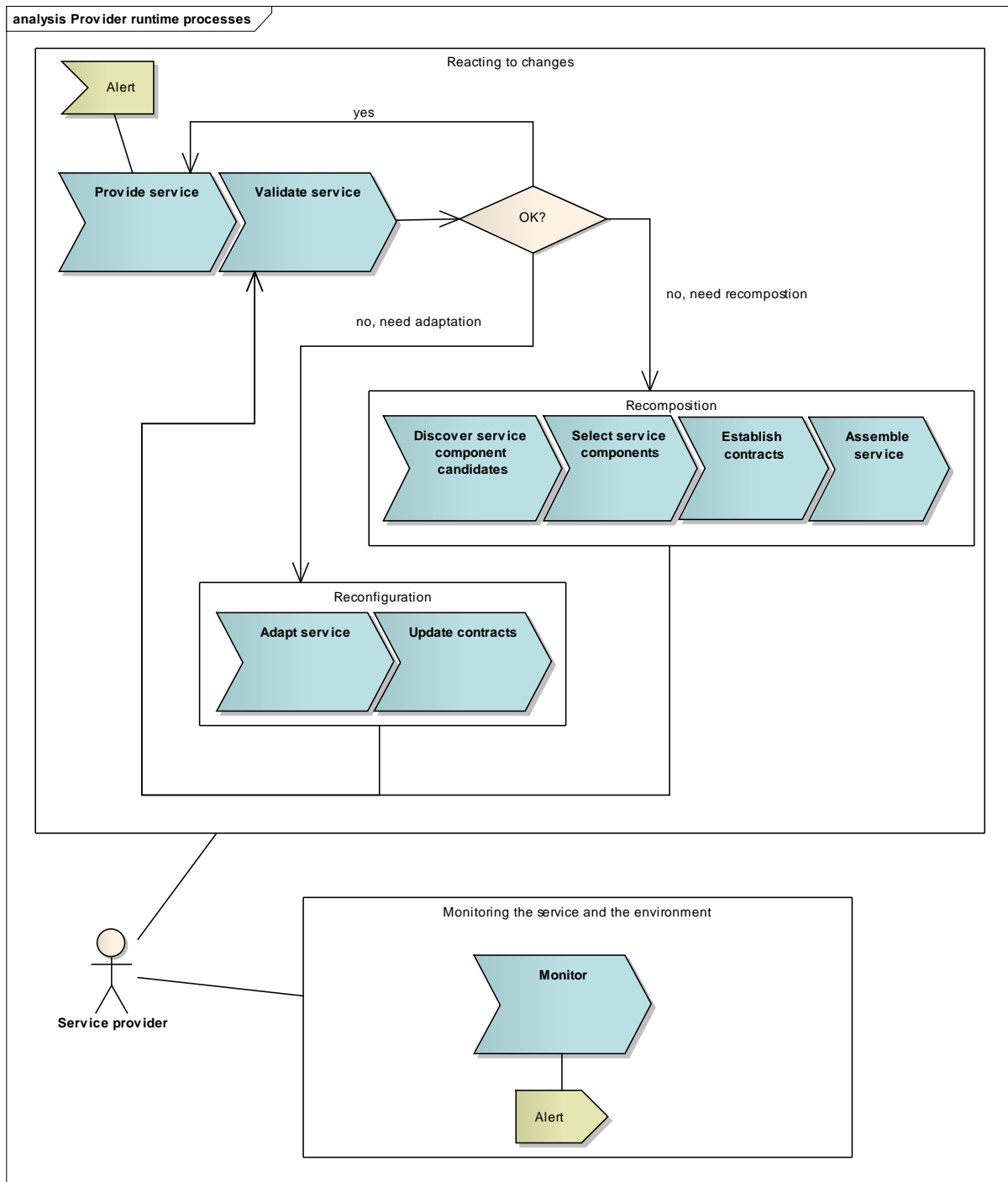


Figure 28: General processes related to runtime reaction to changes and monitoring (please refer to D1.2 [1])

The Runtime Aniketos Platform is based on a simplified architecture, which is depicted in Figure 29

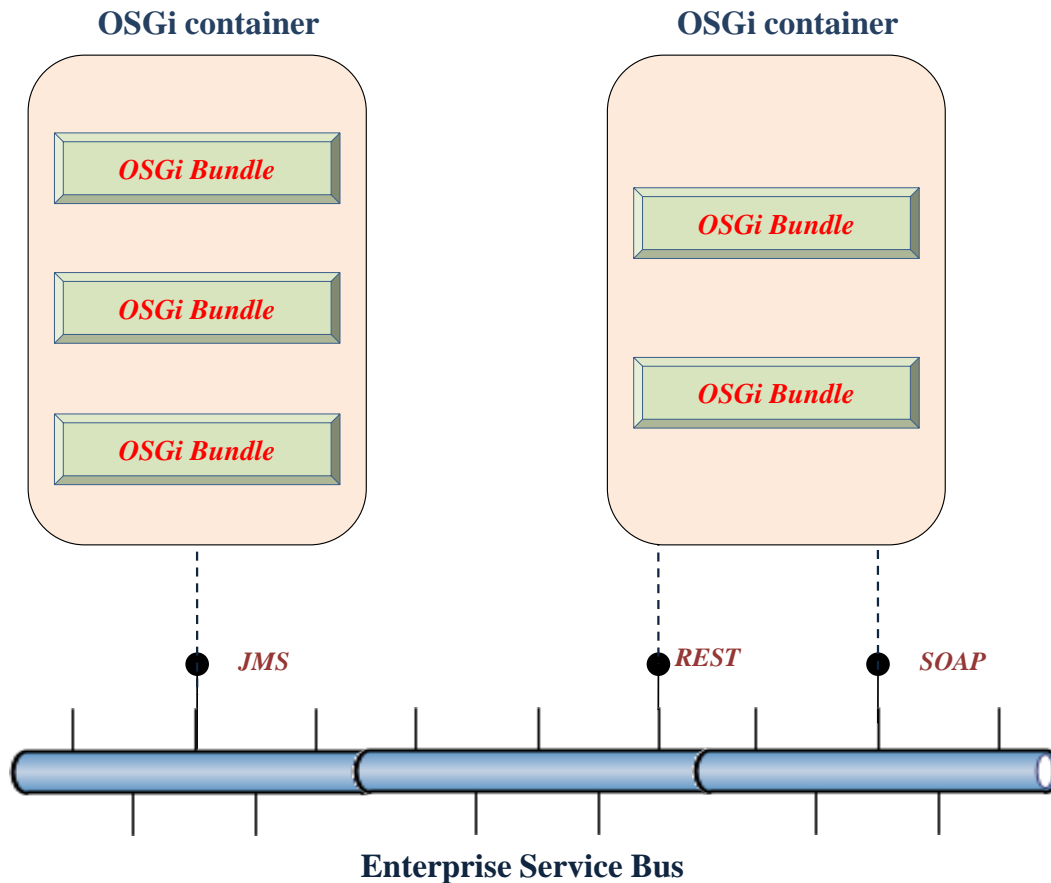


Figure 29: The Runtime Platform

One or more OSGi containers for hosting a number of OSGi bundles can be used. Within the same container the bundles can interact with each other through the services API defined in the services layer of the OSGi specification. Bundles in different containers will interact with each other through the ESB.

6.2.1 Template OSGi Bundle

For the platform basis implementation a template OSGi bundle has been developed. This bundle features all the functionalities required by the Aniketos platform components. More specifically, it supports:

- Interaction with other bundles in the same container
- REST and SOAP interface points
- Lifecycle management

The Aniketos Runtime Template Component is based on OSGi technology and is distributed as a set of OSGi bundles. It uses the Apache CXF implementation of Distributed OSGi to expose Web Services. It demonstrates how one can use Eclipse to create, run and debug components. Also the use of Declarative Services is demonstrated. Currently a SOAP Web Service is provided, but in future versions REST interfaces will be added.

Based on this template bundle dummy implementations of the runtime components have been created. These dummy implementations expose the interface points that are described in Section 4 and the Interface Specification Model in D1.2 [1].

The Runtime Template Component, which is based on OSGi bundles, is described in more details in Annex 8.1.

6.3 Implementation of the Aniketos Platform and Environment Components

Following the interface specifications in D1.2 [1] and on Section 4 of this document, an initial implementation of the internal functionalities and their interfaces is already available in the project SVN space at <https://hestia.atc.gr/svn-aniketos/trunk>. The implementation is in an early phase, thus the components are currently stubs that are used to ensure operability and integration between them. Such implementation is necessary at an early stage to provide a realisation on how all these components can be wrapped under the same conditions to provide an integrated view of the Aniketos platform early in advance in the project life cycle.

Table 7: Status overview of the Aniketos platform and environment components implementation

Component Name	Status of implementation
Socio-technical Security Modelling Tool	<ul style="list-style-type: none"> ▪ Basic skeleton on supporting the definition and analysis of goals, resources, socio-technical relationships and security needs ▪ Limited business logic on the internal functionalities ▪ Stub implementation of the interface methods
Model Transformation Module	<ul style="list-style-type: none"> ▪ Stub implementation of a contract template generation ▪ Stub implementation of a security descriptor generation ▪ Stub implementation of the model transformation interfaces
Trustworthiness Component	<ul style="list-style-type: none"> ▪ Stub implementation of the prediction value calculation ▪ Stub implementation of the interface methods
Contract Manager Module	<ul style="list-style-type: none"> ▪ Stub implementation of the method for analysing secure compositions
Property Verification Module	<ul style="list-style-type: none"> ▪ Stub implementation of the method for checking if a service implementation fulfils a certain security/trust property
Composition Security Validation Module	<ul style="list-style-type: none"> ▪ Stub implementation of the methods for validating the service compliance to agreement templates both at design-time and at runtime
Security Property Determination Module	<ul style="list-style-type: none"> ▪ Stub implementation for providing the value of a security property ▪ Sample implementation of a user interface to view and access to defined services ▪ Stub implementation of the interfaces
Secure Composition Planner Module	<ul style="list-style-type: none"> ▪ Stub implementation of the method to wrap a BPMN composition to an XML ▪ Stub implementation of the interface methods and for ordering the compositions ▪ Draft implementation of the method for comparing two compositions
Security Policy Monitoring Module	<ul style="list-style-type: none"> ▪ Stub implementation for getting a security contract and deriving rules

Component Name	Status of implementation
Threat Response Recommendation Module	<ul style="list-style-type: none"> ▪ Stub implementation for managing the check of countermeasures of selected services towards secure specifications and at runtime
Service Threat Monitoring Module	<ul style="list-style-type: none"> ▪ Stub implementation of the binding and unbinding services for threat monitoring ▪ Stub implementation for receiving events for the environment monitoring services ▪ Stub implementation of the respective interfaces
Notification module	<ul style="list-style-type: none"> ▪ Stub implementation of the notification interfaces
Community Support Module	<ul style="list-style-type: none"> ▪ Stub implementation of the internal methods for supporting community-based functionalities ▪ Stub implementation of the respective interfaces
Threat Repository Module	<ul style="list-style-type: none"> ▪ Stub implementation of the internal methods for providing threats information ▪ Stub implementation of the respective interfaces
Marketplace	<ul style="list-style-type: none"> ▪ Stub implementation of the internal methods for announcing and discovering services in the Marketplace and retrieving relevant security descriptors and the respective interfaces ▪ Draft implementation of the front-end interface
Training Material Module	<ul style="list-style-type: none"> ▪ Stub implementation of the internal methods for managing training material ▪ Stub implementation of the respective interfaces
Service Composition Framework	<ul style="list-style-type: none"> ▪ Stub implementation of the specified service composition functionalities and loading BPMN based service compositions
Service Monitoring Component (PRRS)	<ul style="list-style-type: none"> ▪ Initial adaptation of the PRRS tool to serve the Aniketos requirements for monitoring service status on an event-based way
Service Runtime Environment (MUSIC)	<ul style="list-style-type: none"> ▪ Stub implementation of the component for linking the MUSIC runtime environment with Aniketos security monitoring and notification mechanisms ▪ Initial adaptation of the PRRS tool to serve the Aniketos requirements for service reconfiguration
Identity Management Service	<ul style="list-style-type: none"> ▪ Stub implementation of the internal methods for entity registration and verification ▪ Stub implementation of the respective interfaces

It must be noted that the interfaces specifications in this document for the Aniketos components may differ with the descriptions provided in D1.2 [1]. The reason for doing so is the fact that, as the relevant work performed in the other technical WPs (WP1-WP4) progresses, the data models and the respective interface definitions may change. All the updates on these interfaces will be included in the updated version of D1.2, which is Deliverable D1.5 and it is expected on M33.

Thus, the description in D1.2 [1] can be understood as an abstract view of the interface specifications, while the interfaces described in the Aniketos Deliverables D2.1 [9], D2.2, D3.1[10], D3.2 and D4.1

[11] provide the more concrete/technical view” and they have guided the implementation in the context of this deliverable. As D1.2 [1] is a live document, all the architectural aspects will be revised during the course of the project and updated to reflect the actual implementation.

In order to keep track of any inconsistencies raised between the specifications document D1.2 (which will be updated in D1.5 due to M33) and the current document D5.1, as well as the actual implementation steps adopted by all partners, a concrete plan has been specified and will be followed until the final specifications documentation in the forthcoming Aniketos Deliverable D1.5, which is due M33. According to this plan, the interfaces descriptions (as reflected on Section 4) will be maintained as an internal living document, which will be an evolution of Section 4.3 and will be continuously updated as per the requirements of the development phase. The structure of this component will be as follows:

- The name of the component
- The name of the implemented interface
- The name and the description for all the methods of the interface
- For each method, the data model for the input parameters and the return call (or dash “-”, in case that the method is marked as “void”)
- A column describing the dependencies with other Aniketos components
- A reference to D1.2 sequence diagrams, describing the proposed changes to these diagrams

In order to handle and maintain the document, the following processes have been defined:

- When an interface specification changes, the responsible partner makes the necessary modifications to the document
- All the involved partners are notified on the proposed changes in order to assess the impact on their components and the need for additional methods and/or interfaces
- As long as, such changes do not affect other components, the changes to the document are agreed
- In case of conflicts, a task force process should be initialised with the participation of the involved partners, the WP5 leader and the Technical Manager. Based on a majority vote system, the conflicts should be resolved and the document is updated to reflect the outcome of the task force

This process will be active up to M33 when the final specification of the Aniketos architecture should be documented.

7 Conclusions

This deliverable presented the activities for the delivery of a baseline implementation of the Aniketos platform, which acts as the reference point for the actual development of this platform and offers a shared and common background for the Consortium participants on the envisaged technologies that are necessary to build such a platform.

The Aniketos target work constitutes a difficult task, as the integration of various parts has to be efficiently accomplished. As shown in this document, the Aniketos platform is based on many components, many of which act on top of existing environment components, offering the appropriate mechanisms for ensuring that security and trustworthiness is maintained in service composition and execution. As the project is progressing, the specification of the interfaces and their respective methods is subject to changes, which will enable the refinement of the data models exchanged between the Aniketos platform and environment components to conform to the findings and the progress of the work in WP1-WP4.

Through a comprehensive roadmap of internal and external releases of the Aniketos platform, it will be used to build real life scenarios and assessed by the target stakeholders on whether it can effectively support secure service composition at design-time and runtime. Thus, a close interaction with the non-technical WPs is planned in the next months of the projects, which will enable the actual involvement of business partners in the delivery of a high-end solution for addressing security and trustworthiness in service provisioning.

In a nutshell, this document acts as a baseline for further implementation towards the production of the Aniketos components and their smooth integration for facilitating the target functionalities. As described on Section 6.3, future work involves resolving of inconsistencies with the design document D1.2.

References

- [1] Aniketos Consortium, Deliverable D1.2: First Aniketos architecture and requirements specifications, July 2011
- [2] FP7 257930 Aniketos Contract: “Secure and Trustworthy Composite Services”, July 2010
- [3] OASIS, “Reference Model for Service Oriented Architecture 1.0”, August 2006
- [4] Waseem Roshen, “SOA-Based Enterprise Integration: A Step-by-Step Guide to Services-Based Application Integration”, Mc Graw Hill, ISBN: 978-0-07-160553-3, May 2009
- [5] The Open Group, “SOA Reference Architecture”, <http://www.opengroup.org/projects/soa-ref-arch/>, April 2009
- [6] <http://www.w3.org/TR/ws-gloss/>, last accessed 2011-07-20
- [7] “Software development methodology”, http://en.wikipedia.org/wiki/Software_development_methodology, last accessed 2011-07-20
- [8] Wikipedia, “Rapid Application Prototyping”, http://en.wikipedia.org/wiki/Rapid_application_development, last accessed 2011-07-20
- [9] Aniketos Consortium, Deliverable “D2.1: Models and methodologies for embedding and monitoring trust in services”, July 2011
- [10] Aniketos Consortium, Deliverable “D3.1: Design-time support techniques for secure composition and adaptation”, July 2011
- [11] Aniketos Consortium, Deliverable “D4.1: Methods and design for the response to changes and threats”, July 2011
- [12] http://en.wikipedia.org/wiki/Eclipse_%28software%29, last accessed 2011-07-20
- [13] http://en.wikipedia.org/wiki/Equinox_OSGi, last accessed 2011-07-20
- [14] <http://eclipsepluginsite.com/>, last accessed 2011-07-20
- [15] Abdaladhem Albreshne, Patrick Fuhrer, Jacques Pasquier, “Web Services Orchestration and Composition”, September 2009
- [16] Martin Owen, et al., “BPMN and Business Process Management”, Popkin Software white paper, September 2003
- [17] Business Process Model And Notation (BPMN) Specification, Version 2.0, January 2011, <http://www.omg.org/spec/BPMN/2.0/>
- [18] Stephen A. White, “Using BPMN to Model a BPEL Process”, IBM white paper, February 2005.
- [19] OASIS, “Web Services Business Process Execution Language Version 2.0”, April 2007, <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>
- [20] Poornachandra Sarang, Matjaz Juric, Benny Mathew, “Business Process Execution Language for Web Services”, Packt Publishing, ISBN: 978-1-904811-81-7, January 2006
- [21] What is SOA? [An introduction to Service-Oriented Computing](#), last accessed 2011-07-20
- [22] <http://uddi.xml.org/>, last accessed 2011-07-20
- [23] SOA Systems Inc., <http://www.soasystems.com/>, last accessed 2011-07-20
- [24] <http://www.whatissoa.com/soaspecs/ws.php>, last accessed 2011-07-20
- [25] <http://www.whatissoa.com/soaspecs/ws-security.php>, last accessed 2011-07-20

- [26] Claudio Ardagna , Sabrina De , Capitani Vimercati, “Comparison of modelling strategies in defining XML-based access control languages”, *Computer Systems Science and Engineering*, v 19, n 3 (2004)
- [27] <http://www.oasis-open.org/committees/security/>, last accessed 2011-07-20
- [28] <http://openid.net/>, last accessed 2011-07-20
- [29] <http://oauth.net/>, last accessed 2011-07-20
- [30] <http://www.xyzws.com/scdjws/SGS41/3>, last accessed 2011-07-20
- [31] http://fireeagle.yahoo.net/developer/documentation/web_auth, last accessed 2011-07-20
- [32] http://www.googlecodesamples.com/oauth_playground/, last accessed 2011-07-20
- [33] David Chappell, "Enterprise Service Bus", O'Reilly: June 2004, ISBN 0-596-00675-6
- [34] Wikipedia link, Enterprise service bus, http://en.wikipedia.org/wiki/Enterprise_service_bus, last accesses 2011-07-20
- [35] Fiorano Enterprise Service Bus, <http://www.fiorano.com/products/ESB-enterprise-service-bus/ESB-enterprise-service-bus-architecture.php>, last accessed 2011-07-20
- [36] <http://www.mulesoft.org/>, last accesses 2011-07-20
- [37] <http://servicemix.apache.org/home.html>, last accesses 2011-07-20
- [38] James McGovern, Oliver Sims, Ashish Jain, Mark Little (Author), Enterprise Service Oriented Architectures: Concepts, Challenges, Recommendations, Springer; April 28, 2006, ISBN-10: 140203704X
- [39] Leonard Richardson, Sam Ruby, “RESTful Web Services: Web services for the real world”, O'Reilly Media, May 2007, ISBN: 978-0-596-52926-0, ISBN 10: 0-596-52926-0
- [40] <http://www.osgi.org/Main/HomePage>, last accessed 2011-07-20
- [41] An Oracle White Paper, “The Oracle GlassFish Server Advantage for Small Businesses”, June 2010
- [42] Oracle GlassFish Server, <http://www.oracle.com/us/products/middleware/application-server/oracle-glassfish-server/index.html>, last accessed 2011-07-20
- [43] <http://felix.apache.org/site/index.html>, last accessed 2011-07-20
- [44] <http://www.mysql.com/>, last accessed 2011-07-20
- [45] <http://www.postgresql.org/>, last accessed 2011-07-20
- [46] <http://www.mongodb.org/>, last accessed 2011-07-20
- [47] <http://www.hibernate.org/>, last accessed 2011-07-20
- [48] D. Fotakis and S. Gritzalis, “Efficient heuristic algorithms for correcting the Cascade Vulnerability Problem for interconnected networks,” *Computer Communications*, vol. 29, 2006
- [49] <http://drupal.org/>, last accessed 2011-07-20

8 Annexes

8.1 Runtime Template Component

The Aniketos Runtime Template Component is based on OSGi technology and is distributed as a set of OSGi bundles. It uses the Apache CXF implementation of Distributed OSGi to expose Web Services. It demonstrates how one can use Eclipse to create, run and debug components. Also the use of Declarative Services is demonstrated. Currently a SOAP Web Service is provided, but in future versions REST interfaces will be added.

The following instructions are based on the following two sources:

- [OSGi with Eclipse Equinox - Tutorial](#)
- [Apache CXF - DOSGi DS Demo page](#)

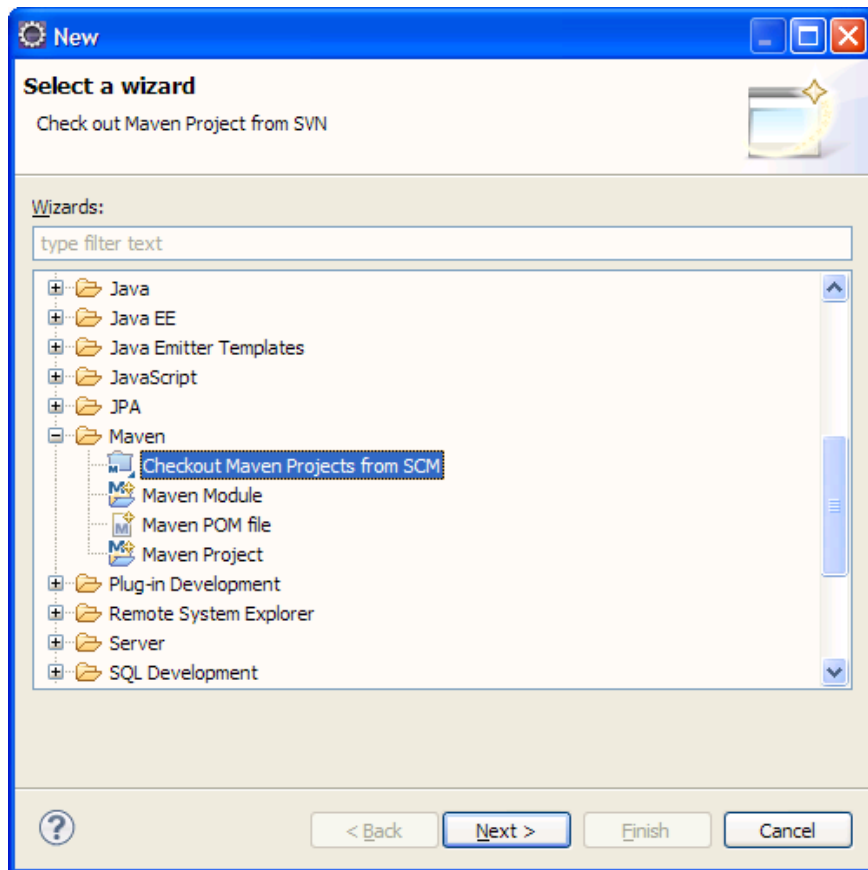
8.1.1 Pre-requisites

In order to install and test the Aniketos Runtime Template Component the following software components are needed:

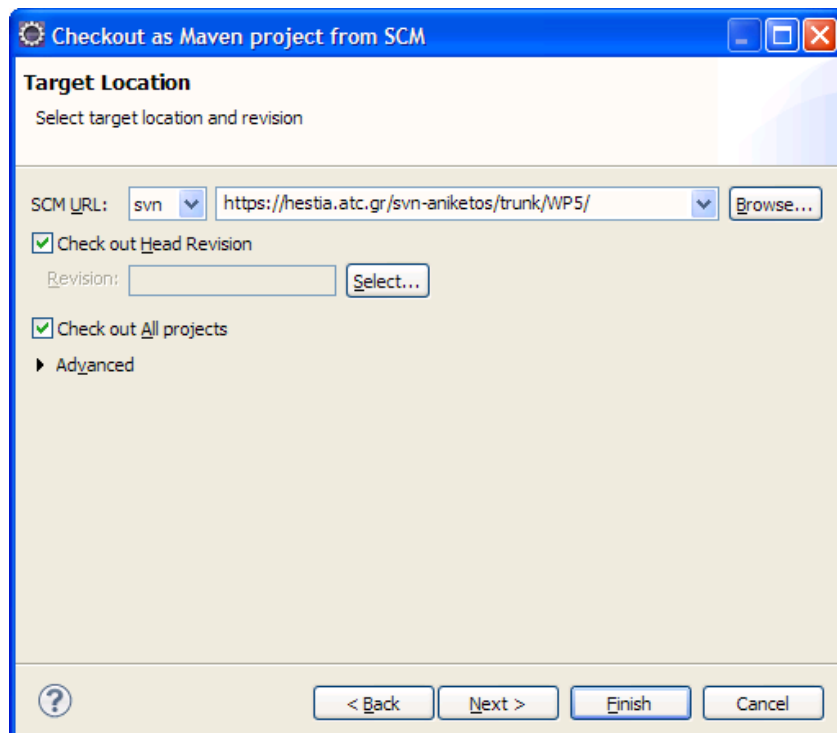
Component	Description
Eclipse 3.6 IDE	Eclipse IDE will be used for development as well as the OSGi framework (Equinox) for testing
Apache CXF DOSGi	Instructions for installing CXF DOSGi will be provided

8.1.2 Downloading

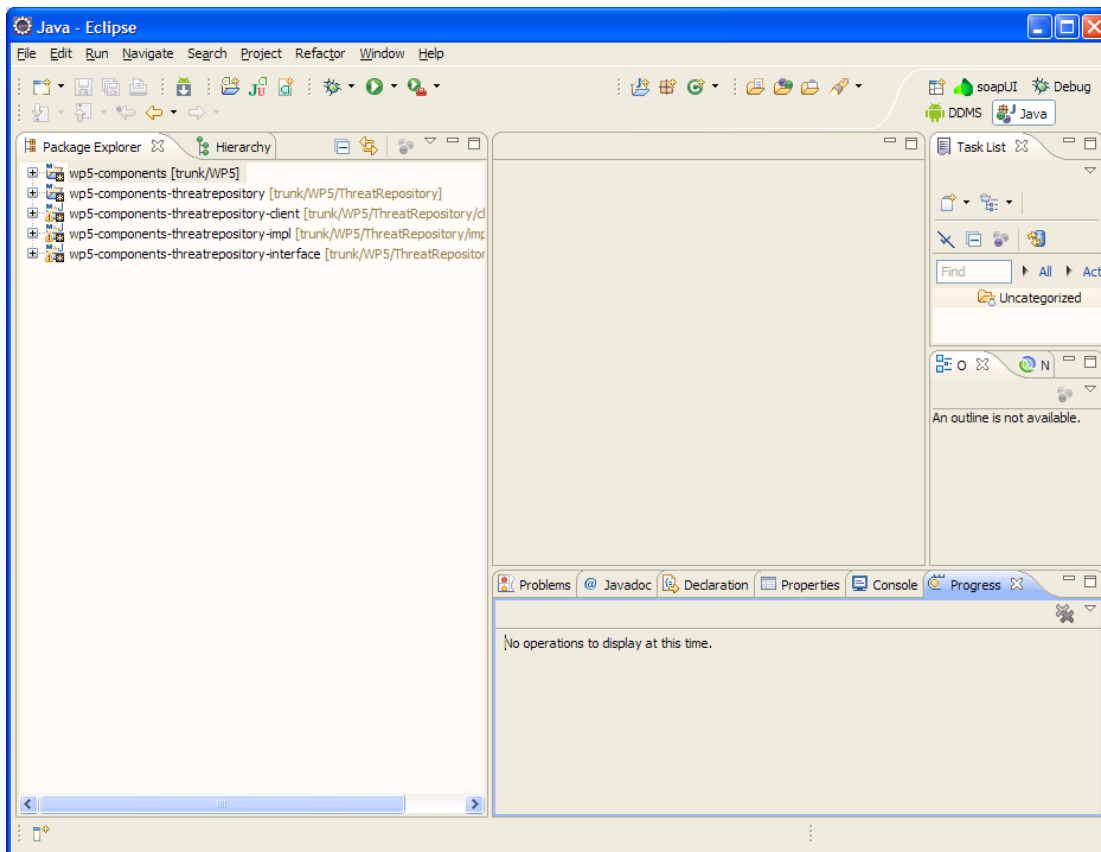
The Aniketos Runtime Template Component is available at the SVN (<https://hestia.atc.gr/svn-aniketos/trunk/WP5/>). You can check out the whole WP5 folder and build it using Maven. However, it will be easier to use Eclipse IDE. From the file menu select “New” and then “Other”.



Then select “Checkout Maven Projects from SCM”. In the next dialog enter the address of Aniketos SVN.



You can now click finish and the required projects will be downloaded imported into Eclipse.



The Aniketos Runtime Template Component presents a dummy implementation of the Threat Repository Module. The Threat Repository Module consists of two bundles:

- wp5-components-threatrepository-interface
- wp5-components-threatrepository-impl

A third bundle that demonstrates a client is also provided. To build all three bundles right click at the wp5-components-threatrepository project and select “Run as” and then “Maven install”. The corresponding jar files will be created in the Maven local repository (\$UserDirectory/.m2/repository).

8.1.3 Running

The component consists of three OSGi bundles:

- Service Interface
- Service Implementation
- Service Client

All of them are created using Eclipse. To create the service interface bundle select New, Project, Plugin Project in Eclipse. Name the project: eu.aniketos.wp5.components.threatrepository and select Standard as OSGi Framework.

In the following steps choose not to generate an activator and don’t use a template.

New Plug-in Project

Plug-in Project
Create a new plug-in project

Project name:

Use default location
Location: Browse...

Choose file system:

Project Settings

Create a Java project

Source folder:

Output folder:

Target Platform
This plug-in is targeted to run with:

Eclipse version:

an OSGi framework:

Working sets

Add project to working sets

Working sets: Select...

< Back Next > Finish Cancel

New Plug-in Project

Content
Enter the data required to generate the plug-in.

Properties

ID:

Version:

Name:

Provider:

Execution Environment: Environments...

Options

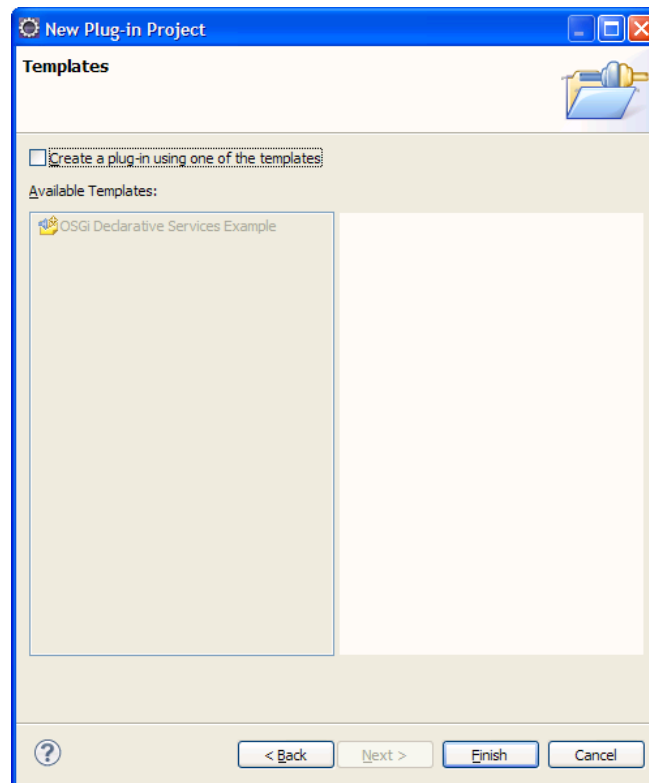
Generate an activator, a Java class that controls the plug-in's life cycle

Activator:

This plug-in will make contributions to the UI

Enable API Analysis

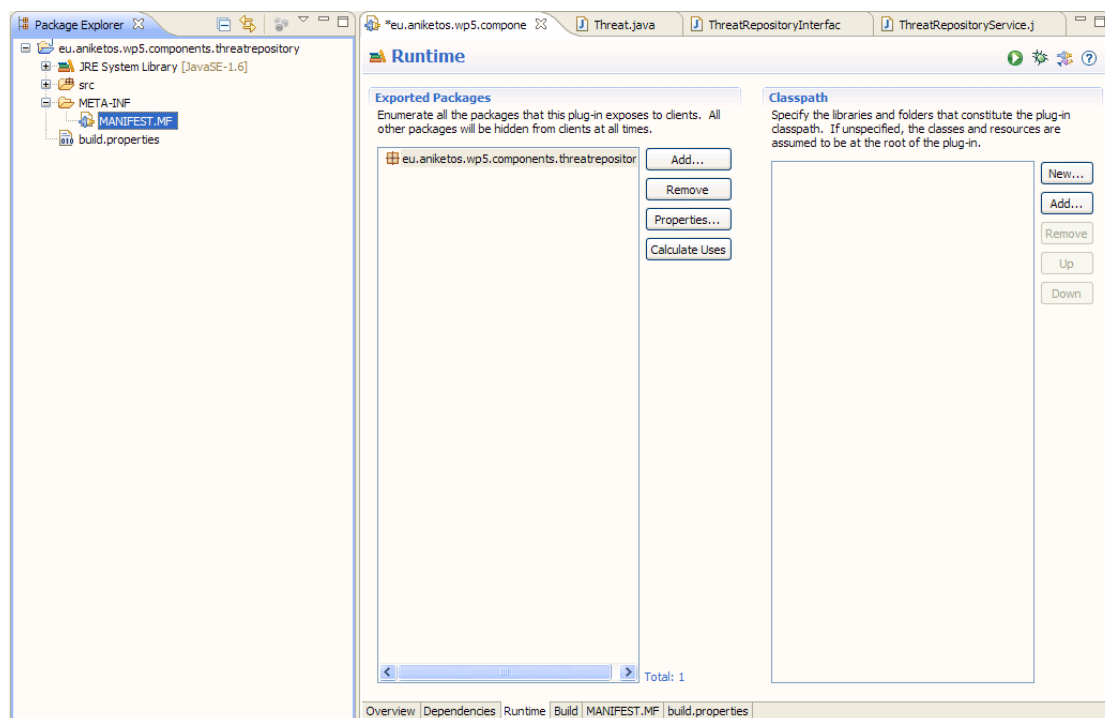
? < Back Next > Finish Cancel



Then add to the project the following classes:

- `eu.aniketos.wp5.components.threatrepository.Threat`
- `eu.aniketos.wp5.components.threatrepository.ThreatRepositoryService`

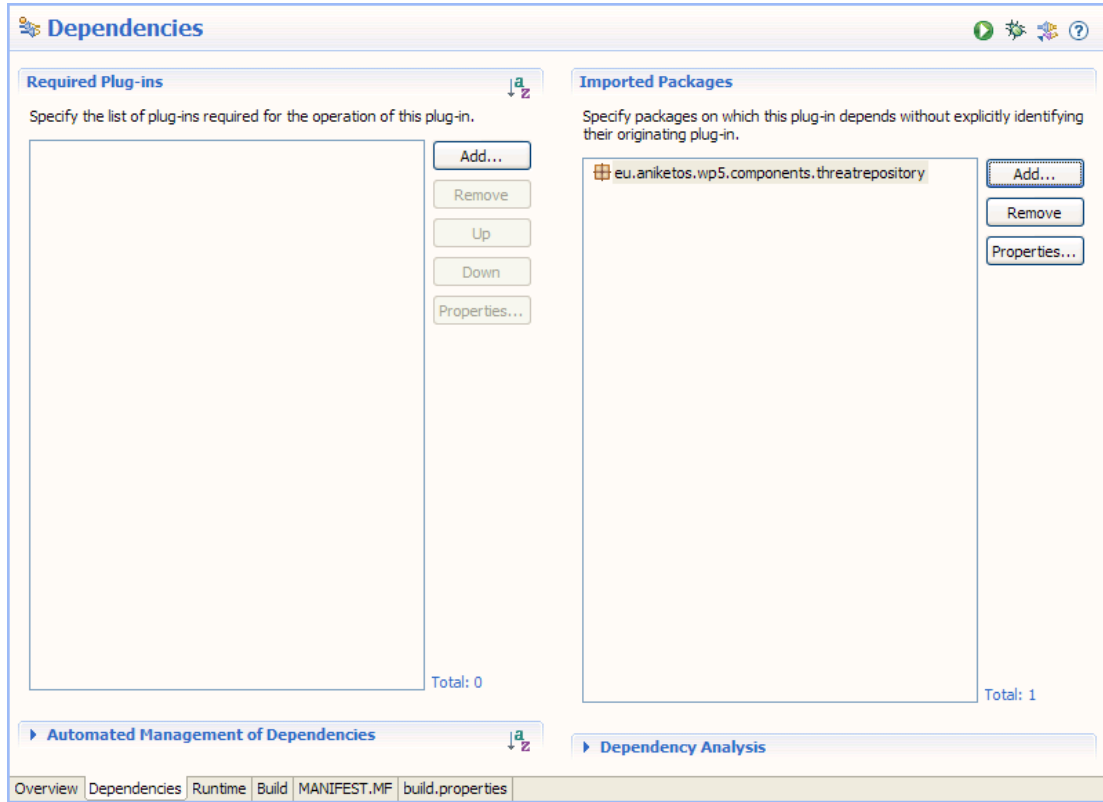
Then double click the `MANIFEST.MF` file and add `eu.aniketos.wp5.components.threatrepository` as an exported package.



8.1.4 Service Implementation

Create a new plug-in as before. Name it `eu.aniketos.wp5.components.threatrepository.impl` and do not use an Activator or a

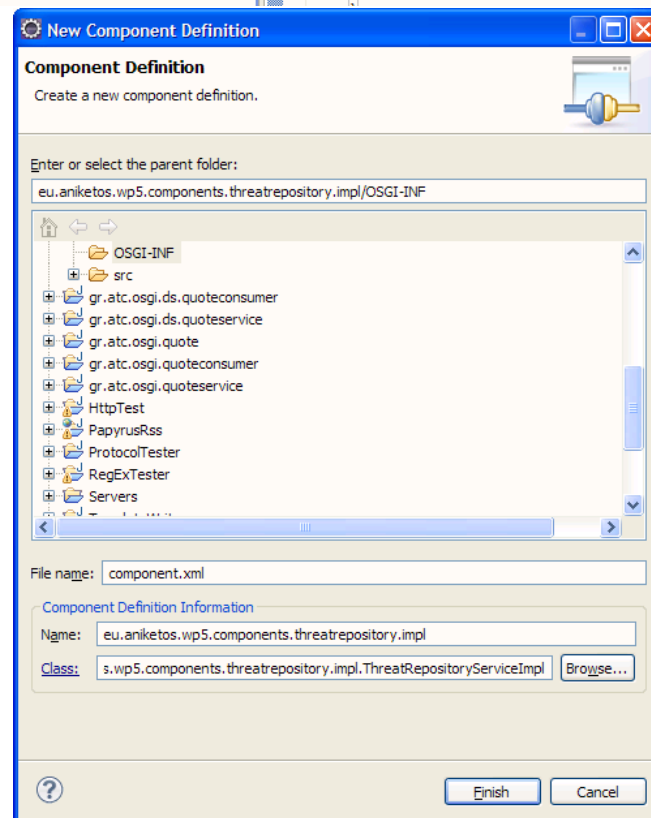
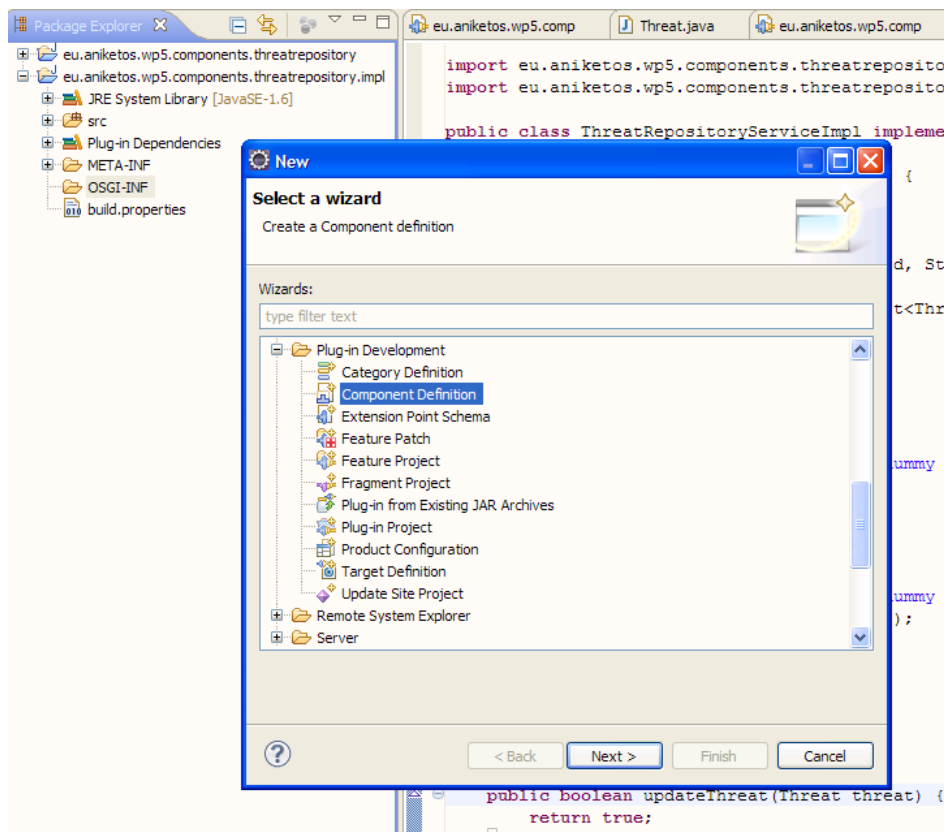
template. Then double click the MANIFEST.MF file and add `eu.aniketos.wp5.components.threatrepository` as an imported package.



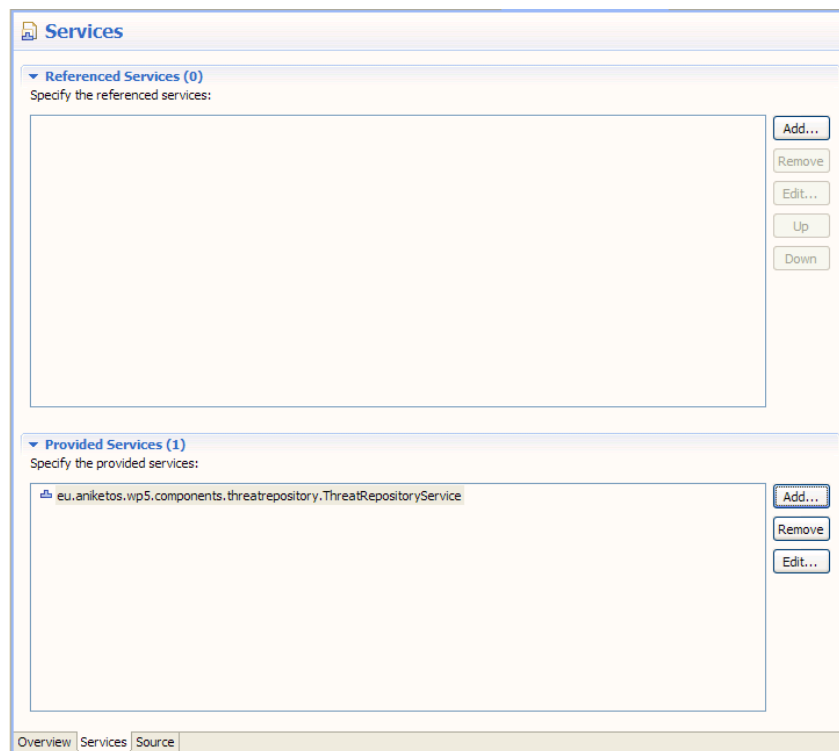
Add the `ThreatRepositoryServiceImpl` class in the project (the same class as in the previous version of the tutorial). Then create an OSGI-INF folder, right click on it and add a component definition.

In the `build.properties` file enter:

```
eu.aniketos.wp5.components.threatrepository.impl.ThreatRepositoryServiceImpl
```

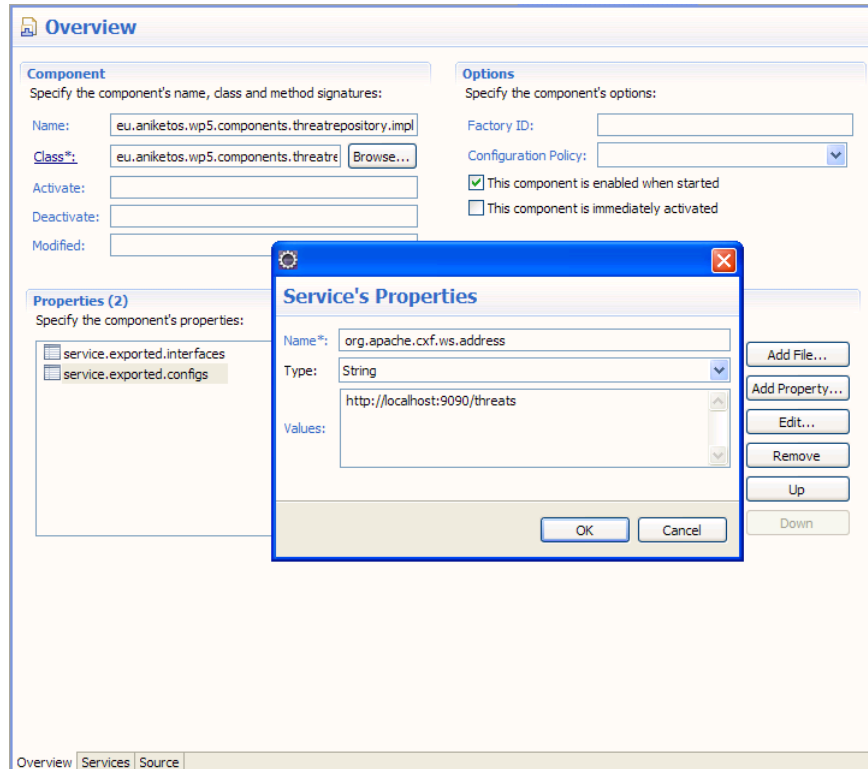


At the services tab add the ThreatRepositoryService as a provided service.



Then go at the overview tab and add the following properties:

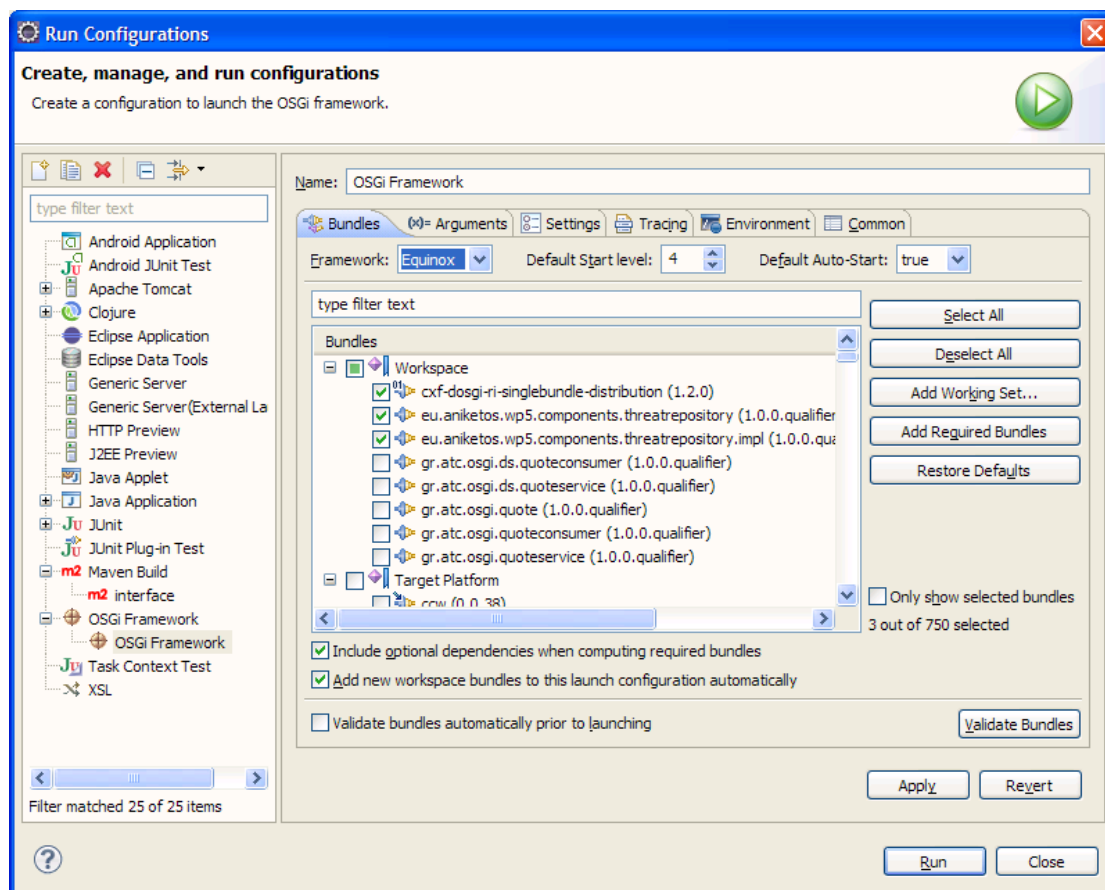
- `service.exported.interfaces = *`
- `service.exported.configs = org.apache.cxf.ws`
- `org.apache.cxf.ws.address = http://localhost:9090/threats`



The source of the component.xml file should look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0"
name="eu.aniketos.wp5.components.threatrepository.impl">
  <implementation
class="eu.aniketos.wp5.components.threatrepository.impl.ThreatRepositorySer
viceImpl"/>
  <service>
    <provide
interface="eu.aniketos.wp5.components.threatrepository.ThreatRepositoryServ
ice"/>
  </service>
  <property name="service.exported.interfaces" type="String" value="*"/>
  <property name="service.exported.configs" type="String"
value="org.apache.cxf.ws"/>
  <property name="org.apache.cxf.ws.address" type="String"
value="http://localhost:9090/threats"/>
</scr:component>
```

In order to test the service, we need first to import the CXF DOSGI single bundle distribution into the workspace. You can do that by downloading the file and then choosing Import, plug-ins and fragments from Eclipse. We can then create a run configuration in order to run the two modules in Equinox.

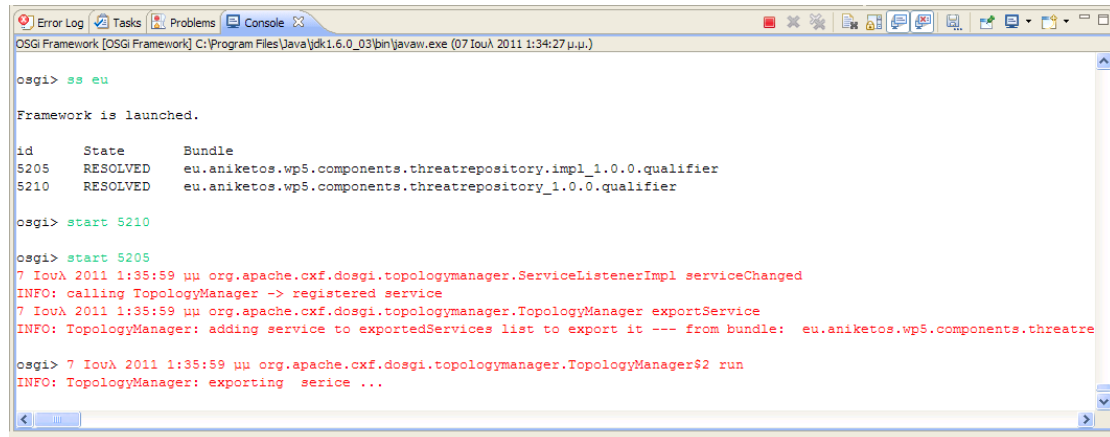


From the Target Platform, the following bundles are needed:

- org.eclipse.equinox.ds
- org.eclipse.equinox.util

All other bundles can be unselected. From the Workspace select all bundles. Then click "Add Required Bundles" and then "Run" to launch Equinox framework.

You can use the Console to control the framework. Type `ss eu` to see all bundles, which have `eu` in their name. Locate the two Aniketos bundles and start them. Some logging messages will appear. If everything went smoothly, then you should be able to see the WSDL file at <http://localhost:9090/threats?wsdl>.



```

OSGi Framework [OSGi Framework] C:\Program Files\Java\jdk1.6.0_03\bin\javaw.exe (07 Iou\ 2011 11:34:27 μ.μ.)

osgi> ss eu

Framework is launched.

id      State      Bundle
5205    RESOLVED   eu.aniketos.wp5.components.threatrepository.impl_1.0.0.qualifier
5210    RESOLVED   eu.aniketos.wp5.components.threatrepository_1.0.0.qualifier

osgi> start 5210

osgi> start 5205
7 Iou\ 2011 1:35:59 μμ org.apache.cxf.dosgi.topologymanager.ServiceListenerImpl serviceChanged
INFO: calling TopologyManager -> registered service
7 Iou\ 2011 1:35:59 μμ org.apache.cxf.dosgi.topologymanager.TopologyManager exportService
INFO: TopologyManager: adding service to exportedServices list to export it --- from bundle: eu.aniketos.wp5.components.threatre
osgi> 7 Iou\ 2011 1:35:59 μμ org.apache.cxf.dosgi.topologymanager.TopologyManager$2 run
INFO: TopologyManager: exporting service ...
  
```

8.2 Template Eclipse Plugin and OSGI Component(s)

The Aniketos Eclipse plugin uses Eclipse plugin development environment (PDE) and OSGI technology. It is distributed as a set of Eclipse plugin and OSGI bundles. This dummy plugin is intended to demonstrate:

- The basics of creating an Eclipse plugin that contributes to the UI
- Packaging a plugin for installation within the Eclipse workbench
- Creating and registering an Eclipse View object
- Creating an OSGI-compatible Eclipse plugin that can locate and communicate with other OSGI bundles running in the same Equinox instance, comprising:
 - Using the Bundle's Activator to initialise OSGI services that you're making available to others
 - Using the Bundle's context to get a reference to OSGI services you want to use
- Starting an Eclipse runtime environment that includes these plugins and bundles

The client plugin uses OSGI methods to locate and invoke a method on a separate bundle. Currently, it uses a default context (Equinox), and relies on the platform being correctly configured in order to locate, install, and start the relevant bundles. Full configuration details for a suitable Eclipse platform are included. A future version of this plugin will aim to incorporate some of the WP5 work on D-OSGI.

Some other web-based references that you might find helpful are:

<http://www.vogella.de/articles/EclipsePlugIn/article.html> - vogella.de plugin tutorial

<http://www.vogella.de/articles/EclipseExtensionPoint/article.html> - vogella.de plugin extension point tutorial

<http://www.vogella.de/articles/OSGi/article.html> - vogella.de OSGI in Equinox tutorial

<http://www.vogella.de/articles/EGit/article.html> - vogella.de EGIT (Eclipse GIT client) tutorial

MDT BPMN2:

<http://wiki.eclipse.org/MDT-BPMN2> - the Eclipse wiki for MDT-BPMN2 (the BPMN library used for one of the OSGI bundles).

This wiki entry references several useful tutorials from SAP for MDT-BPMN2, which are very useful to the newcomer to EGIT or BPMN2 and worth a read. A link to the first of these is shown below:-

<http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/c04f0691-0a76-2d10-1098-ec518f7bdf68>

Note: We plan to revisit this plugin collection to add enhanced UI functionality, alternative OSGI service registration and location, and D-OSGI service registration and location. If you have any problems installing or running it, please do not hesitate to contact the LJMU team.

8.2.1 Pre-requisites

In order to install and test the dummy plugin and components the following software is required:

Component	Description
Eclipse 3.6 IDE	The latest version of Eclipse IDE
A suitable SVN / Eclipse plugin	You can choose either of the following...
Either: subclipse	A plug-in that provides SVN functionality within Eclipse
Or: Subversive	A plug-in that provides SVN functionality to Eclipse (author's choice!)
http://www.eclipse.org/egit/	EGIT - A plug-in that provides GIT functionality for eclipse
http://git.eclipse.org/c/bpmn2/	The org.eclipse.bpmn2 libraries (obtain with GIT client)

Please note, you only require ONE of the SVN plugins – they both have popular followings and integrate SVN functionality with Eclipse. If you already have a suitable SVN plugin, do not feel that you have to replace it with one of those above.

You require GIT in order to download the BPMN2 project from Eclipse.org. However, if you want to obtain the libraries and simply include a bpmn2 JAR, you are welcome to. The following instructions assume that you've downloaded the BPMN2 source and have it installed as projects in your workspace.

Some other Aniketos tutorials/descriptions use GIT for source control, so it may be worthwhile to install a client now.

8.2.2 Getting started

The overall process you should follow to get everything running is best described as:-

1. Installing Eclipse
2. Installing required utility Eclipse plugins (SVN, EGIT, etc)
3. Installing dependencies (BPMN2)
4. Checking out the dummy projects (eu.aniketos.wp3.dummy.scp.*)
5. Creating a new "Eclipse Application" Launch configuration

Installing Eclipse is straightforward, and the distribution pages for the required plugins detail the installation process, so this will not be duplicated here. As such, this document will give a brief overview of installing the dependencies, but will primarily focus on getting our dummy code (eu.aniketos.wp3.dummy.scp.*) checked out, and getting it running.

8.2.3 Installing BPMN2

The author makes the assumption that at this point, you have:

- A working JDK and Eclipse installation

- Chosen and installed a suitable SVN Eclipse plugin and any native/Java drives
- Installed a GIT client and the required integrator/connector to your chosen SVN plugin

The SAP guide at <http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/c04f0691-0a76-2d10-1098-ec518f7bdf68?QuickLink=index&overridelayout=true> gives a really good overview of how to install BPMN2, along with installing a GIT client, so we will not reproduce too much of this information here. You should follow SAP's tutorial as far as step 8 (page 5) (i.e. you do not need to run it)

If you are familiar with EGIT and do not need the walkthrough provided by SAP, the key details from the SAP tutorial are:

- <http://git.eclipse.org/c/bpmn2/> - GIT location URI
- You should import all org.eclipse.bpmn2 projects (excluding org.eclipse.bpmn2.ecore, which has likely been deleted by now)

Either way, the full checkout is likely to take some time. Please be patient!

8.2.4 Obtaining our (eu.aniketos.wp3.dummy.*) plugins and bundles

The source code for our plugins and bundles is located on the Aniketos SVN server. The repository contains four projects, but the current version of this tutorial only deals with three of them. We plan to integrate the fourth project in a future version of this tutorial. The projects are:

- eu.aniketos.wp3.dummy.scp - is the secure planner interface bundle. It's a placeholder for the WP5 interfaces for the secure composition planner. At present, it's not strictly WP5 compliant and is more for demonstration purposes! (in fact, it's just one method)
- eu.aniketos.wp3.dummy.scp.mdtimplementation - This is our simplified BPMN comparison bundle. It uses org.eclipse.bpmn2 libraries as described in the SAP tutorial and simply returns the BPMN string with the greatest number of tasks. The activator registers a new instance of the comparison utility, and makes it available as a service on the OSGI context.
- eu.aniketos.wp3.dummy.scp.client - This is the client plugin. It provides a simple Eclipse View, which finds a suitable OSGI service for BPMN comparison
- eu.aniketos.wp3.dummy.scp.activitiengine - This is an alternative comparison tool to mdtimplementation. It uses the activitiengine BPMN libraries. It is still in development, and may be added in the future as an alternative service provider.

To get these projects, you'll need to check them out from the SVN server. The location on the SVN server is:

<https://hestia.atc.gr/svn-aniketos/trunk/WP3/dummy>

You should check out each of the eu.aniketos.wp3.dummy.* folders as an Eclipse project. To check out an SVN project, right click in the project explorer Window, and select Import. The following dialogue box will open, as shown below. Expand the SVN folder and select Project from SVN.

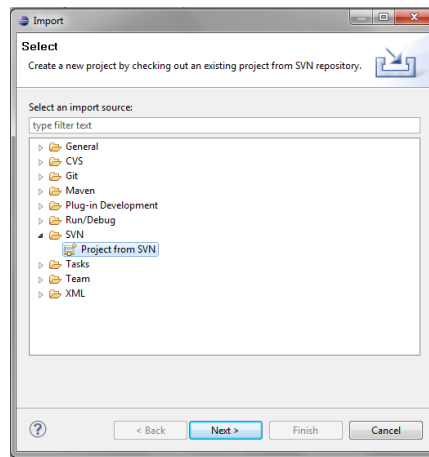


Figure 30: Import project dialogue box

Click Next, and you will be prompted for the SVN information. Enter the Aniketos SVN server information (<https://hestia.atc.gr/svn-aniketos/trunk/WP3/dummy>) along with your Aniketos SVN username and password, as shown in Figure 31.

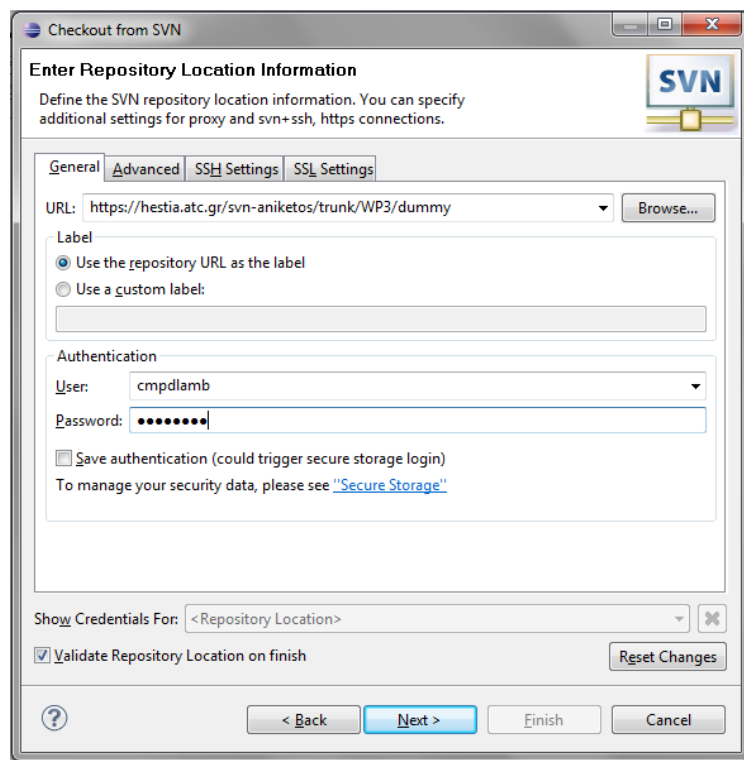


Figure 31: SVN Repository Information

Click Next, then Finish. After a short time, the Eclipse “Check Out As...” wizard will appear. Select “Find Projects in the children of the selected resource”, and click Finish

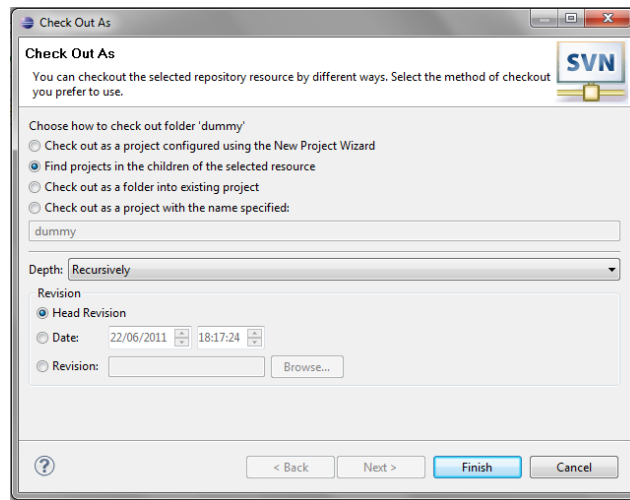


Figure 32: Eclipse "Check out as" project wizard

Eclipse will now contact the SVN server again and determine which children projects exist in the folder. After a short time, a dialogue box will appear showing it has found four projects (Figure 33). If on a slow connection, please uncheck the eu.aniketos.wp3.dummy.scp.activitiengine project (we don't use this in the present version of this document). Click Finish, and the projects will be checked out.

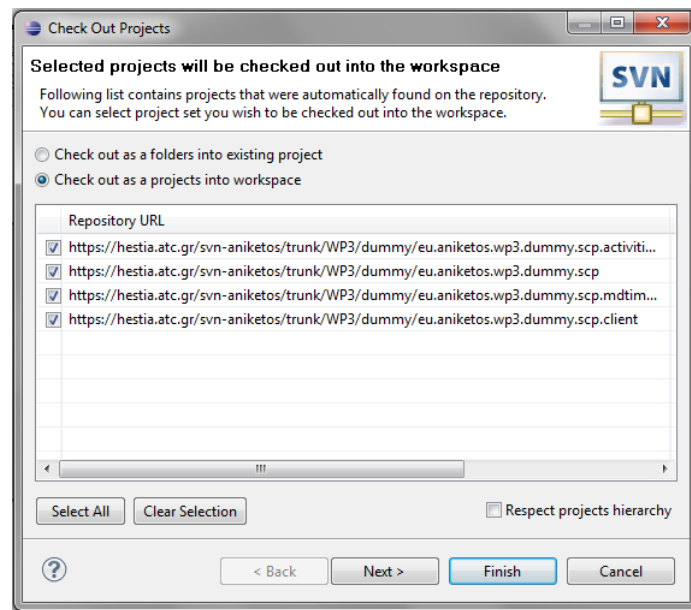


Figure 33: Check out project selection

The projects should now check out to your Eclipse workspace and build successfully. If they do not, please double check you have the BPMN2 project installed (from GIT), or a suitable BPMN2 JAR on your classpath. If you do have the BPMN2 project installed, and it still doesn't build successfully, then try cleaning and rebuilding the workspace (Project Menu → Click Clean...)

8.2.5 Executing the plugin and bundles

Once you've checked out all of the code into your Eclipse workbench, you're ready to start the bundles. We'll use a new Eclipse runtime configuration for this. Click Run -> Run Configurations... Select Eclipse Application, and click the New Launch Configuration and then give the new launch configuration an appropriate name. Click the plugins tab (see Figure 34)

Note: A future version of this tutorial will look at exporting the plugins as OSGI bundles, deploying the “server” bundles on a separate OSGI container, and using D-OSGI in the client to communicate with the “server” bundle.

When the new Eclipse instance loads, select the Window menu, and then click ‘Show View->Other...’. Expand the ‘Other’ folder, and select SCP View. This will show our client plugin’s view.

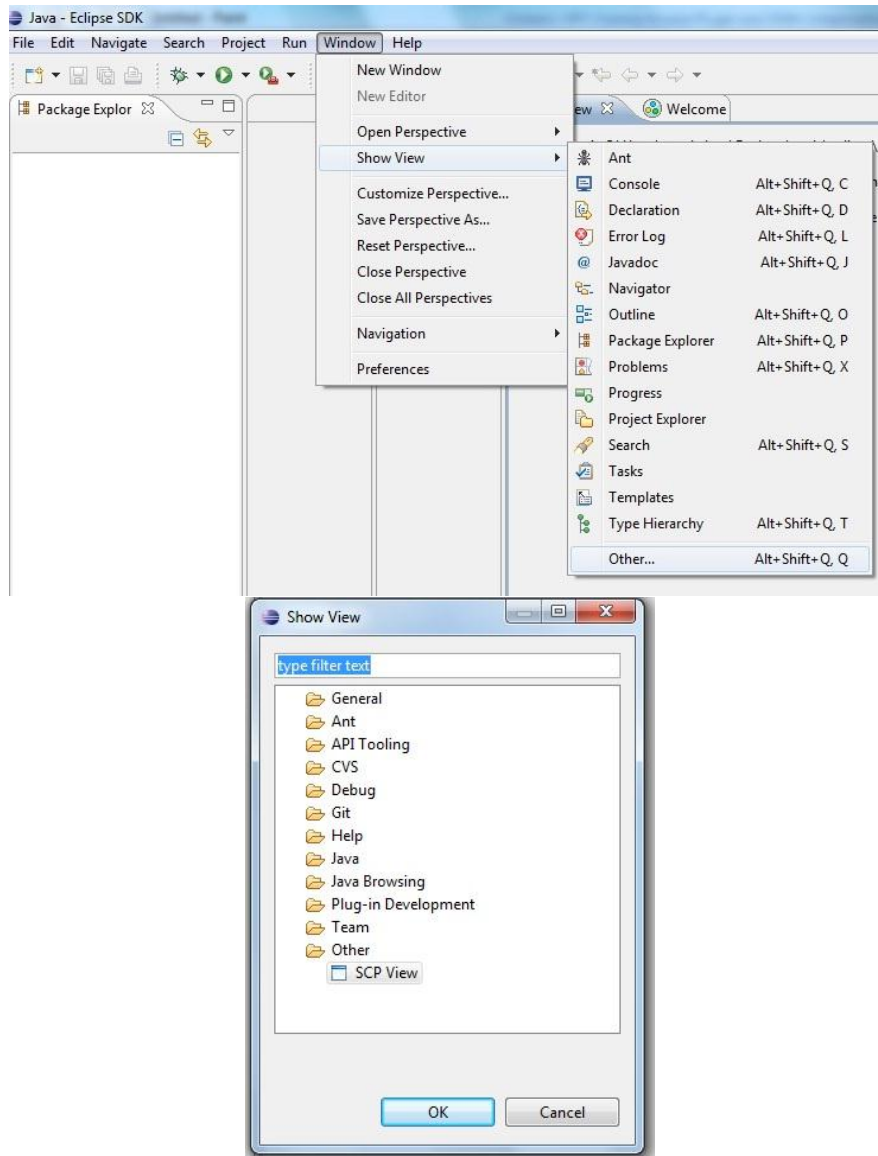


Figure 36: Show the View panel created by scp.client

You can then click the buttons to choose plans 1 and 2 (it is expecting BPMN2 in XML format). There are two very simple test files included in the client project’s testBPMN directory.

Once you’ve selected the files, you can then click “Compare them for me”. It will produce some feedback through the UI, though the bundle location and invocation is tracked by calls to System.out.println(), and will appear in the console window of the original, development Eclipse. As with any other Eclipse application, the Eclipse Application launch can be run in debug mode, allowing you to step line-by-line through the execution. The following figures show the sort of screen output you can expect.

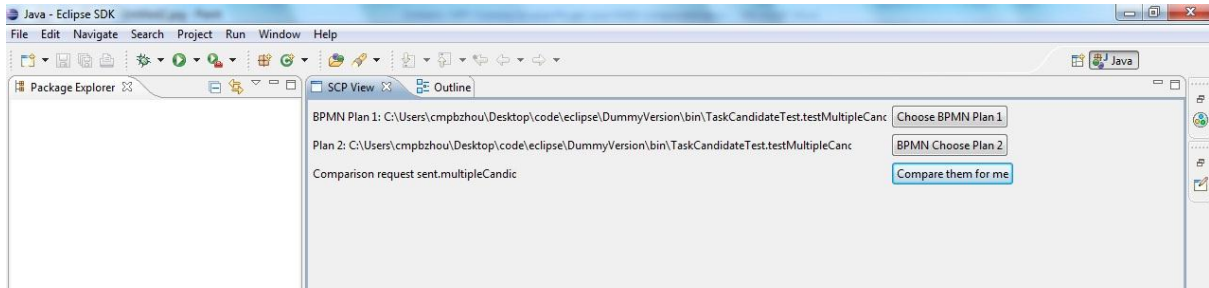


Figure 37: The user interface of the template in runtime Eclipse

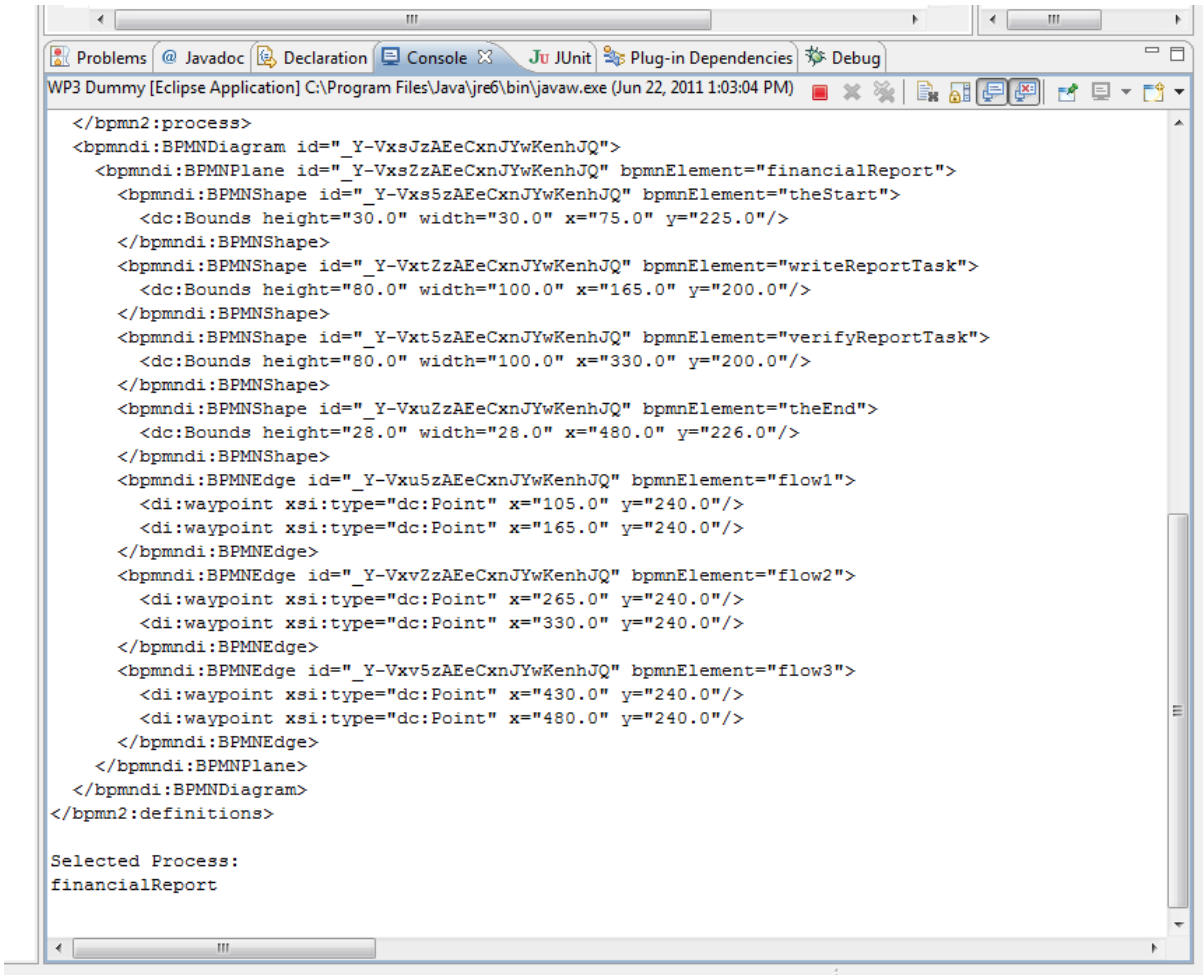


Figure 38: Output of the template shown in console of development Eclipse