284928

Collaborative Manufacturing Network for Competitive Advantage

D3.5.1 Guidelines for the secure collaboration model (public)
<table>
<thead>
<tr>
<th>Grant Agreement No.</th>
<th>284928</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym</td>
<td>ComVantage</td>
</tr>
<tr>
<td>Project title</td>
<td>Collaborative Manufacturing Network for Competitive Advantage</td>
</tr>
<tr>
<td>Deliverable number</td>
<td>D3.5.1</td>
</tr>
<tr>
<td>Deliverable name</td>
<td>Guidelines for the secure collaboration model</td>
</tr>
<tr>
<td>Work package</td>
<td>WP 3 – Secure Information Model</td>
</tr>
<tr>
<td>Lead beneficiary</td>
<td>NEXTEL</td>
</tr>
<tr>
<td>Authors</td>
<td>Oscar Lopez (NEXTEL), Manuel Carnerero (NEXTEL), Mikel Uriarte (NEXTEL), Etxahun Sanchez (NEXTEL), Oscar Lázaro (INNO), Patricia Ortiz (INNO), Francis Jeandel (EVIDIAN), Dimitris Karagiannis (UNIVIE), Robert Buchmann (UNIVIE), Patrik Burzynski (UNIVIE), Jasmin Brakmic (UNIVIE), David Orensanz (BOC IB), Oscar Pardo (BOC-IB), Sergio Guijarro (BOC-IB)</td>
</tr>
<tr>
<td>Peer Reviewers</td>
<td>David Orensanz (BOC), Georg Suess (K&amp;A)</td>
</tr>
<tr>
<td>Nature</td>
<td>R – Report</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>PU – Public</td>
</tr>
<tr>
<td>Delivery date</td>
<td>31/08/2013 (M24)</td>
</tr>
</tbody>
</table>
Executive Summary

The objective of this deliverable *D3.5.1 Guidelines for the secure collaboration model* is to conceptualise the results to provide a comprehensive and coherent set of guidelines for secure collaboration model within the ComVantage project. In this document the whole security model architecture is defined as well as its basic operation.

In order to define such guidelines for the secure collaboration model, the deliverable has been structured in different sections. First of all, section 2 provides a description of the security prototype that has been developed. The general operation and architecture of the security framework is presented. The authentication process is also defined in this section, as well as both authorisation processes that will be executed depending on the type of information requested. This section also presents some specifications for the business model tool and for the secure information model.

Afterwards, section 3 presents the guidelines for using the modelling tool support resulted from task 3.4, including modelling with the OMI prototype, which has been provided through the Open Model Initiative community space.

To sum up, all the different analysis performed in this deliverable have led to the definition of a enhanced multi-tiered access control framework which will be able to protect all data sets (RDF data, transient data and data in traditional formats) in the collaboration network, helping to ward off threats and eliminate vulnerabilities while proving compliance and maximizing the efficiency of the operations.
Table of Contents

1 OVERVIEW .............................................................................................................................................. 9
  1.1 INTRODUCTION ................................................................................................................................ 9
  1.2 SCOPE OF THIS DOCUMENT ............................................................................................................. 10
  1.3 RELATED DOCUMENTS .................................................................................................................... 10

2 DESCRIPTION OF THE SECURITY PROTOTYPE .............................................................................. 12
  2.1 MULTI-DOMAIN MOBILE ENVIRONMENT’S SECURITY ASPECTS .................................................. 12
  2.2 GENERAL OPERATION AND ARCHITECTURE OF THE SECURITY FRAMEWORK ...................... 13
    2.2.1 General operation ....................................................................................................................... 13
    2.2.2 Architecture of the security framework .................................................................................... 16
  2.3 AUTHENTICATION PROCESS ............................................................................................................ 17
    2.3.1 Main architectural entities involved in the authentication process ........................................... 18
    2.3.2 Description of the authentication process in a non browser environment ............................. 19
  2.4 AUTHORISATION PROCESS .............................................................................................................. 21
    2.4.1 Authorisation process based on SPARQL query rewriting ....................................................... 22
    2.4.2 Authorisation process based on XACML .................................................................................. 33

3 MODELLING GUIDELINES ..................................................................................................................... 40
  3.1 OMI PROTOTYPE GUIDELINES ......................................................................................................... 41
    3.1.1 General User Interface .............................................................................................................. 41
    3.1.2 ComVantage Modelling Guidelines ........................................................................................... 49
  3.2 THE COMVANTAGE MODELLING PROTOTYPE GUIDELINES ...................................................... 77
    3.2.1 Object Repository ..................................................................................................................... 77
    3.2.2 Model type Guidelines .............................................................................................................. 80

4 CONCLUSIONS AND OUTLOOK ........................................................................................................... 111
  4.1 CONCLUSION ...................................................................................................................................... 111
  4.2 OUTLOOK AND OPEN ISSUES ....................................................................................................... 111

5 GLOSSARY .............................................................................................................................................. 112

6 REFERENCES .......................................................................................................................................... 115
# List of Figures

Figure 1: Related Deliverables to D3.5.1 ................................................................. 11
Figure 2: Collaboration through Linked Data example scenario .................................. 14
Figure 3: Access-control solution applied to the example scenario ......................... 15
Figure 4: Access-control solution - schematic design ............................................ 15
Figure 5: Combined AC approach to enable secure mobile collaboration through the Web of Data .......................................................... 17
Figure 6: SAML IdP ......................................................................................... 19
Figure 7: Authentication example between two companies .................................... 20
Figure 8: Cross domain authentication workflow .................................................... 21
Figure 9: Authorisation process based on an out of band and a runtime process ........ 22
Figure 10: Linked Data example ......................................................................... 23
Figure 11: Views creating process ........................................................................ 24
Figure 12: Example of original Linked Data .............................................................. 24
Figure 13: Example public view inferred ................................................................ 27
Figure 14: Example OPERATORS view inferred ..................................................... 29
Figure 15: SPARQL query launching and rewriting ................................................ 30
Figure 16: SPARQL rewriting process .................................................................. 31
Figure 17: Original SPARQL query ...................................................................... 31
Figure 18: Rewritten SPARQL query ..................................................................... 32
Figure 19: XACML architecture .......................................................................... 34
Figure 20: XACML context .................................................................................. 36
Figure 21: Prototypes general user interface ........................................................... 41
Figure 22: Available components .......................................................................... 41
Figure 23: Create a new Process model ................................................................. 42
Figure 24: Modelling elements bar for process model type ..................................... 43
Figure 25: An edge with bend points .................................................................... 43
Figure 26: Change model size ............................................................................. 44
Figure 27: Example for a Notebook ..................................................................... 44
Figure 28: Attribute actions ................................................................................ 45
Figure 29: Traverse a relation without a notation in the opposite direction ............ 45
Figure 30: Open query dialog ............................................................................. 46
Figure 31: Dialog for building simple queries ....................................................... 46
Figure 32: Example query result ........................................................................... 47
Figure 33: Start XML export ................................................................................ 47
Figure 34: XML export model selection ............................................................... 48
Figure 35: XML export file selection .................................................................... 48
Figure 36: XML import model-group selection ..................................................... 49
Figure 37: Example for Roles in Organizations ................................................... 51
Figure 38: Example WP6 process of multiple roles collaborating ......................... 52
Figure 39: Right and wrong parallelisms ............................................................... 52
Figure 80: Event aggregation options ................................................................. 86
Figure 81: Different graphical representation options for this concept ......................................................... 86
Figure 82: Representation options panel ................................................................................................. 86
Figure 83: Horizontal/Vertical Swinlanes .................................................................................................. 87
Figure 84: Notebook options to change orientation in Swinlane concept .................................................. 87
Figure 85: Example of Process model ........................................................................................................ 88
Figure 86: Product default representation and example alternative .............................................................. 88
Figure 87: Product options to change default graphical representation ...................................................... 89
Figure 88: Referenced activities from a Product concept ............................................................................... 89
Figure 89: Different Customization Features according to its type .............................................................. 89
Figure 90: Different Customization Features according to its Kano’s type .................................................. 90
Figure 91: Disabled Customization Feature .................................................................................................. 90
Figure 92: Customization Feature Notebook .............................................................................................. 90
Figure 93: Example of Product Configuration, based on a T-shirt product ................................................ 91
Figure 94: Market segment concepts. Right representation shows the % of share market ................................ 91
Figure 95: Market segment Notebook options .............................................................................................. 92
Figure 96: Different representations of Factor concept .................................................................................. 92
Figure 97: Factor concept General options ................................................................................................. 93
Figure 98: Example of Voice of the Customer model ..................................................................................... 93
Figure 99: Organizational unit concept ......................................................................................................... 94
Figure 100: Organizational unit General options .......................................................................................... 94
Figure 101: Performer concept ....................................................................................................................... 95
Figure 102: Performer general options ......................................................................................................... 95
Figure 103: Role concept ............................................................................................................................... 95
Figure 104: Team concept ............................................................................................................................. 96
Figure 105: Example of Organizational Unit................................................................................................ 96
Figure 106: Actor concept ............................................................................................................................ 97
Figure 107: Market segment concept ............................................................................................................ 97
Figure 108: Representations of some Value interface orientations ............................................................... 97
Figure 109: Value interface options ................................................................................................................ 98
Figure 110: Start stimulus concept ................................................................................................................. 98
Figure 111: Start stimulus options ................................................................................................................ 98
Figure 112: AND/OR concepts ...................................................................................................................... 99
Figure 113: AND/OR options ....................................................................................................................... 99
Figure 114: Example of E3 Business model ................................................................................................. 100
Figure 115: Business entity group concept ..................................................................................................... 100
Figure 116: Business entity concept ............................................................................................................. 101
Figure 117: Business entity capabilities options ............................................................................................ 101
Figure 118: Business role options ................................................................................................................ 101
Figure 119: Example of Scope configuration model ...................................................................................... 102
Figure 120: Business entity concept ............................................................................................................ 102
Figure 121: Process category concept ............................................................................................................ 103
Figure 122: Process category general options................................................................. 103
Figure 123: Process category metrics options.............................................................. 104
Figure 124: Process category alternative representations............................................. 104
Figure 125: Process category representation options..................................................... 104
Figure 126: Process concept.......................................................................................... 105
Figure 127: Process general options............................................................................. 105
Figure 128: Link to referenced Process model............................................................... 105
Figure 129: Process metrics options............................................................................. 106
Figure 130: "Described By" reference ......................................................................... 106
Figure 131: Process alternative representations............................................................ 107
Figure 132: Process left arrow example options........................................................... 107
Figure 133: Splitting hub / Joining hub concepts ........................................................... 107
Figure 134: Decision concept ...................................................................................... 108
Figure 135: Start concept ............................................................................................ 108
Figure 136: Critical and Non-Critical End concepts ..................................................... 108
Figure 137: End general options................................................................................... 108
Figure 138: Push boundary concepts .......................................................................... 109
Figure 139: Push boundary options to change side ..................................................... 109
Figure 140: Example of Thread model.......................................................................... 110
1 OVERVIEW

1.1 Introduction

Collaborative networked organizations represent an important paradigm to help manufacturing companies to cope with the challenges of market turbulence. However, although manufacturing companies are increasingly aiming to implement mobile collaboration models, there are still some barriers that prevent manufacturers from effectively sharing big data leveraged in particular by the use of Internet of Things technologies.

These barriers are basically related to the lack of security controls and lightweight data management enablers that would guarantee the protection of sensitive information published on the Linked Data cloud and leverage customers from the burden of linked data sharing.

To overcome the aforementioned limitations industry is working on the definition of the technology enablers to provide an inter-organizational mobile collaboration space through a Linked Data cloud turning today’s organization-centric manufacturing approach into a product-centric one. Manufacturers will benefit from a flexible, efficient platform that helps them to operate as one virtual factory and hence gain competitive advantages in their markets.

However, the realization of such product-centric collaboration space poses significant challenges in terms of providing the right security controls to ensure effective mobile access control management to distributed linked information data sources that create the specific product-centric information spaces shared by the enterprises engaged in collaboration. Traditional access control models are not particularly well suited to meet the major security challenges that product-centric information spaces raise. They do not facilitate a fine-grained access control model for the Web of Data, provide a limited support to mobile operations and do not deliver a common access control framework to the heterogeneous data sources including transient data and traditional binary resources and documents.

As ComVantage specifically addresses user centric collaboration tools, WP3 focuses on a secure access control to information requests. The access control model that will be proposed will enable mobile agile inter-organization collaboration in the Semantic Web addressing the limitations of more traditional solutions. This access control model will be aligned with the principles of dynamic and de-centralized collaboration.

Security implementation is not obvious in the ComVantage project. Some of the aspects that have to be considered are:

- **Multi-domain.** Each organization defines its own security policies. There is no centralized security component.
- **RBAC access control model** in multi-domain environment requires extensions to the well accepted identity federation mechanisms.
- **Linked Data.** The SPARQL standard doesn’t include any mechanism for security. More often than not, the queries use RESTful instead of SOAP.
- Applications developed on **mobile devices.** The computational power of those is low. In addition programming environments are specific.
  - Applications on Android may be developed in Java with the usual APIs, but they don’t use a standard JVM and the byte code is not compatible. Problems may appear with the libraries. For instance OpenSAML seems not working on Android.
  - iOS native language is Objective-C.
• Heterogeneous platforms on the server side. Servers may be implemented on JEE application server or Apache, or even exotic systems such as Hiawatha. SPARQL endpoints may run on dedicated standby applications or industrial systems.

• Modeling. Applications are designed through modeling. This approach is interesting for security perspectives. In the modeling phase, the actors, tasks and involved resources may be identified. So we can at this stage determine the permissions that should be granted to each actor (role). The modeling process should specify parts of the security policies. This guarantees coherence of the policies and the job of the users.

In this context, the choices about the architecture and the communication protocols are very important. Choosing the good technologies is one thing, but it is more essential the way we will use them and where we locate the different components. For example, some functions may be implemented in proxies (or reverse-proxies) and brokers, but it may be advantageous to capitalize on the existence of the Domain Access Server and implement as much as possible security functions there, thus minimizing non-Java developments.

1.2 Scope of this Document

The aim of this deliverable, as outlined in the ComVantage Description of Work (DoW), is to conceptualise the results to provide a comprehensive and coherent set of guidelines for secure collaboration model within the ComVantage project. In this document the whole security model architecture is defined as well as its basic operation.

This document is organised as follows:

• First of all, section 2 provides a description of the security prototype that has been developed. The general operation and architecture of the security framework is presented. The authentication process is also defined in this section, as well as both authorisation processes that will be executed depending on the type of information requested. This section also presents some specifications for the business model tool and for the secure information model.

• Afterwards, section 3 presents the guidelines for using the modelling tool support resulted from task 3.4, including modelling with the OMI prototype, which has been provided through the Open Model Initiative community space. Documentation on how to join the community space and how to get access to the remotely accessible tool has been provided as a companion document for D3.4.1. The current document only covers the usage guidelines for the modelling tool support.

• Finally, section 4 concludes the document.

1.3 Related documents

The following figure presents this deliverable position in the global ComVantage context with respect to related deliverables.
The relations expressed in the figure are the following:

- Use case refined scenarios provide the security needs that must be covered.
- Global ComVantage architecture and functional and technical requirements provide the context where the secure information model must fit.
- The security model must be aligned with the user interface presentation and modelling.
- The security model must be adapted with the other technological work packages, in terms of Linked Data format specification as well as usability and trust models.
- The secure information model, as it is presented in the current deliverable, is generic and it will be further specialized in the adaption deliverables from WP6, 7 and 8.
- The secure information model specified in this deliverable will be the base to develop the prototypes of the security mechanisms.
2 DESCRIPTION OF THE SECURITY PROTOTYPE

2.1 Multi-domain mobile environment’s security aspects

As already stated in (Lázaro, et al., 2012), in order to define a security model based on controlling access to Linked Data, some assumptions must be considered previously. These assumptions are the following:

1. **Establishment of a Circle of Trust between collaborative enterprises**
   
   A Circle of Trust (CoT) must be set up for all collaborative enterprises. A list of global roles must be defined for this CoT. This is what we will name as **common ComVantage roles**. The assertions exchanged between domains should include only these Global Roles.

   User Assignments indicate which roles are assigned to each user. Permission Assignments indicate which permissions are assigned to which role. In inter-domain environments User Assignments (UA) and Permissions Assignments (PA) are done independently in separate domains.

2. **Qualification of the origin domain by means of an Identity Provider**

   Even if there is a contract between two domains, the target domain will not trust the origin domain blindly. We should be suspicious about user assignments made in other domains. The common ComVantage roles should be filtered in function of the origin domain. **Global roles must be qualified by the origin domain and the target domain has to check this qualifier.** This process requires that the target domain trusts the Identity Provider (IdP) of the origin domain.

3. **Publishing of information as a responsibility of each data provider**

   At this stage and in order to focus on securing access to Linked Data information, it is assumed that the **publication process of information to Linked Data is responsibility of each data provider**. That is, it is assumed that each partner will use its own tools to control the consistency of data being published in Linked data but will use the tools provided by WP3 to define the access control policies affecting this data. For this first mock-up, thus, we will focus on providing **access control management functionalities to information already published in Linked Data**. The securing of the publication of information to Linked Data will be discussed later in the project as part of future demonstrators, but at this stage, it is out of scope of this document.

4. **Each consortium application has its own security rules and access control mechanisms**

   In fact, each consortium application authorized to perform SPARQL queries should have its own security rules and access control mechanisms to control which Linked Data is accessed/modified by its authorised users. For example, if the same authorised application is used by a service technician and by a salesman, it is responsibility of each domain not showing prices to the technician because he is not authorized to see them. But also, the application should allow the salesman access to prices. Those security rules and access control mechanisms are out of the scope of WP3.

5. **Focus on controlling queries to Linked Data**

   At this stage of the project, we will focus on **controlling queries to Linked Data** in order to consult information published. We will not control initially permissions of writing, modifying or deleting Linked Data information.

6. **The user applications will store the addresses of the Domain Access Servers with whom they collaborate**

   It is assumed that, at this stage of the project at least, the **addresses of the several Domain Access Servers** are stored in the user application. In fact, they should have been preconfigured at design time.

7. **A user application will directly call the relevant Domain Access Servers, independently of its domain**
Thus, a user application will directly call the relevant Domain Access Servers, independently of its domain. That is, a user application from domain A, who needs information from both Domain A and B, is able to call a Domain Access Server from Domain A and another Domain Access Server from Domain B.

8. There will be a limited and fixed amount of domains within each application area

There will be a limited and fixed amount of domains within each application area, that is, a fixed amount and addresses of Domain Access Servers. In fact, all dynamic aspects like partnering and contracting with new companies and updating of a network configuration is out of scope for ComVantage (at initial stage). In fact, adding a new domain to a workflow will be a manual process which will be realized by manually updating the business process model and redeploying the mobile app.

2.2 General operation and architecture of the security framework

2.2.1 General operation

In a mobile collaborative environment, it normally occurs that a user from one company wants to access some information from another company. As an example, Figure 2 shows an scenario in which there are two different companies that are collaborating together and thus two different information domains: one factory and the maintenance company in charge of repairing the machines when there is a failure or when they break down.

The factory has:

- Some machines and each machine has:
  - some sensors that produce data that is continuously changing, such as the temperature of the machine or the air pressure...
  - some other static information related to the machine, such as its name, or the manufacturer of that machine...

- Apart from that, the factory also stores traditional information in PDF format, excel sheets, etc, containing for example stock or financial information...

The maintenance company also has some information such as the list of the last repairments, stock material, etc.

Those companies want to collaborate together acting as one virtual factory and they decide to take advantage of the benefits that the Semantic Web can bring to them, so they publish the information they want to share with the other company in the Linked Data cloud.

If one of the machines breaks down, the maintenance company has to fix the failure. In order to do that, the technician decides to look through Linked Data on some parameters of the broken machine to see if it is possible to fix the error remotely.
However, it is essential to guarantee that the information shared between both companies remains accessible just to authorized users and that no malicious attacker will be able to see that information. Indeed, collaboration between different organizations can only be successful if partners can trust each other and really believe that the information they share remain secure while only authorized users can access their data. Hence, facilitating trust and ensuring that Linked Data information remains secure and only accessible to authorized members is a crucial issue.

Thus, it is important to establish a Role Based Access Control, which will restrict access to authorized users based on the roles they belong to. It is essential that access control should be supported at a fine granularity (i.e., RDF level), and not only at a coarse-grained level (i.e., repository level). Figure 3 shows how the access control approach will be applied over a collaboration scenario such as the one exposed above.
Figure 3: Access-control solution applied to the example scenario

Or in a more schematic way as shown in Figure 4:

As depicted on the above figure, the client doesn’t directly connect to the final server, but all communications are controlled by the Domain Access Server (DAS) module. A complete description of the DAS can be found in D2.2.1. For our purpose, it is worth to note that the DAS and the final server are in the same domain. Thus, the access control services can be implemented within the DAS. Please, note that each domain in a collaborative scenario will have one DAS, although it is not identified in Figure 3 and Figure 4.
As already stated in section 2.1, precisely in the points 6 and 7 the user applications will store the addresses of the Domain Access Servers with whom they collaborate and will directly call the relevant Domain Access Servers.

Following with the scenario that has been highlighted, first of all, so that the factory can trust on an operator coming from the maintenance company, a cross-domain authentication will take place, which will be explained in Section 3.

After that, an authorization process will take place in order to guarantee that the operator accesses just the information he is authorized to see depending on his role. This authorization process will be in the Factory Access Server (DAS), which will be the one receiving any queries asking for access to information from the Factory. Depending on the type of information the user wants to access, two different authorization processes can be performed. It is important to take into account that within a factory, we could find different types of information. On the one hand, above all when dealing with manufacturing environments, machines could provide information gathered by its sensors, such us its temperature, its pressure, etc. which are changing values continuously over the time. The factory could also host traditional resources such as binary resources, excel sheets or text documents. These data are not published in an SPARQL endpoint; instead, its location (its URL) can be accessed but not the data itself. On the other hand, the static information of the machine that will not change continuously, such as its name, its last check date, etc or even other information of the factory such as stock management or financial information will be published in RDF.

So, if the user wants to access some excel or word file or some changing information of the machine, such as its temperature, a traditional XACML access control will be executed. However, if the user wants to access some historic information of the machine, such as the day in which the last check occurred, the query will go through the innovative query rewriting process.

Thus, we have two authorization processes in parallel that will be executed or not depending on the type of information the user requests:

1. SPARQL rewriting
2. XACML access control

### 2.2.2 Architecture of the security framework

Figure 5 shows the designed access control architecture integrated within an example of two different domains as shown in the example (Domain A and B). Users will use the mobile applications (located in the Mobile Application Layer) in order to execute requests for information to all the involved domains. Each domain has a Domain Access Server (DAS), which is the single point of access to all data sources of one domain. Thus, all the requests for information of one domain will arrive at the DAS of that domain. The DAS hosts the access control systems.

As the DAS always participates in the links from the client, we will opt for communication profiles where the big part of the processing is done in the DAS, reducing developments on mobile devices.

The strategy focused on the DAS has the following advantages:

- Maximize developments for security in Java;
- Facilitate integration of heterogeneous servers;
- Minimize developments on mobile devices.

In the bottom of the picture, the Domain Source Layer is located, where the different SPARQL endpoints and servers are hosted, both the SPARQL endpoints where the semantic data are located and the servers where the live/transient data and the resources in traditional format are hosted.
The boxes painted in green in Figure 5 are the security systems that will be integrated into the whole collaborative architecture:

```
Figure 5: Combined AC approach to enable secure mobile collaboration through the Web of Data
```

### 2.3 Authentication process

So as to being able to perform an access request to the information in the target domain, first of all, the target domain will need to trust the requester. The target domain will not trust the origin domain blindly. Administrators should be suspicious about user assignments made in other domains. Global roles must be qualified by the origin domain and the security policies in the target domain have to check this qualifier. This process requires that the target domain trusts the Identity Provider (IdP) of the origin domain. In fact, as the different domains are just responsible of managing their own users, a user from domain A will have to authenticate against its own domain and deliver a security assertion to domain B so that it can now trust the requester.

The line of research has been based on SAML (Security Assertion MarkUp Language) (Anderson, 2005). This OASIS standard defines both an XML schema for security assertions and protocols to exchange these assertions. SAML allows business entities to make assertions regarding the identity, attributes and entitlements of a subject to other entities, such as a partner company or another enterprise application. An
assertion contains one or more statements. An attribute statement contains attributes of a user such as roles of this user, in this case, the global role of the user.

When talking about a way of exchanging authentication and authorization data between secure entities in different security domains, SAML (Security Assertion MarkUp Language) (Anderson, 2005) standard arises as one of most suitable technologies to face the "ComVantage Secure Authentication" approach. SAML is an XML-based open standard data format defined by OASIS Security Services Technical Committee. It is a not a new standard, it has been around since 2002. The single most important problem that SAML addresses is the web browser single sign-on (SSO) problem. SSO solutions are abundant at the intranet level (using cookies for example) but extending these solutions beyond the intranet has been problematic and has led to the proliferation of non-interoperable proprietary technologies.

Some of the main features of SAML are as follows:

- Standard-based: as aforementioned, SAML is based on a standard, which ensures interoperability across identity providers.
- Usability: One-click access from portals or intranets, deep linking, password elimination and automatically renewing sessions make easier for the user to use.
- Security: Based on strong digital signatures for authentication and integrity, SAML, is a secure single sign-on protocol that the largest and most security conscious enterprises in the world rely on.
- Phishing Prevention: in absence of password for an application, it won't be tricked into entering it on a fake login page.
- IT friendly: SAML simplifies life for IT because it centralizes authentication, provides greater visibility and makes directory integration easier.

An important characteristic of the current mobile collaboration networks the context of ComVantage is the use of mobile devices. Most applications will run directly on mobile devices, the SPARQL queries are launched from the devices and are directly sent to the SPARQL endpoints. In such a situation it is not possible to rely on a web browser to support identity federation. Unlike traditional SAML implementations based on a client trying to access a SP from a browser, the planned approach must be fully compliant with OASIS SAML ECP (Enhanced Client or Proxy) profile, responsible of defining a SAML profile suitable for non-browser clients.

2.3.1 Main architectural entities involved in the authentication process
The main entities involved in the scenario proposed are as follows:

- **Identity Provider (IdP):** the element responsible of creating, maintaining, and managing identity information for principals (users, services, or systems). It hosts a LDAP repository where users’ credentials/attributes of its own domain are stored, as addressed in Figure 6. It provides principal authentication to other service providers (applications) within a collaborative or distributed network. It is a trusted third party that can be relied upon by users and servers when users and servers are establishing a dialog that must be authenticated. It also works as a STS (Security Token Service), responsible for issuing security tokens.
D3.5.1 Guidelines for the secure collaboration model

Guidelines for the secure collaboration model

© ComVantage Consortium – 2013

19

Figure 6: SAML IdP

- **Service Provider (SP):** the element responsible for protecting the resource/application. Makes authorization decisions. Acts as the Policy Enforcement Point (PEP).

- **Mobile/Application Client:** the element that requires access to the information of the resource stored in SP, on behalf of the User-Client.

As it has been already said, the multi-domain security framework needed in the context of the Web of Data must be built on SAML identity federation. Each user must authenticate towards the IdP in its own domain which must be able to deliver Assertions containing global roles. This standard will allow authentication messages (SAML over SOAP) to be exchanged between parties such as the Identity Provider (IdP) and the Service Provider (SP), both located in different domains.

2.3.2 Description of the authentication process in a non browser environment

Following with the previous example, first of all, so as to being able to perform an access request to the information of the machines in the factory, the factory will need to trust the operator from the maintenance company, as shown in Figure 7.

The operator from the maintenance company will have to authenticate against its own domain, against its own Identity Provider, which stores an LDAP repository where users’ credentials/attributes are stored. The Identity Provider of the Maintenance Company will provide a security certificate (token) to the operator. The operator will be able to deliver that security certificate to the SP of the factory domain, which will now be able to trust the requester.
Figure 7: Authentication example between two companies

Figure 8 shows the different steps that a user client has to follow in order to get a security token to be used so as to get access to a Service Provider in the target domain.

1. The user application in domain A (user A from now on), by means of his mobile application, launches a HTTP request to the SP in domain B (SP B from now on), with the aim of establishing a connection.

2. As SP B is not able to check the identity of a user from another domain if no credentials to validate its identity are provided, SP B requests a security token from user A.

3. User A then authenticates against its own domain IdP (IdP A from now on) in order to obtain an identity assertion which will be seen as a trust guarantee for SP B. Before delivering the identity assertion to User A, IdP A may request some information from the principal in order to authenticate the user. Note that SAML does not specify the implementation of the identity provider service, i.e. it does not define the way to authenticate users, so they can use a userID/password or even fingerprint or biometrical authentication methods.

4. With the authentication information provided, the IdP A is capable of identifying User A, as each IdP has a repository with all the users belonging to its own domain.

5. The service consumer obtains a SAML attribute assertion (a security token) from its IdP. The IdP must be able to deliver attribute assertions containing global roles.

6. The service consumer includes this assertion in the SOAP envelope and sends it to the Service Provider.

7. SP B removes the PAOS envelope and verifies the signature of the assertion (During setup time IdP and SP SW components exchange digital keys, including them in the configuration steps, aiming at doing these signature checks during runtime). So the SP validates it and extracts the roles of the user A from the attribute statement. Based on the attribute assertion, SP B will take a ‘deny’ or ‘permit’ decision.

8. The SP communicates this decision to the requester.

If the decision has been ‘permit’, a trusted connection between User A and SP B is established from this moment on, and each time a request is sent from User A to SP B, the credentials obtained must be included and provided to SP B.
This approach allows any user (coming from any domain) to be authenticated via a non browser environment, in its own/origin domain and then using trustworthily the token in a potentially different server-side domain, what is in accordance to multi-domain collaborative scenario expected in the context of ComVantage.

The authentication prototype developed within ComVantage project can be found in D3.3.1a Prototype of security mechanisms - Secure Authentication Software Deployment.

### 2.4 Authorisation process

After this authentication process has been completed, the query will be received in the Query Interface and the authorization process starts. The Query Interface receives HTTPS requests with embedded SPARQL queries or requests for single dereferenceable URIs. Depending on the nature of the requested data, different authorization processes will be executed. If the requested data are published in the Linked Data cloud as RDF, the SPARQL query rewriting process will be performed and will provide a SPARQL rewritten query as a result, which will be launched to the corresponding SPARQL endpoint. (Section 2.4.1)

However, in those situations where live data are demanded, the access will be performed to an URL where the changing RDF information is stored instead of to the data itself, that is, the access will be to the location of the data, not to the data itself. If the mobile application from the requester domain does not know the URL where the live data are located, it should launch a SPARQL query to obtain this URL. This SPARQL query
will be rewritten and as a result the URL will be obtained. Once the URL where the live data are stored is known, a traditional XACML authorization process will happen. (Section 2.4.2)

The XACML prototype developed within ComVantage project can be found in D3.3.1b Prototype of security mechanisms – XACML security mechanisms. The SPARQL rewriter prototype developed can be found in D3.3.1c Prototype of security mechanisms - Query rewriter Software Deployment.

2.4.1 Authorisation process based on SPARQL query rewriting

Once the user has authenticated successfully and obtained a security token, an authorization process will take place when the user wants to access some RDF resource. This access control is based on an out-of-band process and a runtime process.

First of all, out of Band, we will take the original data published in Linked Data and will apply the security policies to them, which will restrict the access of concrete roles to concrete sets of information. As a result, what we call Data Views will be created and will be published instead of the original RDF data.

Then, during runtime, when a query is received from a particular user role, for example, the operator, a rewriting process will take place, which will include authorization checks so that the rewritten query that includes requester roles/groups will just allow authorized access to data contained into the views associated with those roles/groups. Note that the “Roles” and “Groups” usually describe different concepts. A role is defined by a business context and a user group is defined in a user administration context. However, within this document, both concepts are used indistinctly and refer to the same idea.

![Figure 9: Authorisation process based on an out of band and a runtime process](image)

2.4.1.1 Creation of data views (out of band process)

Linked Data is defined as a set of triples formed by subject-predicate-object, where a triple will be the smallest unit that could be protected.
Figure 10 shows an example of this concept of Linked Data, where the machine’s name is Grinding, its Maintenance Company is RST or its owner company is K&A.

Authorization policies describe a set of rules/formulae concerning the RDF data and will organize the original Linked Data into a set of data Views which are assigned to a concrete group of users, which are the ones who are going to be able to see or use that view.

These policies will be written with N3 notation (Notation3, 2013) which is a readable RDF syntax. Some considerations to write a policy will be that being allowed to use some View is not the same as being allowed to see it. In fact canUse property will be more restrictive than canSee. If canSee is allowed to a View, then canUse is permitted, this concept will have a concrete authorization policy rule in the implementation, as described below using Notation3,

\[
@forAll :x, :y. \{ :x ac:canSee :y \} \Rightarrow \{ :x ac:canUse :y \}
\]

...which will means that for all the set of data if x can see y, then x will be able to use y.

The next examples describe authorisation policies that can be written, using Notation3:

- **Policy 1:** Company X operators can see all configuration parameters of machines that they maintain.
- **Policy 2:** Engineers related with the project X can see all the properties of project X, except those properties related with cost.
- **Policy 3:** Managers related with the project X can see all the properties of project X, except sensor parameters values.
- **Policy 4:** T-Shirt manufacturers can see all the stock quantities of T-Shirts that they have sell to Company X.
- **Policy 5:** Company X can see the amount of time spend by the maintenance operators on machines of their own, but it could not see which operators have been working on them.
As seen in Figure 11, finally Views will be created inferring new facts and importing them into RDF data store, as a result of this process a new set of RDF data with the facts inferred and organized in Views will be created and published replacing the original RDF data.

The idea is to assign these views to a concrete group of users which are the ones who are going to be able to see or use that view. This is going to be specified in the domain authorization policies.

So, if it is desired that a concrete role can access (see or use) certain information, a view must be created for that role. If that view already exists, that role/group should be added as member of the groups that are able to see or use that view.

Take this example in Figure 12 where there is, as the original Linked Data from the Factory company, one machine with different attributes: name, manufacturer, owner, maintenance company, last check day and the status of that last check. It also has three configuration parameters, which are the Maintenance Responsible, the Price and the Maintenance Frequency.
This original Linked Data (what is known as domain instances) represented as Notation N3 would be as follows:

```n3
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix voc: <http://www.comvantage.eu/vocabulary#>.
@prefix ins: <http://www.comvantage.eu/instances#>.

ins:machine0 a voc:Machine;
  foaf:name "Grinding";
  voc:manufacturer "COMAU";
  voc:owner "K&A";
  voc:maintenance "RST";
  voc:lastCheck "2012-09-17"^^xsd:date;
  voc:checkStatus "passed".

ins:machine0_config_0 a voc:Parameter;
  foaf:name "MaintenanceResponsible";
  voc:value "Bryan Olsen";
ins:machine0 voc:configParam ins:machine0_config_0.

ins:machine0_config_1 a voc:Parameter;
  foaf:name "Price";
  voc:value "12000"^^xsd:integer;
ins:machine0 voc:configParam ins:machine0_config_1.

ins:machine0_config_2 a voc:Parameter;
  foaf:name "MaintenanceFrequency";
  voc:value "Yearly";
ins:machine0 voc:configParam ins:machine0_config_2.
```

As an example, it is desired to make the machine’s features available to be seen or used by a public view and that, apart from that, users belonging to the role MANAGERS can also have access to the three configuration parameters. Nevertheless, users belonging to the role OPERATORS can just see the Maintenance Responsible and Maintenance Frequency, but not the price of the machine.

In order to do that, some security policies must be built that will restrict the access of some groups of users to some particular RDF data, thus inferring the data views. The policy that must be created is the following:

```n3
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix ac: <http://www.comvantage.eu/ac-schema#>.
@prefix gr: <http://www.comvantage.eu/ac-schema/groups#>.
@prefix log: <http://www.w3.org/2000/10/swap/log#>.
@prefix string: <http://www.w3.org/2000/10/swap/string#>.
@prefix voc: <http://www.comvantage.eu/vocabulary#>.
@prefix ins: <http://www.comvantage.eu/instances#>.

{ } log:implies { ins:everyone a foaf:Group }.
{ } log:implies { ins:public a ac:View }.
{ } log:implies { ins:everyone ac:canSee ins:public; ac:canUse ins:public }.

@forAll ins:m, ins:u, ins:v, ins:w, ins:x, ins:y, ins:z.
  { ins:m a voc:Machine;
    foaf:name ins:u;
```
voc:manufacturer ins:v;
voc:owner ins:w;
voc:maintenance ins:x;
voc:lastCheck ins:y;
voc:checkStatus ins:z\}

log:implies
\{ ins:public ac:hasTuple
[ a ac:Tuple; ac:subject ins:m; ac:predicate foaf:name; ac:object ins:u ],
[ a ac:Tuple; ac:subject ins:m; ac:predicate voc:manufacturer; ac:object ins:v ],
[ a ac:Tuple; ac:subject ins:m; ac:predicate voc:owner; ac:object ins:w ],
[ a ac:Tuple; ac:subject ins:m; ac:predicate voc:maintenance; ac:object ins:x ],
[ a ac:Tuple; ac:subject ins:m; ac:predicate voc:lastCheck; ac:object ins:y ],
[ a ac:Tuple; ac:subject ins:m; ac:predicate voc:checkStatus; ac:object ins:z ] \}.

\{ \}

log:implies
\{ ins:cv_managers_view a ac:View.
gr:cv_managers a foaf:Group;
acc:canSee ins:cv_managers_view;
acc:canUse ins:cv_managers_view;
acc:canSee ins:public;
acc:canUse ins:public}. \}.

@forAll ins:x, ins:y, ins:z, ins:n, ins:v, inst.
\{ ins:x voc:configParam ins:y.
insy foaf:name ins:n.
insy voc:value ins:v. \}

log:implies
\{ ins:cv_managers_view ac:hasTuple [ a ac:Tuple; ac:subject ins:x; ac:predicate voc:configParam; ac:object ins:y ].
ins:cv_managers_view ac:hasTuple [ a ac:Tuple; ac:subject ins:y; ac:predicate foaf:name; ac:object ins:n ].
ins:cv_managers_view ac:hasTuple [ a ac:Tuple; ac:subject ins:y; ac:predicate voc:value; ac:object ins:v ]. \}.

log:implies
\{ ins:cv_operators_view a ac:View.
gr:cv_operators a foaf:Group;
acc:canSee ins:cv_operators_view;
acc:canUse ins:cv_operators_view;
acc:canSee ins:public;
acc:canUse ins:public}. \}.

@forAll ins:x, ins:y, ins:z, ins:v, inst.
\{ ins:x voc:configParam ins:y.
insy foaf:name "MaintenaceResponsible".
insy voc:value ins:v. \}

log:implies
\{ ins:cv_operators_view ac:hasTuple [ a ac:Tuple; ac:subject ins:x; ac:predicate voc:configParam; ac:object ins:y ].
ins:cv_operators_view ac:hasTuple [ a ac:Tuple; ac:subject ins:y; ac:predicate foaf:name; ac:object "MaintenaceResponsible" ].
As it can be seen in the defined policy, it is said that MANAGERS are authorised to see and use the views public and cv_managers_view, while OPERATORS are authorised to see and use the views public and cv_operators_view. On the one hand, cv_managers_view is composed of all the configuration parameters of all the machines within the factory. On the other, cv_operators_view is composed of the configuration parameters MaintenanceResponsible and MaintenanceFrequency. This implies that both MANAGERS and OPERATORS will be able to see and use the machines features (name, manufacturer, owner, maintenance, last check and check status). MANAGERS will also be able to see and use all the configuration parameters of all the machines and OPERATORS will just be able to see and use the parameters MaintenanceResponsible and MaintenanceFrequency, but not the prices of the machines.

So, as a result of having applied these policies to the Original LD, the different data views are generated. Figure 13 shows the inferred public view:
Figure 14 shows the OPERATORS view inferred:
As it can be seen, the view is formed by a set of triples:

```
ins:cv_operators_view  a :View;
   :hasTuple [ |
      a :Tuple;
      :object ins:machine0_config_0;
      :predicate voc:configParam;
      :subject ins:machine0 ], |
      a :Tuple;
      :object "MaintenaceResponsible";
      :predicate <http://xmlns.com/foaf/0.1/name>;
      :subject ins:machine0_config_0 ], |
      a :Tuple;
      :object "Bryan Olsen";
      :predicate voc:value;
      :subject ins:machine0_config_0 ], |
      a :Tuple;
      :object ins:machine0_config_2;
```
The inferred view for the MANAGERS is not shown in this document but would be similar to this view but also including the corresponding triples for the Configuration Parameter of the Price of the Machine.

Of course, as I have already said, all this process of data views inferring, being based on the original LD and the domain policies, and publishing them in the corresponding endpoints, is done out of band.

What is really interesting about the data views is that by default, all the information is protected. The domain inferred views will be published in the corresponding SPARQL endpoint. Thus, these views will be the ones that the Domain Access Server, which is the only one that can have access to the SPARQL endpoint, will be able to access. The domain instances (that is, the original Linked Data) will not be published in the store. So, if we do not create a view for a particular user including that information, that information will remain protected and no user will be able to access it.

2.4.1.2 SPARQL Query rewriting (runtime process)

The rewriting process consists on adding control checks to the received SPARQL query so that the resulting query will work only with information that is related with the roles that the requester belongs to, as shown in Figure 15.

Those checks added by the rewriting algorithm defined are composed by triple patterns and constraints. During query resolution, the added triple patterns and constraints will limit the access to the information that has been collected in any view related with the groups of the requester.
The rewritten query will be launched to the endpoint where the views had been published out of band. Just the information related to the views that the operator is able to see or use will be returned to the user. Figure 16, following with the initial example posed in Figure 2, shows what happens when the maintenance operator wants to access some of the configuration parameters of the machine that has crashed. The operator, after having authenticated as explained before, sends a query to the Domain Access Server of the Factory domain. This query will be an SPARQL query which is the query language used to request information stored in RDF format. In this Access Control Server a rewriting process will be executed. This process consists on adding control checks to the received SPARQL query so that the resulting query will work only with information that is related with the roles that the requester belongs to.

The rewritten query will be launched to the endpoint where the views had been published out of band. Just the information related to the views that the operator is able to see or use will be returned to the user.

Figure 16: SPARQL rewriting process

Take the following example in Figure 17. The original SPARQL query is the following one:

```
SELECT ?name ?value
WHERE {
}
```

Figure 17: Original SPARQL query

By means of this query the user is trying to access the name and value of all the configuration parameters of machine 0.

This original query will be rewritten by adding triples and constraints which check whether a view accessible to that role and containing that data exists.

For instance, if the user’s role is [http://www.comvantage.eu/ac-schema/groups#cv_managers](http://www.comvantage.eu/ac-schema/groups#cv_managers) (taking into account that this role information will be the one obtained from the IdP in the authentication process), the original query will be rewritten as follows, resulting in the following rewritten SPARQL query shown in Figure 18:
Hence, the original query will be rewritten by adding triples which check whether a view accessible to that role and containing that data exists and allows the user to access to each set of ?name and ?value bindings. For each view ?v that applies to a group 'g', its internal nodes will be bound (subject, predicate, object).

So, if the user trying to access the information of the machine is a MANAGER, he will be able to see the three configuration parameters of the machine: Maintenance Responsible, Price and Maintenance Frequency.

If the user trying to access the information of the machine is an OPERATOR, he will be able to see just the configuration parameters Maintenance Responsible and Maintenance Frequency, but not the price of the machine, because as we saw before, the OPERATOR's view just consists on those two parameters.

If the user trying to access the information of the machine is another role that does not have a view associated to it, for example, UNKNOWN, he will not be able to see anything.

It is important to note that the rewriting process is independent from the policies defined, i.e. the same conditions are always added with independence from the policies. In fact, the defined rewriting algorithm does not need to see the set of configured policies.

2.4.1.3 Results of query rewriting algorithm implementation

In order to verify the efficiency and performance of the query rewriting algorithm and the application of the policies in terms of processing load and time, the following tests have been performed:
Taking into account the Original Linked Data published in the endpoint there were 42,199 statements and, as a result of inferring the data views, 212,532 statements were obtained, which is five times larger.

Taking into account the query execution times, it has been proved that it takes 109 ms to execute an original query, whereas it would take 697 ms to execute a rewritten query.

The tests have been executed by means of Allegro Graph RDF Store Web 3.0 Database (AllegroGraph, s.f.), which is a database and application framework for building Semantic Web applications.

Of course, it can be appreciated that the query rewriting process involves much more processing time (six times more than without rewriting) but this is not an important issue in the context of the mobile collaboration in manufacturing environments, where these delays do not interfere in the ordinary operation. The prototype evaluation shows that, although the overall performance needs to be optimized, the delay introduced by our fine-grained access control is acceptable given that data protection comes with a cost.

### 2.4.2 Authorisation process based on XACML

Fulfilling the needs of distributed architectures, the XACML architecture (Godik, 2008) logically separates the access components responsible for policy definition, policy enforcement and policy evaluation. Specifically, XACML architecture is basically composed of the four coloured elements in Figure 19.

- **Policy Enforcement Point (PEP):** PEP is the interface with the outside world, where the access request is received. It intercepts the application requests and submits decision requests to the PDP.

- **Policy Decision Point (PDP):** PDP is the evaluation engine that evaluates decision requests against the policies coming from the PAP. PDP consults and evaluates the policies that apply to the requested resource and takes a decision (permit, deny, indeterminate or not applicable). In order to evaluate the policies, PDP has to consult PIP to obtain the attributes that it does not know (subject, resource or environment).

- **Policy Information Point (PIP):** PIP is in charge of consulting and sending back the attributes demanded by the PDP so that it can take an appropriate decision. For instance it can look up an LDAP directory to get the roles of a user.

- **Policy Administration Point (PAP):** PAP is the repository which creates and stores the policies for the PDP. Usually the policies are administrated through a data model such as RBAC. Then it is translated into XACML.
The above figure shows the information flow between the actors in the XACML model. PAP stores the policies and sends them to PDP. PEP receives an access request to a particular resource to perform some action. The attributes in the request may be in an SAML format. PEP sends the request to the Context Handler, which maps the request and attributes to the XACML Request context and sends the request to the PDP. In order to take a decision about the request, PDP needs some attributes and sends attribute queries to the Context Handler, which picks them up by the help of the PIP from the resources, subjects, and the environment and returns them to PDP by means of the Context Handler. With all this information available, PDP is able to take a decision so it sends the XACML Response to the Context Handler and the Context Handler translates the Response context to the native response format of the application environment and sends it to PEP. PEP fulfils the obligations if they exist and applies the authorization decision that PDP concludes.

The PEP may be implemented in the application, as a Web filter, as a Web Service handler, in a framework. Or it may be a proxy that intercepts all the communications with an application.

The PEP builds a request context containing attributes about the Subject, Resource and Action. This context is submitted to the PDP for evaluation. If the PDP is set up in an authorization server as a Web Service, the request context will be submitted using SAML protocol in a SOAP envelope. But the PDP can be also set up as a library, in this case the request context is passed directly in the memory of the process, avoiding overhead and facilitating load balancing.

XACML defines the content of some of the messages needed to implement the model, but does not specify the protocols or the transport mechanisms, nor does it define how to implement PEP, PAP, PDP, PIP or the context handler. Therefore, XACML must be combined with other standards to interchange information. SAML 2.0 covers these needs by providing schemes to carry out the process of request, response and validation.

XACML defines the XML format that should have the control access policies that are stored in PAP and are used by PDP. The most important elements of an XACML model are explained briefly here:
### Policy Set and Policy
A PolicySet is a container for a number of Policies or other PolicySets. A policy represents a single access control policy, expressed by a set of Rules. It applies for a defined Target. Thus, policy sets and policies are composed of:

- **Obligation**: actions that must be carried out after a policy is executed. PDP sends them back to the PEP so that it executes them.
- **Target**: it allows determining when the policy must be applied: there are conditions about the consumer, the resource, the action and the environment.
- **Rule**: the rules of the policies are made of:
  - **Target**: it is the resource or object the rule is trying to protect
  - **Condition**: it is a refined condition to know if the rule must be executed or not
  - **Effect**: it is the action that must be applied: Permit or Deny.

The basic Policy / Rule Combining Algorithm defined by XACML are:

- **First-Applicable**: it indicates to PDP that it should stop the processing of the rest of the policies/rules if it finds one that applies to the access request that is being evaluated.
- **Permit/Deny-Overrides**: it indicates to PDP that, in case two possible decisions may exist to the same request, Permit or Deny choice prevails.
- **Only-One-Applicable**: it indicates to PDP that for one concrete access request, only one policy/rule is applicable.

There are different open source implementations of XACML and, apart from that, a great variety of organizations base some of their products and even their internal security on XACML.

### 2.4.2.1 XACML Context

XACML pretends to be appropriate for many different application environments. The core language is isolated from the application environment by the XACML context, as shown in the figure below, in which the scope of the XACML specification is indicated by the shaded area in grey. Therefore, applications can use other representations like SAML, which is the most suitable one for the attributes. Then the PEP applications convert these attribute representations to the XACML context attributes. The XACML context is defined in **XML schema**, describing a canonical representation for the inputs and outputs of the PDP.

**A Request element** is made of four components as **Subject**, **Resource**, **Action**, and **Environment**. One request element has only one collection of resource and action attributes, and at most one collection of environment attributes. But it may have collections of subject attributes, such as name, e-mail, role and so on. Resource attribute details the resource for which access is requested and action attribute specifies the requested action to be performed on resource such as read, write or execute.

**A Response element** represents the authorization decision information made by PDP. It contains one or more **Result** attributes. Each result includes a Decision such as Permit, Deny, NotApplicable, or Indeterminate. The **Status** information indicates the errors occurred and their descriptions while evaluating the request. Optionally one or more **Obligations** specify tasks in the PolicySet and Policy elements in the policy description which should be performed before granting or denying access.

**A Rule element** defines the target elements to which the rule is applied and details conditions to apply the rule. It is composed of three elements: **target**, **effect** and **condition**. A target element specifies the resources, subjects, actions and the environment to which the rule is applied. A condition element shows the conditions to apply the rule, while the effect element is the consequence of the rule (permit or deny).
A **Policy** is the set of rules which are combined with some algorithms. These algorithms are called Rule-combining algorithms. For instance, as explained above, "Permit Override" algorithm allows the policy to evaluate to "Permit" if any rule in the policy evaluates to "Permit". A policy also contains target elements which show the subjects, resources, actions, and environment that policy is applied.

A **PolicySet** consists of Policies and PolicySets combined with policy-combined algorithm. It has also target like a Policy.

### 2.4.2.2 XACML Policies

The eXtensible Access Control Markup Language (XACML) defines XML schemas to write access policies. XACML policies are rule-based. We can think of XACML policies as a list of rules. Each rule has the following format:

**IF condition THEN permit / deny**

‘condition’ may be a complex boolean expression. This allows defining transaction amount threshold, test on time or date, condition on the execution environment etc.

The **“effect” of a rule is ‘permit’ or ‘deny’**. To resolve conflicts between rules and/or policies, we specify the method used for combining these rules and/or policies by using a specific algorithm.

The list of roles of a user is just an attribute of the Subject. In order to evaluate policies referencing roles, the evaluation engine needs to get the values of the corresponding attribute. There are two possibilities:

- Either the engine (more specifically the PIP) looks up in a directory to get the roles of the user;
- Or the roles are pushed in the request context.

### 2.4.2.3 Decision Requests

Beyond of the policy encoding, XACML defines the way to format requests and responses for the authorization service. XACML doesn’t specify a protocol, but relies on the SAML protocol (samlp).
Authorization decisions are encapsulated in SAML assertions. So they benefit of the security features of
SAML, such as signature, and can be forwarded to other service providers.

Hereafter, an example of XACML request and response using the Web Service profile of SAML is shown.
(The XACML statements without the SOAP and SAMLP envelopes are painted in red).

**XACML request:** a user wants to make a "GET" on the URL "http://localhost:8080/demo/"

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soap11:Envelope xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/">
  <soap11:Body>
    <xacml-samlp:XACMLAuthzDecisionQuery
      ID="ebbe3040d38fe750fd789deba86685b8" InputContextOnly="false"
      <saml:Issuer
          <xacml-context:Attribute
            AttributeId="http://evidian.com/security/authorization/demo/http-method"
            DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="http://evidian.com/security/authorization/demo/filter-issuer">
            <xacml-context:AttributeValue>GET</xacml-context:AttributeValue>
          </xacml-context:Attribute>
        </xacml-context:Action>
      </xacml-context:Request>
    </xacml-samlp:XACMLAuthzDecisionQuery>
  </soap11:Body>
</soap11:Envelope>
```

The request consists of a request context containing the values of attributes from the Subject, the Resource and the Action.

**XACML response:** the decision is “Deny”

```xml
<?xml version="1.0"?>
<soap11:Envelope xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/">
  <soap11:Body>
    <samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
      ID="b92b2060cb088d092b2e1342e439292" InResponseTo="ab7daf28b92a885dacad6dfca4d74f" IssueInstant="2010-10-20T18:03:57.980Z" Version="2.0">
      <xacml-context:Resource>
        <xacml-context:Request/>
      </samlp:XACMLAuthzDecisionQuery>
    </soap11:Body>
</soap11:Envelope>
```

© ComVantage Consortium – 2013
The decision response can be resumed by the Decision element. Here:

```xml
<Decision>Deny</Decision>
```

All this stuff is performed to transmit a boolean (permit or deny) but you get an assertion for the price. And it could have been even bigger if the assertion was signed. Here the assertion is not signed because we used SSL and so the origin of the response is guaranteed. We would have to sign the assertion if we needed to forward it.

As we will see later in the API discussion, it is possible to avoid the XML marshalling and the Web Service cost when the evaluation engine is embedded within the application.

### 2.4.2.4 Benefits of XACML

The XACML model has several advantages:

- The policies can be based on an arbitrary set of attributes (context, subject, resource, etc.).
- The decision can involve a set of obligations that must be fulfilled (e.g. delete the document after the access).
- XACML defines the format of all the messages that will be interchanged among the different parts of the architecture.
- The policy language is based on descriptive logic.
- It is an interoperable system that allows the interchange of policies between different systems, being their components distributed (e.g. PDP does not have to be in the same place as PEP)
The XACML solution applies to any kind of resources. It works with non-RDF resources as well as SPARQL endpoints. XACML policies contain rules that are evaluated using attributes (properties) of the Subject, Resource, Action, and Environment of an access query. The XACML PDP doesn’t analyze the content of the SPARQL queries, but control the access to SPARQL endpoints. The XACML PDP can also be invoked by the target application during the query processing for a finer control.

The XACML prototype developed within ComVantage project can be found in D3.3.1b Prototype of security mechanisms – XACML security mechanisms.
3 MODELLING GUIDELINES
The first implementation of the modelling method reflects a separation of concerns and has been split in two prototypes due to:

- modelling support required earlier than the planned M24 prototype delivery, particularly the necessity to expose models through the RDF model export for further processing, as support for other prototypes (the Industrial App Framework of WP5);
- different granularity of the change requests/agile evolution steps;
- the necessity of enabling certain exploitation activities for University of Vienna;
- the necessity for earlier experimentation to support scientific papers.

Thus, the task resulted, for this period, in:

a) An early prototype (“the OMI modelling prototype”) has been deployed on the Open Model Initiative portal for registered members\(^1\), with RDP on-line access. This was necessary for development steps that required collaborative modelling support, mainly for the app orchestration and requirements (WP5) and the scenario refinements. For this purpose, some change requests had to be incorporated earlier, based on hands-on modelling experience from partners.

b) The ComVantage Modelling prototype has been developed according to the initial roadmap, capturing the more stable elements of the method specification, where change requests and evolution are bulkier and do not impact in any way the run-time developments of ComVantage. This prototype currently captures supply chain modelling aspects (as specified in D3.1.1, including some application area adaptations) and is assimilating lessons learned from the OMI prototype evolution.

The main commonality of the two prototypes is the process-centricity and the collaborative aspects. Several types of collaboration are being supported:

- Collaboration as a modelling concept, is represented in business processes by assigning roles and organisations to various steps or fragments of the control flows, that can also be visually suggested with swimlanes. Control flow patterns relying on parallelity, delegation of work and work synchronisation are captured both visually, by modelling means, and in the machine readable format to be consumed and further processed by process aware information systems (for example, by the orchestration engine developed in WP5);

- Collaborative modelling is supported by providing the OMI modelling tool in the Open Model Initiative community space, where:
  - different modellers can work on the same model or can share them in read-only or editable regimes, depending on the access rights established for model folders;
  - editable note objects can be used to communicate issues between remote modellers working on the same model;
  - concept reuse can be achieved by linking models to elements from a common resource pool or by linking different models and navigating those links in the tool; these links are preserved outside of the modelling tool in the RDF serialization of models so they also enable model linking at RDF level.

The means of accessing the prototypes and of joining the OMI community space are presented in the D3.4.1. companion document. This chapter only presents the usage guidelines for the current status of the evolving modelling tool support. It is split in two subchapters, each addressing one of the mentioned prototypes. The OMI prototype guidelines (chapter 3.1) also include a “General User Interface” section (3.1.1) which describe generic functionality independent of the modelling language, that is largely similar in both prototypes and will not be duplicated for the ComVantage modelling prototype (chapter 3.2)

\(^1\) http://www.openmodels.at/web/comvantage/home
3.1 OMI PROTOTYPE GUIDELINES

3.1.1 General User Interface

The prototypical implementation is built on the metamodeling platform hosted by the OMI portal. An overview of the platform’s user interface template (extended for our implementation) is shown in Figure 21. For the guidelines it is assumed that the user already has some knowledge of using software in general (e.g. how to save, what the “Open” menu does, how to copy and paste, drag and drop, tooltips etc.) as well as some basic knowledge about conceptual modelling. Therefore, these guidelines will mostly present one way to achieve some modelling goal. The user is invited to experiment with the platform and prototype on their own. Additional help can be found in the “Help” menu of the platform.

![Prototypes general user interface](image)

Figure 21: Prototypes general user interface

Certain elements like the Explorer and the Navigator can be closed to make more room for the Modelling area. If they are missing, they can be reactivated through the “Tools” submenu in the “Window” menu at the top. The user interface also changes depending on the selected component. Changing the component mostly influences the available menus and toolbar icons. The components can be switched using the corresponding icons in the toolbar shown in Figure 22.

![Available components](image)

Figure 22: Available components
The relevant components in this prototype are Modelling, Analysis and Import/Export. They are described in further detail in the following sections.

3.1.1.1 Modelling

The Modelling component deals with the creation, modification and management of models inside the platform. For this the Explorer is most helpful since it provides a list of all models available to the user. The models themselves are organized in a tree of model-groups. The structure can be compared to the general file system where files (models) are inside folders (model-groups).

To create a new model right click in the Explorer on the model group where it should be created and select the desired model type in the sub menu “New” of the context menu. This can be seen in Figure 23. Model-groups are created in a similar fashion.

![Figure 23: Create a new Process model](image)

To rename a model right click in the Explorer on it and select “Rename” in the context menu. This will open a dialog asking for the name and version for the model. The version is shown as a suffix to the model name and both merged can be considered the full name of the model. At no time can there be two models of the same type with the same full name, independent of the model-group they are in.

An additional functionality has been added to the OMI modelling prototype in order to clone models. This is necessary since several partners are working on the models with different read and write accesses and sometimes they need to copy models in order to edit them in their own folders rather than working on the original copies. Through the cloning a set of models can be copied with proper relations between the clones (according to the original models). The functionality can be accessed in the “Cloning” menu through the “Clone” item. This will show a popup where the models to be cloned have to be selected. After clicking on OK a new popup will ask in which model-group the clones should be created. Selecting OK will show a third popup asking for a suffix which will be attached to the name. This is necessary since the full name of a model has to be unique. After all this information has been provided the clones will be generated in the target model-group.

To create objects in a model first open it by double clicking on it in the Explorer. This will open and show the model in the Modelling area and the available concepts for that model type in the Modelling elements bar. Once the model is opened simply select the desired concept (e.g. Process start, Activity etc.) from the Modelling elements bar and click in the Modelling area where the object should be placed (see Figure 24).

---

2 Those access rights have been set to prevent one partner from changing someone else’s models.

3 The list of available concepts is also influenced by the selected mode, which can be changed through the “Mode” submenu of the “View” menu.
Relations (e.g. Sequence relation) on the other hand are drawn between two objects, a source and a target object. To create a relation first select it in the Modelling elements bar, then select the source of the relation and then the target. Additional points called bend points can be added to put angles in the relation. To do this select the relation, click on its line and drag it to a new place in the Modelling area. The bend point is represented by a white rectangle when the relation is selected. To remove it, simply drag it over an already existing bend point. In addition to the bend points two other things are visualized when a relation is selected. The position of text displayed by the relation is represented by a green diamond and the yellow circles allow changing the source and target objects. An example relation with six bend points and “Some text” attached to it between two objects can be seen in Figure 25. In case a relation does not have a notation or is between objects in different models it is implemented as a reference. This means it can be found as an attribute of the source object.

The model itself has a certain size which can be changed. A new model has by default the size of an A4 sheet. Also the size of the model can never be smaller than the space that is required by its contents. To change the size of a model, usually making it smaller, click and drag the bottom right corner of the canvas as seen in Figure 26. Note that this changes the size of the model, not the zoom factor. To change the zoom, select “Zoom” in the “View” menu or double click on the percentage value at the bottom right of the main window.
In order to change the attributes of an object, double click on it in the Modelling area which opens the Notebook containing its attributes with values. Models themselves can also have attributes. The Notebook containing the model attributes can be accessed through “Model attributes” in the “Model” menu. An example for a Notebook can be seen in Figure 27. In the Notebook the attributes are organized in chapters. The arrows at the bottom of the Notebook can be used to change pages in case a chapter contains more attributes than can be shown. The type of the attribute controls how it is shown in the Notebook and how it can be edited. In this platform the type of the object, its name and the model to which it belongs are used to identify it. This means that one model cannot contain two objects of the same type with the same name.

Attribute actions like the ones seen in Figure 28 can appear above an attribute field. Which ones are shown depends on the attribute type and available information. Their meaning from left to right is:

4 If it is absolutely vital that two objects pretend to have the same name then whitespaces at the end of the name can be used as a work around.
1. Add – Adds a row to a table or an inter-model relation (relation between different models).
2. Delete – Removes the selected row or inter-model relation.
3. Execute / Follow
   a. Execute passes certain attributes to the operating system.
   b. Follow jumps to the target of an inter-model relation.
4. Browse – Allows selecting a file through the file browser of the operating system.
5. Dialog – A special dialog which supports editing the attribute value.
6. Info text – Provides an information text about the attribute.

Figure 28: Attribute actions

It is important to remember that relations which connect objects between two different models or which don’t have a notation (here called references) are also found in the Notebook of the source object together with the attributes. To see if and which such relations point towards a certain object, or in other words find out for which relations the object is the target of, simply right click on the object (“the target”, for example an instance of Role) and select “Object references” in the context menu as shown in Figure 29. This will show a dialog with all the objects which are in a relation with “the target” and from which model they are. They are grouped by the relation type in which they participate (e.g. Assigned role) and their object type (e.g. Activity). Selecting an object in this dialog and clicking on “Follow” will jump directly to that object and of course open the corresponding model.

Figure 29: Traverse a relation without a notation in the opposite direction

The Modelling component also provides functionality to create a graphic out of the selected model. It can be accessed through the menu item “Region” which can be found in “Generate graphics” in the “Edit” menu. This will show a dialog where further details can be specified (e.g. file type, storage location, zoom etc.). It will create a graphic for the selected region in the currently open model, which by default is the whole thing. The region can be set by holding down the ALT key on the keyboard, performing a left click with the mouse in the Modelling area where one of the corners should be and dragging towards the opposite edge to create a region. The current region is represented by a blue rectangle which can be resized like a window and removed by clicking anywhere on the Modelling area.

To do this for a model just right click on an empty space in the Modelling area and select “Model references”.

---

© ComVantage Consortium – 2013
3.1.1.2 Analyzing

The Analysis component allows executing queries using the proprietary language of the platform. However, the platform also provides a dialog which can be used to build simple queries. Two Examples for simple queries executed using the Analysis component are “What Activities have a cost higher than X?” or “Which Roles have no description specified?” Generally speaking the first example allows finding potential for optimization while the second example supports identifying incomplete objects.

To access the query dialog first select “Queries/Reports” in the “Analysis” menu as shown in Figure 30. In the following dialog specify on which models the query should be executed. This will open the query building dialog as shown Figure 31. The top part provides some forms for query building blocks. By pressing the “Add” button the building block will be added to the actual query using the proprietary syntax in the bottom part of the dialog. The “and”/“or”/“diff” operators for the query can also be added there. Selecting “Execute” will run the current query on the previously selected models. An example result for a query asking for activities with costs of higher than 20 can be seen in Figure 32. For additional help use the corresponding button.

Figure 30: Open query dialog

Figure 31: Dialog for building simple queries

6 Make sure the Analysis component is selected in the tool bar if the menu isn’t there.
3.1.1.3 Importing and Exporting

The Import/Export component allows storing models in files outside of the platform, where they can then be used as backups or for further processing. The models can be exported or imported from either the proprietary format ADL or using XML. The XML files of the export follow a Document Type Definition (DTD) which is also created when exporting one or several models. In those guidelines only the XML export and import is covered, since it is more versatile and usually enough for most cases.

To start the export first select “XML Export (default)” in the “Model” menu as shown in Figure 33. This will open a dialog where the models and model-groups have to be selected for the export as seen in Figure 34. Take note of the checkboxes below the list for some additional export options. Some of those only become available when certain criteria are met (e.g. a model-group is selected). When using “Including referenced models” the reference depth has to be specified through the “References” button. Under “Export file” at the bottom of the dialog the location for the XML file to be created has to be provided. Clicking on “Browse” will open the file selection dialog as seen in Figure 35. However, be careful about the file selection. Since the prototype is running on a remote computer this also means that the “C:” drive is the local one of that computer. Instead, choose a drive in the “Other” category (e.g. “C on DELL 755-4” in Figure 35). After the location for the XML file has been specified a click on the “Export” button will start the export. When it is finished a result dialog will be shown. In addition to creating the file with the model data a file (“adoxml31.dtd”) containing the DTD for the XML structure is created in the same folder.

Make sure the Import/Export component is selected in the tool bar if the menu isn’t there.
D3.5.1 Guidelines for the secure collaboration model

Figure 34: XML export model selection

Figure 35: XML export file selection
To import models from an XML file select “XML Import (default)” in the “Model” menu, right above “XML Export (default)” in Figure 33. This will open the file selection dialog similar to Figure 35. Here select the XML file to import the models from. Again remember that the prototype is running on a remote computer and “C:” is that computer’s local drive. Note that the XML file has to follow the DTD provided by the platform, which is created when a model is exported. Also the DTD file has to be in the same folder as the XML file. Otherwise the XML parser will throw an error, because it cannot find the DTD. After selecting the file a dialog as shown in Figure 36 will ask where to import the models and model-groups. Again several options are available at the bottom of the dialog, like how to handle conflicts or to import the model-groups as well. After selecting “OK” the import will be performed and a result dialog will be shown.

![Figure 36: XML import model-group selection](image)

3.1.2 ComVantage Modelling Guidelines

Until now the guidelines have been rather general on how to create and maintain models using the modelling platform. In this section we will focus on providing guidelines specific for the ComVantage project. First in section 3.1.2.1 some guidelines on how to create proper models will be given. Afterwards in section 3.1.2.2 some procedures describe how those models can be used to achieve a certain goal.

3.1.2.1 Model Type Guidelines

The current prototypical implementation makes use of four model types. For each of those some guidelines will be provided on how to create them.

3.1.2.1.1 Model Type Independent

One object type which is available in each model type is the Note. Through the Note, text for the human reader can be placed in a model. It can be used to describe assumptions that have been made or to point out problems or errors in a model which have to be solved or at least documented. The text is provided...
through its table attribute, which can have a date and an author attached to it. The Note will always show the last text entry in the table. Since it is drawn behind most objects it can also be used to seemingly group objects for the human reader. Additionally the Attached to relation can be used to attach notes to elements. Also a link to an image file can be provided which is then shown in the place of the Note. The supported file types are BPM, JPG, GIF and PNG among others. With all this it is however important to remember that the Note itself is not and should not be used beyond anything than to provide information to the human reader.

Also as a general recommendation the flow of the main relations in models should adhere to the direction of reading. This means that models describing a sequence should have the objects arranged so the main flow goes from left to right and top to bottom. Models depicting a hierarchy on the other hand should have the root element at the top which is then decomposed towards the bottom.

3.1.2.1.2 RESOURCE POOL

The Resource pool model is used as a central repository to define objects which are then reused through other models. The central resource types are Mobile IT support features, Information resources and Roles (think human resource). Additionally some simple organizational structures can be described using Organization objects and can be used to structure Roles. Roles can also be described in a hierarchy to indicate that one role also covers another. Also Capabilities are described in the Resource Pool, in order to avoid the creation of a new model type containing only Capabilities.

The Mobile IT support features can be compared to an app. They represent a feature that can be executed, support a user in some task and are meant to be run on a mobile device. The scope of a Mobile IT support feature is not fixed to a certain size and can therefore represent small and big apps or composed apps. Composed apps can use a reference to an Orchestration model, indicating from what other Mobile IT support features they are composed and how those are ordered. They can also be described through their input and output types, their capabilities and a Mobile IT support model (see section 3.1.2.1.4) providing further details.

Information resources represent a collection of meaningful information and can be generally categorized in tangible and intangible resources. Tangible Information resources represent information on a medium that is tangible, like contracts or invoices on paper. Intangible resources on the other hand are mostly digital, where information can be stored on a server and flow to different clients to be read. This differentiation is of interest for security reasons since tangible information has to be accessed and secured differently than intangible information. Additionally there is a special intangible Information resource type to denote linked data. The Information resource can also be further detailed by describing what it provides and how it can be accessed.

With a Role a certain task range can be described either in regard to function (e.g. “SAP Expert”) or in form of a position (e.g. “CEO”). Roles represent a certain skill-set that is either required by an Activity or covered by a human that can take on the role. They can be assembled in a hierarchy using the Is-a relation. It is easiest to understand it as substitution, where the use of the target of the Is-a relation could be replaced by the source of the Is-a relation. The roles can also be assigned to an organization by creating an Organization object and putting the role inside of it.

An example with Roles in a hierarchy and in Organizations can be seen in Figure 37. There are two organizations each containing one role. Also the hierarchy states that “SvTn”/“ME” can also be considered a “Tech”, meaning that whenever a “Tech” is used, it could be replaced by “SvTn” or “ME”.

© ComVantage Consortium – 2013
Figure 37: Example for Roles in Organizations

For each of those central resources the security policy requirements can be described through their “Access control requirement” table. More details about this can be found in section 3.1.2.2.2.

Capabilities represent abstract features that can be provided or required by for example a Mobile IT support feature. Describing the capability requirements in a process and also providing the available capabilities by an app can be used together to find app candidates for the process. They can be of a certain type like “Functionality” or “User interaction”. Also a URI can be used to link them to further descriptions for example in the linked data cloud.

3.1.2.1.3 PROCESS MODEL

A Process model describes a sequence of activities performed by someone impersonating a role to achieve a certain goal or reach a specific outcome (e.g. solve some issue, create a product etc.). The process should describe what has to happen in order to achieve the goal/outcome, not what could also happen besides it. For this it uses some more or less simple concepts:

- **Process start** – to indicate where to begin.
- **Activity** – to represent a task that is performed.
- **Process end** – to indicate that the process terminates.
- **Sequence relation** – to describe the execution order, they connect two of the other concepts.

Any process model should not have more than one Process start and one Process end and should only contain Activities which are related to the process. Also none of the above concepts should have multiple outgoing Sequence relations. Two additional concepts are provided to control the sequence flow in a process:

- **Decision** – representing the need to decide on something and depending on the outcome to choose one specific outgoing path.
- **Hub** – represents a point where several paths can be taken in parallel (splitting hub) and where the paths have to be synchronized again (joining hub). The differentiation between those two is made through the amount of incoming and outgoing sequence relations.

The safest way to think of those is that only one path is taken after a Decision similar to an exclusive OR and all paths are taken after a Hub similar to an AND. The execution order of paths after a Hub doesn’t matter and if the necessary resources are available they can also be performed at once. The sequence of those individual paths executed in parallel still has to be adhered to. Unlike the other concepts both the Decision and the Hub can have multiple outgoing Sequence relations. Also the “Transition condition” of a Decision’s

8 Still a single Hub should not have both multiple incoming and multiple outgoing Sequence relations. In such a case just use two Hubs after another.
outgoing Sequence relations should be specified in order to know which path to take depending on the choice made. A correct process using Decisions and Hubs properly can be seen in Figure 38. In the example also Swimlanes are used, which allow to structure the model in a certain way (e.g. by performing roles). The Swimlane however only has a meaning to the reader of the model.

<table>
<thead>
<tr>
<th>LM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Deg</td>
<td></td>
</tr>
<tr>
<td>TL</td>
<td></td>
</tr>
<tr>
<td>A-Eng</td>
<td></td>
</tr>
</tbody>
</table>

Figure 38: Example WP6 process of multiple roles collaborating

In order to create Process models that can further be processed some rules have to be considered. First, for each splitting Hub (i.e. one with multiple outgoing sequence relations) there has to be one corresponding joining Hub (i.e. one with multiple incoming sequence relations). In other words Hubs must be used in pairs and each Hub can only be part of one such pair. The elements and paths between the splitting and joining Hub (the Hub pair) are considered to be inside the parallelism. Second, there should also be no Sequence relations that jump between the different paths, especially not between one path inside and one path outside of a specific parallelism. Figure 39 shows one correct example and two wrong examples which break one of the above rules.

Figure 39: Right and wrong parallelisms
The cases for an exclusive decision and for a parallelism are covered by the Decision and Hub elements. However, there is still the case that not all but only some paths should be taken in a parallelism, similar to an OR. This can be achieved by combining Hubs and Decisions. Simply put a binary decision at the beginning of the parallelism path (i.e. right after the splitting Hub) and make one of its outgoing Sequence relations go to the corresponding joining Hub. An example for this can be seen in Figure 40. The decision does not necessarily need to lead directly to the joining Hub, but can also jump only over some activities as long as the above rules for the parallelism are adhered to. The benefit of this is that it rigorously describes what should happen and avoids the case where none of the paths of an OR would be taken.

Still, even with all those different ways of controlling the sequence flow, the major parts of the process are the activities. At the very least the name of an activity should indicate what is happening, therefore it should at least contain a verb and additional words providing context to the verb (e.g. “process order”, “execute heat test”, “analyse cycle time”). A Role is assigned to an activity through the “Assigned role” attribute and indicates who is responsible for the activity. If the activity is not decomposed further then it is also assumed to be performed by the assigned role. If possible the roles should be assigned to the activities. Activities can also be described using many different attributes and references, like the time necessary to perform the activity, the cost associated with their execution, the different resources needed for the execution and the capabilities an app should provide among others. If necessary the activities of a process can also be further decomposed and described by other processes by using the Referenced process reference.

The collaboration between different organizations can be described through the roles assigned to activities. Figure 38 shows a process where four different roles are collaborating to achieve a common goal, in this case the correct cycle time for a machine. It has both parts where activities are performed in parallel by different roles (e.g. “Open Issue” by the role “LM” and the activities of the role “TL”) as well as activities that cannot be performed until someone else has finished their work (e.g. “File BOM’s for PR” by the role “TL” cannot be performed until “Dsg” finishes “Provide necessary Docs and Info” and decides whether spare parts are needed).

In ComVantage the Process model can be used to describe processes as three different views:

- Business view – focuses on activities and their sequence which are necessary to achieve a goal (i.e. the result). The Business view should concentrate on what the human has to do and avoid technical details about the execution (e.g. visualize data). An example can be seen in Figure 38.
- App requirements view – focuses more on the App requirements and sequence. It can therefore look more like a series of service (or app) calls. It can often be created by splitting and merging activities from the corresponding Business view. The Mobile IT support reference should be used to indicate which activities are supported by which apps. An example can be seen in Figure 41, where most activities reference a Mobile IT support feature (indicated by the small black rectangle in the top left of the activity).
Interaction view – describes how an app or app orchestration should be interacted with. The result of the process should be the same result as intended by the app it is for. Here each activity must use the Interacts with POI reference to indicate with what POI the user should interact. An example can be seen in Figure 42, where the activities link to POIs of a Mobile IT support model (see section 3.1.2.1.4).

The view of a Process model can be set in its model attributes. Preferably a single process should not mix views, since otherwise it can get confusing to the person reading the model. Also when referencing a process from an activity, the processes view level should be on the same level as the activities or a “lower” one. For this the order from highest to lowest views is 1) Business view, 2) App requirements view and 3) Interaction view. If an App requirements view has to be described on the same granularity as the Business view, then the View link element can be used to create a link from the process in the Business view to the process in the App requirements view (or Interaction view). For example the View link element (the rhomb/diamond shape) at the top left in Figure 38 links to the process depicted in Figure 41.

3.1.2.1.4 MOBILE IT SUPPORT MODEL
The Mobile IT Support model describes interaction requirements. The Mobile IT Support model can be defined in two different views, depending on the value of the “Abstraction level” attribute:
1. **Abstract UI** – Describes a mobile app in terms of requirements for abstract app interaction. This view is independent of any interaction modality (e.g. haptic, voice, etc.) or visual appearance of the single components of the mobile app. It should suggest the app components and how they can trigger each other, thus giving a structural view on the app (to be complemented by the dynamic view of the interaction process indicating interaction steps for each element).

2. **Concrete UI** – Describes a mobile app through mapping the abstract UI on a concrete visualization for a certain user context and interaction modality, in this case haptic modality. The visual appearance (e.g. size, position, colour etc.) of the single components is relevant at this level in order to provide a suggestive emulation of the user interface, mainly for purposes of requirements validation. It is not intended to generate concrete UI code, as the focus of ComVantage is placed on app orchestration.

In order to model the design requirements for a certain mobile app the developer is provided with following concepts:

The concept of *Screen* is used to represent a single surface which is presented to the user while interacting with the mobile app. Although the name suggests that it was defined for haptic modality, it should be seen here as a rather abstract grouping of components for user interaction, called *POIs*, and other components (*Composition* and *Complex components*) for building more complex and elaborate interactions. A *Screen* object can be assigned to various types of devices with the “*Fragmentation level*” attribute. Another provided feature is the specification of UI parts which are repeated across multiple *Screen* objects. In order to define a repeated UI part it is necessary to define a *Screen* object of fragmentation level *View* and define the repeated *POIs* inside of it. This *Screen* object can then be reused by referencing it through the “*Permanent*” attribute in other *Screen* objects. An example of the *View Screen* object is visible in Figure 43.

![Figure 43: Example of View Screen object linked to a Screen object.](image)
Point of interaction (POI) represents the most important part of the Mobile IT Support model for the developer of a mobile app. Through this concept it is possible to define different interactions with the user by selecting values in the “Abstract UI type”. This concept is divided into two types, called Readable and Interactive POI. The Readable POI is used to represent components of the Screen which provide information to the user. Meanwhile, Interactive POIs are used to represent components of the Screen which provide the user with direct interaction possibilities. Through the “POI behaviour” attribute of Interactive POIs it is possible to define how each POI is influencing (e.g., read, create, delete, etc.) the data of other POIs. Another feature of the POI class is the possibility to build list of triggers which are used as a menu in the UI. Both POI types provide attributes which can be used for concretizing the UI of the mobile app and to specify the expected behaviour by using them. In this case the Simple selection value from the “Abstract UI type” attribute has to be selected and the possible menu items are defined as entries in the “Options” table which can be found under the chapter “Selection”.

The Composition component provides the developer with the option to specify components of the UI which are repeated (e.g. tables, lists, trees, etc.). The Composition component is defined as an aggregation which takes the aggregated POIs and repeats each in the defined composition type. The type of composition can be selected by selecting values from the “Type” attribute in the notebook. Additionally it is also necessary to specify the name of the Composition component inside of the “Composition name” attribute of the used POI.

The Complex component concept allows the specification of components of the UI which provide a combination of different interactions (e.g. “selecting and showing pictures”). The Composition component is defined by creating the POIs inside of the Complex component aggregation. An example of a Composition component which provides combination of two interactions (showing pictures and changing the pictures) is visible in Figure 44.

Figure 44: Example of a Complex POI which provides interactions for changing a map.

In order to represent the navigation from one Screen to another, the Triggers Screen relation is used. This relation is defined between the concept of POI (enables the triggering) and the concept of Screen (represents the output of the triggering). The Triggers Screen relation is usually defined as a one way relation. However by using the “Backable” attribute it can be specified that the “back” functionality of the device can be used to return to the previous screen. Last but not least, the developer is also able to specify the intended execution environment of the mobile app by specifying the device type, operation system (OS) and OS version. Those attributes can be found under the model attributes of the Mobile IT Support model.
In Figure 45 an example of a Mobile IT Support model specified as Abstract UI is presented. This model contains two different Screens, each having one Composition component and three different POIs. In both Screens each Composition component contains one POI, but of different “Abstract UI type” values. Besides, it can be seen that it is possible to get from the “Current values” Screen only with one interaction, and from the “Change actuators” Screen it is possible to get back to the previous Screen by two different interaction modalities, either by using the device specific “back” button or by triggering the “Change values” POI.
In Figure 46 a copy of the model from Figure 45 is used, but this time with the “Abstraction level” attribute set to Concrete UI and different sizes for the POIs. In Figure 46 it can also be distinguished what the different UI representations of the single POIs are. Additionally, the Composition components change the representations of the POIs inside of them. While on the abstract representation positioning is irrelevant, since it only aims to define a grouping of interaction channels in app features, the concrete one can serve as an app emulation for validating app requirements. For the abstract representation, the screens should be seen rather as app features (not necessarily implemented in single screens) and the triggers should be seen as ways of getting from one feature to another (not necessarily implemented in buttons).

3.1.2.1.5 Orchestration Model

The Orchestration model is used to describe the sequences of resource usage for a certain process. In this prototype it currently supports the orchestration of Mobile IT support features which can then be used by an app orchestration.

The two general concepts in the Orchestration model are the resource and the Followed by relation. A Resource represents the thing that is used and the sequence is described through the Followed by relations, which can contain “Transition conditions”. In this regard they are similar to the Activities and Sequence relations from the Process model. However it differs from the Process model in its intention and how it achieves it. First the orchestration is for a process and is not about executing activities but using and reusing resources. Therefore it is not focusing on activity execution time, activity execution costs, the proper interaction with resources or what multitudes of different resources are used in a single step.

Second the orchestration is concentrating on the resource usage for a certain collaboration partner, meaning it is from this collaboration partner’s view. However to properly depict the collaboration it also uses the additional concepts of Suspension points and Notifications. Suspension points allow specifying a point in the sequence where some message or notification is necessary to continue. They are defined by two types: “Entry” and “Dependency”. An entry type Suspension point indicates that the orchestration can be entered here when a certain notification is received, while a dependency type Suspension point specifies that a notification containing additional data is necessary to continue further. Notifications allow describing what should be sent and, if applicable, also towards which activity. Those notifications as well as the ones used in the Suspension points are modelled through Information resources in the Resource pool.

An example for an Orchestration model where the sequence of Mobile IT support features for the A-Eng role can be seen in Figure 47 (the corresponding process can be seen in Figure 41). In this case instead of looping back from “Notification distribution” to the beginning it is assumed that the orchestration will simply be executed a second time when a “check cycle time” notification is received.
The orchestration model can also contain *Synchronization points*, which indicate at what places certain paths have to be synchronized in case they have been previously split. However, how splitting and synchronizing (merging) paths works is dependent on the used orchestration engine.

### 3.1.2.2 Modelling Procedure Guidelines

Several results can be achieved by creating models in the current prototypical implementation. What exactly they are and how to achieve them is described in the sections 3.1.2.2.1 to 3.1.2.2.4. Those procedures represent guidelines and as such it might be useful to stray from the here provided path depending on the case. Modelling does not always follow a rigid procedure. Sometimes creating a certain model provides new insights on the topic that requires changing previously created models. For example when modelling a process one might realize that a certain *Role* is missing in the *Resource pool*. Therefore going back and forth between procedure steps is acceptable.

#### 3.1.2.2.1 MODEL EXPORT AS RDF

In the current OMI prototype implementation it is possible to expose the models using RDF, in order to upload them in a Linked Data environment. Through this, the models or their parts can further be linked to other descriptions and elaborate queries can be executed by using for example SPARQL. The general procedure to accomplish this is as follows:

1. Create models in the prototype (details in sections 3.1.1.1 and 3.1.2.1)
2. Export the desired models as XML (details in section 3.1.1.3)
3. Use the additional RDF Export files which are obtained from the ComVantage page of the OMI portal after registering\(^9\). Further up to date details on how to use the RDF Export and how to upload exported models on a Linked Data server can be found in a PDF manual, together with example files in the ZIP archive available there.

An overview of how the current RDF Export user interface looks can be seen in Figure 48.

---

\(^9\) [http://www.openmodels.at/web/comvantage/home](http://www.openmodels.at/web/comvantage/home), the files are available for registered OMI users or provided as companion files to the prototype deliverable D3.4.1
Figure 48: The RDF Export user interface

Figure 49 shows an example for a Process model and a Resource pool and Code 1 contains the resulting RDF in TriG syntax. Some unnecessary attributes (e.g. for debugging or only for visualization purposes) have been removed from example RDF to reduce the amount of space needed. In Code 2 an example query asking for all Roles that need access to an Information resource can be seen\textsuperscript{10}. The query example also covers the role hierarchy. Further details about the RDF Export structure and examples can be found in D3.1.1 and the metamodel description provided with the RDF Export files. Since D3.1.1 the way tables are represented has been changed. Now they are also described using an RDF structure instead of just plain XML data to allow querying them as well. For this an attribute of the type “table” is represented as an RDF list and each row is a node in this list. Those row nodes then have RDF properties towards their column values (e.g. the “cv:Uses_resource” property of an Activity in the example RDF).

\textsuperscript{10} Tested using the openRDF sever Sesame and its workbench 2.6.9.
Figure 49: Example process and resource pool

@prefix : <http://www.comvantage.eu/example#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix cv: <http://www.comvantage.eu/mm#>.

:Process_model-MaintenanceModelBPM {
  :Sequence_relation-246761-Decision-167809-Repair
    a cv:Relation_class , cv:Sequence_relation ;
    cv:Transition_condition "Yes" ;
    cv:Transition_probability "1" ;
    cv:from_instance :Decision-188636-Decision-167809 ;
    cv:to_instance :Activity-188639-Repair .

  :Sequence_relation-247404-Decision-167809-Process_end-167815
    a cv:Relation_class , cv:Sequence_relation ;
    cv:Transition_condition "No" ;
    cv:Transition_probability "1" ;
    cv:from_instance :Decision-188636-Decision-167809 ;
    cv:to_instance :Process_end-188642-Process_end-167815 .

  :Process_start-188630-Maintenance_start
    a cv:Instance_class , cv:Process_start ;
    cv:Name "Maintenance start" .

  :Activity-188639-Repair
    a cv:Activity , cv:Instance_class ;
    cv:Activity_cost "150" ;
}
D3.5.1 Guidelines for the secure collaboration model

:Role-188620-Maintenance_Technician

:Information_resource-246749-Machine_Documentation

:Activity-188633-Analyse
  a cv:Activity , cv:Instance_class ;
cv:Activity_cost "50" ;
cv:Adds_value "yes" ;
cv:Assigned_role :Role-246742-Expert ;
cv:Name "Analyse" ;
cv:Prefer_same_role_in_subprocess "yes" ;
cv:Time "00:000:02:00:00" ;
cv:Uses_resource (:Activity-188639-Repair-Uses_resource_1) .

:Activity-188633-Analyse-Uses_resource_1
  cv:Action "Use/Read" ;

:Activity-188639-Repair-Uses_resource_1
  cv:Action "Use/Read" ;

:Sequence_relation-188646-Analyse-Decision-167809
  a cv:Relation_class , cv:Sequence_relation ;
cv:Transition_probability "1" ;
cv:from_instance :Activity-188633-Analyse ;
cv:to_instance :Decision-188636-Decision-167809 .

:Sequence_relation-188648-Repair-Process_end-167815
  a cv:Relation_class , cv:Sequence_relation ;
cv:Transition_probability "1" ;
cv:from_instance :Activity-188639-Repair ;
cv:to_instance :Process_end-188642-Process_end-167815 .

:Role-246742-Expert

:Information_resource-246746-Sensor_data

:Decision-188636-Decision-167809
  a cv:Instance_class , cv:Decision ;
cv:Name "Decision-167809" ;
cv:Question "Broken?".

:Process_end-188642-Process_end-167815
a cv:Process_end , cv:Instance_class ;
cv:Name "Process end-167815".

:Sequence_relation-188645-Maintenance_start-Analyse
a cv:Relation_class , cv:Sequence_relation ;
cv:Transition_probability "1" ;
cv:from_instance :Process_start-188630-Maintenance_start ;
cv:to_instance :Activity-188633-Analyse .

:Resource_pool-MaintenanceModelBPM {
:Organization-188623-Maintenance_Org
a cv:Organization , cv:Instance_class ;
cv:Name "Maintenance Org" ;
cv:contains :Role-188620-Maintenance_Technician ,
:Role-247406-Maintenance_Planner .

:Activity-188639-Repair
 cv:Assigned_role :Role-188620-Maintenance_Technician ;

:Role-246742-Expert
a cv:Instance_class , cv:Role ;
cv:Name "Expert" ;
cv:Preferred_occupation "6" ;
cv:Type "Function" ;

:Information_resource-246746-Sensor_data
a cv:Information_resource , cv:Instance_class ;
cv:Access_control_requirements
 (:Information_resource-246746-Sensor_data-Access_control_requirements_1) ;
cv:Name "Sensor data" ;
cv:Type "Intangible" .

:Role-188620-Maintenance_Technician
a cv:Instance_class , cv:Role ;
cv:Is-a :Role-246742-Expert ;
cv:Name "Maintenance Technician" ;
cv:Preferred_occupation "4" ;
cv:Type "Function" .

:Information_resource-246746-Sensor_data-Access_control_requirements_1
 cv:Action "Use/Read" ;
cv:Environmental_constraints "Maintenance ticket created" ;
cv:Subject :Role-246742-Expert ;

:Role-247406-Maintenance_Planner
a cv:instance_class , cv:Role ;
cv:Is-a :Role-246742-Expert ;
cv:Name "Maintenance Planner" ;
cv:Preferred_occupation "2" ;
cv:Type "Function" .

:Information_resource-246749-Machine_Documentation
a cv:Information_resource , cv:Instance_class ;
cv:Name "Machine Documentation" ;
cv:Type "Tangible" .

:Activity-188633-Analyse
   cv:Assigned_role :Role-246742-Expert ;

:Activity-188633-Analyse-Uses_resource_1
   cv:Resource :Information_resource-246746-Sensor_data ;

:Activity-188639-Repair-Uses_resource_1

} cv:graphmetadata {
   :Resource_pool-MaintenanceModelBPM
   a cv:Resource_pool , cv:Model_class ;
   cv:Name "MaintenanceModel" ;
   cv:Version "BPM" .

   :Process_model-MaintenanceModelBPM
   a cv:Model_class , cv:Process_model ;
   cv:Model_state "AS-IS" ;
   cv:Name "MaintenanceModel" ;
   cv:Version "BPM" ;
   cv:View "Business view" .
}

Code 1: Reduced example process and resource pool as RDF

PREFIX :<http://www.comvantage.eu/example#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX cv:<http://www.comvantage.eu/mm#>

SELECT ?name
WHERE {
   GRAPH ?irrep {
      #1 Get the information resource
      ?ir a cv:Information_resource.
      FILTER (?ir = :Information_resource-246746-Sensor_data)
      ?accr cv:Resource ?ir.
   }
3.1.2.2.2 MODELLING ACCESS CONTROL

The current OMI prototype allows modelling two things for access control: access requirements and access policies. The access requirements describe who needs to do what on a certain resource, while the access policies describe who is allowed to do what on a certain resource. Both rely on the concepts of “Subject” performs an “Action” on a “Resource”. The recommended approach is to first describe the access requirements and then base the access policy based on those by performing the following steps:

1. Create a Process model in Business view and describe what has to be done and by whom (i.e. by which Role) independent of resources used. The Roles are assigned through the “Assigned role” attribute of the Activity. Note that unless an Activity is further decomposed it will be assumed that the assigned Role is both responsible and performing the Activity.

2. If necessary decompose the process and describe it in a finer granularity and/or in a different view. The decomposition of an Activity can be done through its “Referenced process” attribute. To link a process on the same level but in a different view use the View link element in the Business view process and the corresponding reference it provides.

3. Reuse existing resources or create new necessary resources and assign them to the activities where they should be used. The resources are assigned through the “Uses resource” table where each row indicates the need to execute an action on a resource. The table contains three columns:
   a. The first column references what resource is used, covering the “Resource” concept of this access control approach.
   b. The second indicates what action will be performed on the resource, covering the “Action” concept of this access control approach. The possible actions have been reduced in our approach to three simple types: 1) “Use/Read”, 2) “Change/Write” and 3) “Full”. The “Use/Read” action indicates that the resource will be used as intended (e.g. use a printer to...
print, use a truck to transport material etc.) or in case it is holding some data that the data will be read (e.g. read order details, read customer information etc.). The “Change/Write” action indicates that the resource will be changed (e.g. change faulty machine part truck, change critical parameters of printer) or written (e.g. write new customer address, append new order etc.). “Full” covers both other types.

c. The third column indicates what environmental constraints might be in place when the action is executed using natural language.

The “Subject” portion can be taken from the Role assigned to the Activity if it is not further decomposed. Additionally the Activity itself can be seen as context for the access to the resource. If the Activity is decomposed then the process referenced through “Referenced process” should contain the details about the necessary access requirements.

Those first steps create the access requirement description. A simple example can be seen in Figure 50 and should be read as: “Expert” (S) needs “Use/Read” (A) access to “Sensor data” (R) [during activity “Analyse”]. An example containing the important RDF statements in TriG syntax can be seen in Code 3.

![Figure 50: Example showing an access requirement description](image)
Now the access policies can be specified from the defined access requirements through further performing the steps:

4. Select the resource in the Resource pool for which to describe access policies. The policies describe what actions should be permitted and everything that is not explicitly stated is considered to be denied. This means that by default if no access policies are described nobody can access any resources.

5. Check the necessary access requirements specified for this resource. The access requirements can then be accessed by either using the functionalities provided by the tool (following references, executing simple queries) or by exposing the models as RDF and executing more sophisticated SPARQL queries like the one shown in Code 2.

6. Specify the access policy for the resource. After the requirements have been gathered the specific access policy should be specified using the “Access control requirements” table of the resource. The table contains six columns:

   a. The first specifies the Role that should have access to the resource, covering the “Subject” concept of this access control approach.

   b. The second allows providing some natural text description about the subject and further restrictions that should be handled in the actual access control implementation.

   c. The third what actions can be performed by the subject, covering the “Action” concept of this access control approach. The meanings of the possible actions are the same as described in step 3 point b.

   d. The forth allows adding some natural text description about the action and further restrictions that should be handled in the actual access control implementation.

   e. The fifth offers the possibility to provide some natural text description about the resource and further restrictions that should be handled in the actual access control implementation.

   f. The sixth allows providing environmental constraints that have to be in place to grant permission.

The “Resource” portion is represented by the element where the table is described.
Those last steps create the access policy description, which can further be used to implement access control that grants the permissions necessary to execute the described processes. A simple example can be seen in Figure 51 and should be read as: “Expert” (S) should have “Use/Read” (A) access to “Sensor data” (R) [when a Maintenance ticket is created]. However, because “Maintenance Planner” and “Maintenance Technician” are considered to be “Expert”, they also should have access to the resource. An example containing the important RDF statements in TriG syntax can be seen in Code 4. Additionally Table 1 provides some examples on how certain access policies can be described using the “Access control requirements” table.

Figure 51: Example showing an access policy description
Example 1: Company X operators can see all configuration parameters of machines that they maintain.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Machine configuration parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>Operator (linked of Company X)</td>
</tr>
<tr>
<td>Subject descr.</td>
<td>---</td>
</tr>
<tr>
<td>Action</td>
<td>Use/Read</td>
</tr>
<tr>
<td>Action descr.</td>
<td>---</td>
</tr>
<tr>
<td>Resource descr.</td>
<td>Only for assigned machines</td>
</tr>
<tr>
<td>Env. constr.</td>
<td>---</td>
</tr>
</tbody>
</table>

Example 2: Engineers related with the project X can see all the properties of project X, except those properties related with cost.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Project properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>Engineer</td>
</tr>
<tr>
<td>Subject descr.</td>
<td>---</td>
</tr>
<tr>
<td>Action</td>
<td>Use/Read</td>
</tr>
<tr>
<td>Action descr.</td>
<td>---</td>
</tr>
<tr>
<td>Resource descr.</td>
<td>Only for assigned project; Except costs</td>
</tr>
</tbody>
</table>
### Example 3: Managers related with the project X can see all the properties of project X, except sensor parameters values.

<table>
<thead>
<tr>
<th>Env. constr.</th>
<th>Engineer and Resource related to same project</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Resource</th>
<th>Project properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>Managers</td>
</tr>
<tr>
<td>Subject descr.</td>
<td>---</td>
</tr>
<tr>
<td>Action</td>
<td>Use/Read</td>
</tr>
<tr>
<td>Action descr.</td>
<td>---</td>
</tr>
<tr>
<td>Resource descr.</td>
<td>Only for assigned project; Except sensor values</td>
</tr>
</tbody>
</table>

### Example 4: T-Shirt manufacturers can see only the stock quantities, only for T-Shirts that they have sold to Company X.

<table>
<thead>
<tr>
<th>Env. constr.</th>
<th>Manager and Resource related to same project</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Resource</th>
<th>T-Shirt properties (incl. “stock quantity” property)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>Manufacturer</td>
</tr>
<tr>
<td>Subject descr.</td>
<td>---</td>
</tr>
<tr>
<td>Action</td>
<td>Use/Read</td>
</tr>
<tr>
<td>Action descr.</td>
<td>---</td>
</tr>
<tr>
<td>Resource descr.</td>
<td>Only T-Shirts sold to Company X; Only stock quantities</td>
</tr>
</tbody>
</table>

### Example 4 alternative: Assuming the information resource has a finer granularity - it is not the whole entity, but only its relevant property.

<table>
<thead>
<tr>
<th>Resource</th>
<th>T-Shirt stock quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>Manufacturer</td>
</tr>
<tr>
<td>Subject descr.</td>
<td>---</td>
</tr>
<tr>
<td>Action</td>
<td>Use/Read</td>
</tr>
<tr>
<td>Action descr.</td>
<td>---</td>
</tr>
<tr>
<td>Resource descr.</td>
<td>Only T-Shirts sold to Company X</td>
</tr>
</tbody>
</table>

### Example 5: Everyone in Company X can see the amount of time spent by the maintenance operators on machines of their own, but they cannot see which operators have been working on them.

<table>
<thead>
<tr>
<th>Env. constr.</th>
<th>Except operator identities</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Resource</th>
<th>Maintenance request properties (incl. “maintenance time” property)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>All (either all Roles, or the highest level Role)</td>
</tr>
<tr>
<td>Subject descr.</td>
<td>---</td>
</tr>
<tr>
<td>Action</td>
<td>Use/Read</td>
</tr>
<tr>
<td>Action descr.</td>
<td>---</td>
</tr>
<tr>
<td>Resource descr.</td>
<td>Except operator identities</td>
</tr>
</tbody>
</table>

### Example 5 alternative: Assuming the information resource has a finer granularity - it is not the whole entity, but only its relevant property.
3.1.2.2.3 MODELLING MOBILE SUPPORT REQUIREMENTS

This subsection describes the necessary steps and guidelines for mobile support requirements modelling. The modelling of mobile support requirements is necessary to establish a better communication between the business expert specifying the requirements for a mobile app and the developer implementing it. Among that, the Mobile IT Support model would partly support the developer in the development process.

The necessary steps can be divided into two parts:

The first part specifies the steps which are necessary in order to specify the requirements for an app from a business perspective. This part should also allow after it is finished to find existing apps for the specified requirements. Necessary steps in this part are following:

1. Define involved Roles. The first step in the modelling of mobile support requirements includes the definition of a Resource pool model with several Roles. The defined Roles represent required skill-sets which should be able to use the created mobile apps.

2. Create a Process model in Business view. In this step a Process model of type Business view has to be defined. The Activities of this Process model have to be connected to the executing Roles from the previous defined Resource pool by referencing them in the “Assigned role” attribute. An example for the Process model and the Resource pool is displayed in Figure 52.

3. Create a Process model in App requirements view. In the next step the App requirements view for the previously made Business view has to be created. A modelling hint is to copy the Business view
model and adapt it. Here the activities can be split into several or several activities merged into one as needed to create a proper App requirements view. After the App requirements view has been created a View link element should be created in the corresponding Business view and reference it. An example of the creation of the App requirements view and references to the Roles of the Resource pool is visible in Figure 53.

![Figure 53: Example of the App requirements view Process model with assigned Roles.](image)

4. Create necessary Capabilities and link them to Activities. In this step the required Capabilities for the app to be developed are defined. The Capabilities of the mobile app can be defined in the same Resource pool as the Roles. Each Capability should have a meaningful name which indicates the capability provided by a mobile app (e.g. “Read machine data”, “Change machine state”, etc.). Each of the created Capabilities should then be linked to Activities of the App requirements view which require support by a mobile app. This assignment of Capability to Activity is done by creating new entries in the “Capability/Initial requirements” table and referencing the required Capability. An example of this step is visible in Figure 54. The defined Capabilities can also be used for the discovery of already defined Mobile IT support features. The discovery of Mobile IT support features can be done either by using the functionalities provided by the tool (following references, executing simple queries) or by exposing the models as RDF and executing more sophisticated SPARQL queries.
The second part describes a scenario in which no existing apps could be found, and therefore it is necessary to create new apps. This is not necessary if a Mobile IT support feature has been found for every Activity which needs app support. The following steps are assumed:

5. Define involved Mobile IT support features and link them with Activities. After the previous step is finished and the App requirements views are defined, it is necessary to define Mobile IT support features in a Resource pool. For this an existing Resource pool or a new one can be used. After the Mobile IT support features are created in the next step the Activities of the App requirements view are identified and connected to Mobile IT support features if they need app support. An example of the linking between Activities and Mobile IT support features is shown in Figure 55.
6. Create initial POI set and link it to Activities. After the Capabilities have been defined and assigned to the Activities the next step is to create for each Capability/Activity link an initial set of POIs which will be used as an initial draft for the definition of Mobile IT Support models. The initial POI set is meant to provide the developer of the app with a set of POI requirements from the business expert’s view. After the initial set of POIs has been created the next step is to connect it to the corresponding Capability entry in the “Capability/Initial requirements” table of the Activity.

7. Create Mobile IT Support model and link to Mobile IT Support feature. In this step the Mobile IT support models have to be created by using the initial set of POIs. Therefore, it is proposed to copy the objects of each initial set of POIs to the newly created Mobile IT Support models and start adapting each model. This adaptation includes also the creation of Screen objects and arranging the POIs inside of it. In order to get a better feeling of the UI the developer can switch the “Abstraction level” attribute to Concrete UI. After each Mobile IT Support model has been created, it is necessary to connect each of those with the corresponding Mobile IT Support feature from the Resource pool. An example for created Mobile IT Support models, the initial sets of POIs it was created from and the connection to the Mobile IT Support Features of the Resource pool are visible in Figure 56.

![Figure 56: Creation of Mobile IT Support features from initial sets of POIs and connecting the Mobile IT Support models to Mobile IT Support features.](image)

8. Define Process model in Interaction view. At this step the developer creates possible guidance for each app. This guidance describes the required steps which the user would have to execute and the sequence of appearing Screens. The required steps for the user are defined through a Process model which is set to Interaction view. Each Activity of the Interaction view should be linked to a POI of a Mobile IT Support model (indicated by a notation switch, in which the icon of the referenced POI is visible). Through the linking of the Interaction view and the Mobile IT Support model it is possible to identify the necessary steps which should be done using a mobile app in order to fulfil a certain task. An example for an Interaction view with references to a Mobile IT Support model is visible in Figure 57.
9. Refinement of Mobile IT Support and Interaction view models to specify final version. After the mentioned Mobile IT Support models and Interaction view Process model have been created, the next step includes the improvement and refinement of the created models. This step is done by using both models and providing it to the intended user group as test cases. The test cases would verify if the intended user group can follow the required steps and fulfill the given task by using the mobile app. Through the results of the test cases it should be possible to identify various lacks and shortcomings in the model, which could then be improved by the developer. This step should be repeated until the final requirements of the business expert are met and the intended user group is able to use the mobile app to solve the given task.

10. Realisation of design. In the last step the created models would be used by the developer as input for the implementation of the mobile app. This step is done by using different technologies and programming languages for mobile app creation and is executed outside of the modelling toolkit. Therefore, this step will only be partially supported by providing the developer with an RDF export of the created models, which can be further used for app creation.

### 3.1.2.4 MODELLING MOBILE ORCHESTRATION

The procedure for describing an orchestration of mobile apps in a model of the current OMI prototype is presented in this section. The goal of the Orchestration model is to be used by an orchestration engine. The procedure aims to provide a coherent way of creating the Orchestration model starting from the requirements described by the Process model. To get from Process model to Orchestration model perform the following steps:

1. Create a Process model in Business view and describe what has to be done and by whom (i.e. by which Role). In this step the Business view that should be supported by the orchestration has to be created and described. It is assumed that the Resource pool with all the Roles and their hierarchy is
already available. If not, then its necessary elements have to be created. The Roles are assigned through the “Assigned role” attribute of the Activity. An example for a properly described Process model in Business view has been shown in Figure 38.

2. Create corresponding Process model in App requirements view. This App requirements view should be based on the previously created Business view. Therefore, it helps to create a copy of the Business view, change its view to “App requirements” and start adapting it. Typically activities from the Business view are split in the App requirements view, but there can be cases where merging of activities might be necessary. The desired granularity is based on the granularity of the Mobile IT support features. In the App requirements view one Activity should have no more than one Mobile IT support feature assigned to it. Also link the App requirements view to the Business view using the View link concept.

3. Link Mobile IT support features to Activities that should be supported. After the basic App requirements view has been created it is necessary to link Mobile IT support features to the Activities that should be supported through the “Mobile IT support” attribute. If necessary create new Mobile IT support features in the Resource pool11. Keep in mind that it is not necessary to provide a Mobile IT support feature for each Activity, only the ones that have to be supported. Also Mobile IT support features can and should be reused where applicable. An example for a properly described Process model in App requirements view with linked Mobile IT support features can be seen in Figure 41. It is based on the Business view from Figure 38.

4. Create the mobile orchestrations based on the App requirements view. Once the Process model to be supported has been described in the App requirements view it can be used to derive one or several orchestrations. The orchestrations are described as Orchestration models. The OMI prototype provides two functionalities to create initial orchestrations based on different assumptions:

a. The first option creates one orchestration for the whole process, based on the assumption that the whole process will be executed on one mobile device by one role. It can be accessed via item “Create resource orchestration” in the “ComVantage” menu when the “Modelling” component is active.

b. The second one creates one orchestration for each role participating in the process (determined through the “Assigned role” of the Activities), based on the assumption that different roles need to collaborate together by using different mobile devices. It can be accessed through the item “Create resource orchestration for each role” in the “ComVantage” menu when the “Modelling” component is active.

Note that neither one promises that a finished orchestration will be created. Therefore manual adaptations are most likely necessary. Typical manual adaptations are:

a. Remove or change endpoints of created Followed by relations.

b. Add transition conditions to Followed by relations.

c. Add necessary Suspension points and Synchronization points.

d. Add the Orchestration element and link it to the corresponding role and process.

Figure 58 shows the automatically generated Orchestration model for the “A-Eng” Role for the process from Figure 41 and the previously shown Figure 47 contains the adapted version of the orchestration.

11 Note that to actually run an orchestration using the Mobile IT support features there also needs to be an implementation for them. However, the implementation of apps is not part of the OMI prototype, but section 3.1.2.2.3 describes some steps on how to capture requirements for an app.
After the orchestration models have been created they can be exposed as RDF and used by an orchestration engine such as the one developed in WP5.

### 3.2 THE COMVANTAGE MODELLING PROTOTYPE GUIDELINES

The ComVantage modelling prototype implementation is based on an improved metamodelling platform compared to the OMI prototype. The notation for most objects / concepts has been reworked to use a cleaner version and to follow a coherent style (e.g. use of gradients). Since this platform is still in development we focused on describing the implemented model types, their object concepts with a short view on the graphical representations, main attributes and a few references to other objects/models (if existent), and an example of each model.

#### 3.2.1 Object Repository

The ComVantage Modelling Tool provides the possibility to use a new feature for object reuse (unavailable in the OMI prototype) known as the Object Repository. We could think of it as an object catalogue where each created object is stored in a common, hierarchical tree in the application and made available to multiple models for reuse. A workaround for object reuse in the OMI Prototype is the Resource pool, but that is rather a model type emulating a repository and not a dedicated feature, as provided by the ComVantage Modelling Tool.

There are some differences between Repository Objects and Non-Repository ones, here called Modelling Objects:

- A Repository Object is managed in the object repository (Object catalogue). On the contrary, a Modelling Object is managed only in one single model.
- A Repository Object can be reused in one or more models of the same model type. A Modelling Object cannot be reused in other models, and it only can be administered in the model it was created.

The Object Repository can be found on the Objects tab, when you choose the Modelling option inside the ComVantage Modelling Tool. You can see an example containing some objects in Figure 59.
To use Repository object follow the next steps (see Figure 60):

1. Get to the Modelling option, inside ComVantage Modelling Tool.
2. Select Objects tab, and then select the Objects tree to expand the tree view of all objects stored on it.
3. Now select the object and drag it to the model view. Please note that you can drag objects to model types that implements that kind of object.

Also you can use a repository object when you copy (or press Ctrl + C keys) and paste (Ctrl + V) an object in the same model or another model type that uses the same kind of object. In this case a dialog (Figure 61) will appear asking if you want to reuse that object or create a new one (in this case, a new object named “Copy of object” will appear in the Object repository).
Please bear in mind that there are some objects that are not Repository Objects. In the following example (see Figure 62), objects from Causality model are modelled but there are not showed in the Objects Repository. In the next version of the prototype a list of repository object will be available.
3.2.2 Model type Guidelines

3.2.2.1 Model type independent
The Note object in this prototype is similar that the one provided in the OMI prototype: Some text or an image can be used to provide information for the human reader, and it is available in each model type.

3.2.2.2 Process model
The Process model describes a sequence of activities and decisions, allowing the modeller to design the operational-level process path to reach a certain goal. This is the core model type, currently present both in the OMI prototype and the ComVantage Modelling Tool, but in slightly different versions (particularly regarding how it links to requirements for various types of resources and how it represents the resource repository).

The main concepts are:
The Process start concept indicates the beginning of the process model. This concept can specify the “trigger” and the “result”.

![Process start](image)

**Figure 63: Process start concept**

![Process start general options](image)

**Figure 64: Process start general options**

The Activity concept describes a task to be executed. Some interesting attributes to bear in mind are the ones referred to “time”, “activity” cost, “waste” and “auditing requirements”.

© ComVantage Consortium – 2013
The “Outgoing relations” chapter in the Activity notebook allows assigning some relations between the activity and other objects/models. The possible references are:

1. Assign units: Allows linking one Role from an Organizational Structure model to this Activity, indicating that this object is performed by that Role. A hyperlink to the referenced Role appears in the graphical representation of this object.

2. Referenced process: Linking an Activity to another Process model means that the linked Process is a description of tasks performed by the Activity. If done, the graphical representation is changed and a hyperlink to the Process model notebook appears instead of the Activity’s name.
3. Event source: Connects this Activity to another Activity or Decision.
4. Uses hardware: Linked to a Hardware resource object means that this Activity uses a Hardware resource to perform its function. More than one Hardware resource concept can be assigned through this reference. A small gear picture appears inside the Activity representation if this reference attribute is used.

5. Mobile support: References this Activity to one or more Mobile support feature concepts. It is similar to the Mobile support relation, but this reference allows connecting this Activity to Mobile support features from other model types that uses this concept, for example: another Process model. A small Mobile icon will appear if this reference attribute is used.

6. Works on information: References this concept to an Information resource concept, and means that this Activity uses some source of information available.
The Hub concept allows splitting into two or more paths which can be executed at the same time. Every Splitting Hub should have another Hub to join all those paths, i.e. a Joining hub. A user can change between Splitting Hub and Joining Hub types, also the modelling direction can be changed to either horizontal or vertical.
The **Decision** concept separates the execution path into two or more ways, but only one path should be chosen. This object stores the “question”, and a user must decide the path to follow by answering that question.

![Decision Concept](image)

**Figure 74: Decision concept**

The **Process End** concept marks the end of the execution path, indicating that the process has been executed.

![Process End Concept](image)

**Figure 75: Question options for Decision concept**
The secondary/complementary concepts are:
The `Section` concept allows assigning activities into different groups as sections. This object can be resized, in order to contain a group of `Activities`.

The `Event aggregation` concept represents handled events and provides information about how an event should be handled. This resizable type of aggregation can contain concepts like `Activities`, `Decisions`, `Hubs` and other `Event aggregations`.

A user can change the “Handling type”; each one chosen type is represented by a different colour in the graphical representation. Also the “Probability” and “Severity” type can be chosen.

If the “Handling process” attribute reference is linked to another `Process model`, a Start-like element appears in the top left corner of the graphical representation. This can be uses as a hyperlink to open the linked `Process model`. Only one `Process model` can be referred at a time.
Figure 80: Event aggregation options

The Mobile support feature concept represents a feature which should be available for an Activity on a mobile device. The graphical representation can be changed and chosen between the “drawing” type, the “smart phone-like” type, and the “generic” type in three different colours.

Figure 81: Different graphical representation options for this concept

The Swimlane concept allows separating objects according to different aspects. As a container, it can contain the other types of concepts inside it. For example: different Swimlanes could refer to different kinds of abstract levels. The Swimlane orientation can be changed either horizontal or vertical.
The guidelines for modelling a process are similar to the ones described in the OMI prototype, so please refer to those. In the next example (Figure 60) a simple Process model diagram shows how Mobile support features are linked to Activities, using the Mobile support relation. Each Activity with this kind of relation is showed with a small mobile picture inside of it.
3.2.2.3 Product configuration model

The Product configuration model describes the structure of a product and its attributes. This product, modelled in an abstract form, has one or more features linked to it. These Customization features acts as leaves of a hierarchy, where each one describes a specific feature of the product.

The first main concept here is the Product, which is the main class inside this model type. Product represents a concrete product which is gained or produced in the supply chain. Only one Product should be modelled at a time. Any product, along with “cost” and “price” options, has the option to change the default graphical notation to a picture, to give it a friendlier look.

Figure 85: Example of Process model

Figure 86: Product default representation and example alternative
A Product object or Customization feature can refer to one or more Activities using the “Worked by” reference, and this means that those Activities work on this concept.

The Customization Feature concept defines a part of a product. It can be connected to a product or another Customization Feature, to create a hierarchy of features of the product. There are two different type ranges for this concept; each one changes the graphical representation of the object:

- **Type** (or general type): Refers what this Customization feature represents. Possible values are: “material”, “component”, “abstract”, “product configuration”. Each type changes the graphical notation, with the first letter of the type chosen.
- **Kano’s Type**: This type is based on Kano’s model. Possible values are: “Basic feature”, “Performance feature”, “Excitement feature” and “Secondary feature”. Each type is reflected in a different colour in the feature.

![Different Customization Features according to its Kano's type](image)

The *Customization feature* can also be shown as “activated”/“deactivated” by enabling/disabling the “Active” attribute. A deactivated *Customization feature* is shown in grey.

![Disabled Customization Feature](image)

Modelling a *Product Configuration model* is simple, although there are some basic guidelines to follow. Simply put a *Product*, one or more *Customization Features* and connect the *Product* with each feature using the Includes relation. In the following example, we connect a *Product* labelled as Shirt with four *Customization features*, where each one represents different features of that product (“Label”, “Material”, “Warranty” and “Eco-Friendly”). Each *Customization Feature* can have more subsequent features as well.
Please don’t forget to select the correct type when editing each Customization Feature, and select an appropriate picture which represents the Product if one is available.

![Diagram of T-shirt product configuration]

**Figure 93**: Example of Product Configuration, based on a T-shirt product

### 3.2.2.4 Voice of the customer model

The *Voice of the customer model* describes the actual market structure for a product. This model complements the *Product configuration model*, and describes market segmentation results as seen from previous market analysis made by the user. Each market segment is linked to one or more *Factors* which describe the needs, goals and so on.

The *Market segment* concept represents the abstract form of the market. It is the classifications of customers by one or more characteristics to group them as a *Market segment*. A user can change the graphical representation of this object to any picture. Also if there is any property in the “Share” attribute it will be shown at the top right corner as a percentage in the graphical representation.

![Market segment concepts with Share property]

**Figure 94**: Market segment concepts. Right representation shows the % of share market
The *Factor* concept describes some certain characteristics of a *Market segment*. Each characteristic is of a certain type: “Need”, “Skill”, “Goal”, “Knowledge”, “Other”. Each type is represented using a different colour in the graphical representation of this concept.

**Figure 95: Market segment Notebook options**

**Figure 96: Different representations of Factor concept**
To model a **Voice of the Customer model** simply place one or more **Market segments** and connect them using the **Distance** relation. Then create as many **Factor** objects as desired and connect the **Market segments** to them using the **Described by** relation. One **Market segment** can be connected to one or more **Factors**.

![Factor concept General options](image)

**Figure 97: Factor concept General options**

![Example of Voice of the Customer model](image)

**Figure 98: Example of Voice of the Customer model.**

### 3.2.2.5 Organizational structure model

The **Organizational structure model** describes the organizational structure of a particular business / company, and shows all the units, roles and performers this structure has. Also the user can create **Teams**
and assign any Performer to them. An Organizational Structure model should be used to provide the roles and performers assigned to activities of a Process model.

The Organizational Unit concept describes the hierarchy of the departments/units in a company. This hierarchy can be structured, connecting different Organizational units between them. An Organizational Unit has a type: “Standard”, “company”, “division”, “department”; as well as a “manager”, their “function” and the “number of positions”, “occupied positions” and “open positions”.

![Organizational Unit](image)

**Figure 99: Organizational unit concept**

![Organizational Unit General options](image)

**Figure 100: Organizational unit General options**

The Performer concept represents a performer/person in a working environment. A Performer can act in one or more roles, and belongs to one or more Organizational Units. The attribute “Further Education” is used to store all the seminars/courses of a Performer and their participation date. Also the user can define the “Hourly Wage” and the “Availability” of the Performer.
Figure 101: Performer concept

The Role concept represents a job description and has one or more Performers assigned.

Figure 103: Role concept

The Team concept is an aggregation of several performers, indicating that the contained performers work together in a team. It is a resizable object and it can also contain Role objects in it.
Modeling an Organizational Structure model is quite simple. Drag the concepts in the modeling area, and then use the Has Subunit relation to connect the Organizational Units. Also use the Has Position relation to connect Performers to Organizational Units and the Acts in Role relation to connect the Performers to Role concepts.

3.2.2.6 e3 Business model
An e3 Business model\(^\text{12}\) describes the abstract design of value exchanges for a supply network to achieve a certain goal, in most cases profit. The model type is domain independent and it is a complement to the supply chain view, focusing not on the sequence of value transfers, but the dependency between.

The Actor concept represents an abstract entity or a real company. The graphical representation is resizable, to ensure that it can contain Start stimulus, AND/OR, Value interfaces and Dependency path relations.

\(^{12}\) Based on the e3 value language, described at http://e3value.few.vu.nl/
Figure 106: Actor concept

The Market segment concept is a part of the market for a product, classified by customers in one or more characteristics. Like the Actor object, the graphical representation of this concept is resizable and can contain the same objects the Actor can.

Figure 107: Market segment concept

The Value Interface concept is an object included in an Actor or Market Segment and exchanges values with another Value Interface from a different Actor/Market segment. The orientation of this concept can be changed to meet the orientation needs of this model; also the graphical representation is resizable in its longest size to be able to add more Value transfer relations.

Figure 108: Representations of some Value interface orientations
The **Start Stimulus** interface concept indicates the origin of the value transfers, the stimulus that leads to the execution of transactions. It is contained in an **Actor or Market segment** and the number of occurrences needs to be defined here. In the next implementation this number of “occurrences” will be used in some functionalities regarding this model type.

The **And/Or** concept allows to split the path where occurrences flow. The concept’s type can be change between the “AND” type and the “OR” type. For the AND type, the modelling direction can be changed if the user needs to model in a different orientation.
Figure 112: AND/OR concepts

Figure 113: AND / OR options

Modelling an *E3 Business model* should follow some rules in order to obtain the maximum satisfaction:

- First of all please bear in mind that the main concepts to draw are *Actors* and *Market segment*, and they work together using the *Value transfer* relation class. Each *Actor* or segment needs a *Value interface*, placed in the border of the object, and then connect to the *Value interfaces* of other *Actors/Market segments* using the *Value transfer* relation as desired.

- Resize the *Actors* or segment that will contain the *Start stimulus* and the AND/OR concepts, plus the *Dependency path* relations. Note that only one *Start stimulus* should be added in the model.

- Since there is no “End” concept object, it is assumed that any *Dependency path* is ended in a *Value interface*.

- Lastly, bear in mind that the functionality regarding the occurrences is still in development, so the tool will not use the fraction attribute found in *Dependency Path* relations, or the valuations attributes showed in the *Value transfer*. 
3.2.2.7 Scope configuration model

The *Scope configuration model* describes the business scope for a certain product, where different abstract entities are connected. It is a complement of the *E3 Business model*, focused on the supply network and location.

The *Business entity group* concept represents a group of different business entities. The size of this object can be changed in other to contain *Business entity* objects inside of it.
The *Business entity* concept represents a company or a part of it, and can be inside a specific *Business entity group*. Some attributes to bear in mind are the ones referred to “Localization” of the entity. They will be used to display the entity on a map in future releases.

![Figure 116: Business entity concept](image)

The *Business role* concept describes a role with its capabilities. It can be fulfilled by one or more *Business entities*.

![Figure 118: Business role options](image)

For modelling in the *Scope configuration model*, drag the desired *Business entity groups* and add the necessary *Business entities* inside them. If required, place also *Role* concepts and connect them to the *Business entities* using the *Fulfils role* relation. Finally connect two or more *Business entities* using the *Relation* connection between them.
3.2.2.8 Thread model

The Thread model describes the design of a supply chain according to process categorization standards. It is based on a set of processes where the flow of material and/or information indicates the sequence of execution.

The Business entity concept represents a company, and provides an overview of the businesses and their processes involved in a supply chain. It is similar to the Swimlane concept from other model types. As a container object (container of other objects like Process category), it can be resized.
The *Process category* concept shows the scope and configuration of each involved *Business entity*. Also this concept is a container of *Process* objects. The “identifier” attribute is displayed at the top left, outside the rectangle.

Numerous metrics can be added to this concept through the provided tables: “Reliability”, “Responsiveness”, “Agility”, “Cost” and “Asset” metrics.
Also the representation can be changed from the original box-like graphic to a left or right arrow, and it can have a fixed size or be a resizable object (non-fixed size).

The Process concept represents a process necessary for the supply chain. The “identifier” attribute is displayed inside the graphical representation.
Like the Process category, numerous metrics can be added to this concept through the tables: “Reliability”, “Responsiveness”, “Agility”, “Cost” and “Asset” metrics. Also there is the “Described by” reference to bear in mind, which is included in the “Outgoing relations” tab. It connects this concept to a Process model, which describes the Activities performed when this Process is executed. A hyperlink to the referenced Process model appears instead of the Process name.
Also the graphical representation can be changed to a left arrow or right arrow. Additionally it can have a default size, or be a resizable object (useful in some situations where some sub-processes are needed, and this Process could contain those sub-processes).
The **Hub** concept allows splitting into two or more paths which can be executed at the same time. Every **Splitting Hub** should have another Hub to join all those paths, i.e. a **Joining hub**. A user can change between **Splitting Hub** and **Joining Hub** types, as well as the modelling direction (either “horizontal” or “vertical”). This concept is the same as used in the **Process model**.

---

The **Decision** concept separates the flow of information into two or more ways, but only one path should be chosen. It is the same concept as shown in the Process model.
Guidelines for the secure collaboration model

The **Start** concept is similar to other Start-like concepts seen in other model types. The main difference is that this **Start** element can be used in a **Thread model** more than once, depending on how many **Push Boundary** elements are represented in the model. This **Start** object symbolizes the beginning of a process chain or path in a **Thread model**.

---

The **End** concept indicates the end of a Path. In the Thread model this is also known as “critical path end”, where the path which it ends is the main path of the supply chain. Other paths deal typically with tasks not directly related to the creation of a product, like waste disposal. If it is no critical end, the graphical representation is changed to one that displays a cross in the graphical representation.

---

© ComVantage Consortium – 2013 108
The *Push Boundary* concept indicates a stop or an interruption of the flow in the supply chain. The graphical representation can be displayed as left side or right side.

![Figure 138: Push boundary concepts](image)

For modelling a *Thread model*, please follow these guidelines:

- Usually start modelling by placing a *Start* element and only one *End* element at the very first beginning. Later more *End* elements, *Push boundary* objects and *Start* elements can be added if it is needed.
- Take *Process* categories as the main concepts in this model. If there is a *Process* category that should be described in depth, unmark the “Default fixed size” attribute of that object and place as many *Process* as needed to describe it inside the *Process category*.
- It is advisable to group the *Process category* elements inside *Business entity* objects to indicate whose *Process category* they are.
- Finally use the *Flow* relation to join all the different *Process categories* and *Process* as desired, to create the “flow” of this model.
Figure 140: Example of Thread model
4 CONCLUSIONS AND OUTLOOK

4.1 Conclusion

The proposed Secure Collaboration model addresses a complete multitiered security approach which will be able to provide a trusted collaboration framework as well as protection for the Linked Data information published by the collaborating companies, helping to avoid threats and vulnerabilities.

In this deliverable, the whole security model architecture has been defined and all the components involved have been presented. That is, first of all, the authentication system has been presented. Then, the two secure access control models have been shown, which will be applied depending on the type of information that the user requests.

Apart from that, guidelines for modelling with the modelling prototypes resulted from task 3.4 have been presented in this document. Means of accessing these prototypes, together with means of joining the community space that hosts one of the prototypes (from the Open Models Initiative), are provided in the companion document of deliverable D3.4.1.

As a conclusion, all the different analysis performed in this deliverable have led to the definition of an enhanced multi-tiered access control framework which will be able to protect all data sets (RDF data, transient data and data in traditional formats) in the collaboration network, helping to ward off threats and eliminate vulnerabilities while proving compliance and maximizing the efficiency of the operations.

4.2 Outlook and open issues

Future research directions should focus towards more effective policy management frameworks for efficient View management that derive into optimized inferring processes and data view maintenance each time a RDF data set is changed.

For the remainder of the project, we will focus on intelligent graph management and policy organization so as to obtain as a result simpler view arrangements and the need for partial inferring processes, with the inherent performance and management advantages.

Moreover, future testing campaign will be carried out to provide a thorough evaluation with other SPARQL query engines, such as Virtuoso, Sesame or Jena.

With respect to the modelling support, the prototypes are evolving behind the modelling method specification evolution and the project requirements, thus their current coverage is partial and fragmented in two separate tools, also due to a) a separation of concerns, b) different granularity of change requests, c) impact on the developments from related work packages and d) rationale given by the dissemination and exploitation goals for the modelling components. However, the aim is to integrate lessons learned from the OMI prototype in the ComVantage modelling prototype, as they evolve towards the final version.
## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action</strong></td>
<td>An operation on a resource</td>
</tr>
<tr>
<td><strong>Android</strong></td>
<td>Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers.</td>
</tr>
<tr>
<td><strong>Assertion</strong></td>
<td>Package of information that supplies one or more statements made by a SAML authority.</td>
</tr>
<tr>
<td><strong>Attribute</strong></td>
<td>Characteristic of a subject, resource, action or environment that may be referenced in a predicate or target.</td>
</tr>
<tr>
<td><strong>AuthN</strong></td>
<td>Authentication</td>
</tr>
<tr>
<td><strong>AuthZ</strong></td>
<td>Authorization</td>
</tr>
<tr>
<td><strong>Broker</strong></td>
<td>Software which mediates between a client objects and a server (or requester or caller)</td>
</tr>
<tr>
<td><strong>Context handler</strong></td>
<td>The system entity that converts decision requests in the native request format to the XACML canonical form and converts authorization decisions in the XACML canonical form to the native response format</td>
</tr>
<tr>
<td><strong>CoT</strong></td>
<td>A Circle of Trust refers to the trusted network that should be established between collaborative enterprises that work together sharing their private information, so that companies can trust each other.</td>
</tr>
<tr>
<td><strong>DAS</strong></td>
<td>Domain Access Server</td>
</tr>
<tr>
<td><strong>DN</strong></td>
<td><em>Distinguished name</em>. Concept used in directories (such as LDAP) to uniquely represents an object.</td>
</tr>
<tr>
<td><strong>DTD</strong></td>
<td>Document Type Definition</td>
</tr>
<tr>
<td><strong>ECP</strong></td>
<td>Enhanced Client or Proxy. The name is historical. The ECP profile is an adaptation of the SAML profile used for Browser SSO with the parts that were designed around the limitations of a browser removed.</td>
</tr>
<tr>
<td><strong>Environment</strong></td>
<td>The set of attributes that are relevant to an authorization decision and are independent of a particular subject, resource or action.</td>
</tr>
<tr>
<td><strong>Hiawatha</strong></td>
<td>Exotic web server</td>
</tr>
<tr>
<td><strong>HTTP</strong></td>
<td>HyperText Transport Protocol</td>
</tr>
<tr>
<td><strong>IAM</strong></td>
<td>Identity and Access Management</td>
</tr>
<tr>
<td><strong>IdM</strong></td>
<td>Identity Management</td>
</tr>
<tr>
<td><strong>IdP</strong></td>
<td>An Identity Provider, also known as Identity Assertion Provider, is an authentication module which verifies a security token coming from a different security domain. The Identity Provider of the origin domain exchanges SAML information with the Service Provider (SP) of the target domain for exchanging authentication information between security domains.</td>
</tr>
<tr>
<td><strong>iOS</strong></td>
<td>Apple mobile platform OS.</td>
</tr>
<tr>
<td><strong>IRI</strong></td>
<td>Internationalized Resource Identifier</td>
</tr>
<tr>
<td><strong>JAX-RS</strong></td>
<td>Java API for RESTful Web Services</td>
</tr>
<tr>
<td><strong>JAX-WS</strong></td>
<td>Java API for XML Web Services</td>
</tr>
<tr>
<td><strong>OMI</strong></td>
<td>Open Model Initiative</td>
</tr>
<tr>
<td><strong>Ontology</strong></td>
<td>Branch of metaphysics that studies the nature of existence or being as such. In computer science and information science, ontology formally represents knowledge as a set of concepts organized in hierarchies within a domain, and the relationships between pairs of concepts. It can be used to model a domain and support reasoning about entities.</td>
</tr>
<tr>
<td><strong>PAOS</strong></td>
<td>Reverse SOAP</td>
</tr>
<tr>
<td><strong>POI</strong></td>
<td>The Point of Interaction represents the most important part of the Mobile IT Support model for the developer of a mobile app. Through this concept it is possible to define different interactions with the user by selecting values in the “Abstract UI type”. Please, refer to section 3.1.2.1.4.</td>
</tr>
<tr>
<td><strong>Policy</strong></td>
<td>A set of rules, an identifier for the rule-combining algorithm and (optionally) a set of obligations. May be a component of a policy set.</td>
</tr>
<tr>
<td><strong>Policy administration point (PAP)</strong></td>
<td>The system entity that creates a policy or policy set in XACML.</td>
</tr>
<tr>
<td><strong>Policy decision point (PDP)</strong></td>
<td>The system entity that evaluates applicable policy and renders an authorization decision in XACML.</td>
</tr>
<tr>
<td><strong>Policy enforcement point (PEP)</strong></td>
<td>The system entity that performs access control, by making decision requests and enforcing authorization decisions in XACML.</td>
</tr>
<tr>
<td><strong>Policy information point (PIP)</strong></td>
<td>The system entity that acts as a source of attribute values in XACML.</td>
</tr>
<tr>
<td><strong>Policy set</strong></td>
<td>A set of policies, other policy sets, a policy-combining algorithm and (optionally) a set of obligations. May be a component of another policy set.</td>
</tr>
<tr>
<td><strong>Predicate</strong></td>
<td>A statement about attributes whose truth can be evaluated.</td>
</tr>
<tr>
<td><strong>RBAC</strong></td>
<td>Role-Based Access Control, an approach to restricting system access to authorized users. Each role assigns a collection of permissions to users. RBAC assumes that, in most applications, permissions needed for an organization’s roles change slowly over time, but users may enter, leave, and change roles rapidly.</td>
</tr>
<tr>
<td><strong>RDF</strong></td>
<td>RDF (Resource Description Framework) is a W3C standard for describing resources in the Web. RDF identifies things using URIs (Uniform Resource Identifiers). It uses simple statements (triples) to describe things: Thing – Property – Value (Subject – Predicate – Object).</td>
</tr>
<tr>
<td><strong>RDP</strong></td>
<td>Remote Desktop Protocol</td>
</tr>
<tr>
<td><strong>Resource</strong></td>
<td>Data, service or system component.</td>
</tr>
<tr>
<td><strong>RESTful</strong></td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td><strong>SAML</strong></td>
<td>SAML is an XML-based open standard for exchanging authentication and authorization data between security domains, that is, between an identity provider and a service provider. SAML is a product of the OASIS Security Services Technical Committee.</td>
</tr>
<tr>
<td><strong>SP</strong></td>
<td>It is the software element integrated within an application server responsible of protecting the resource/application. An SP exchanges SAML information with an IdP to...</td>
</tr>
</tbody>
</table>
### Guidelines for the secure collaboration model

Determine if a user has authenticated and obtain information about the user. The information obtained from the Identity Provider may be used to make authorization decisions.

<table>
<thead>
<tr>
<th>SPARQL</th>
<th>SPARQL Protocol and RDF Query Language (self-referencing 😊)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>STS</td>
<td>Security Token Service. Web service that issues security tokens.</td>
</tr>
<tr>
<td>Subject</td>
<td>An actor, whose attributes may be referenced by a predicate.</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>TriG Syntax</td>
<td>TriG is a serialization format for RDF graphs. It is a plain text format for serializing Named Graphs and RDF Datasets.</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface. It is the space where interaction between humans and machines occurs. The goal of this interaction is effective operation and control of the machine on the user's end, and feedback from the machine, which aids the operator in making operational decisions.</td>
</tr>
<tr>
<td>URI</td>
<td>In computing, a uniform resource identifier (URI) is a string of characters used to identify a name or a resource on the Internet. Such identification enables interaction with representations of the resource over a network (typically the World Wide Web) using specific protocols. Schemes specifying a concrete syntax and associated protocols define each URI.</td>
</tr>
<tr>
<td>WS-*</td>
<td>Web Service Standards</td>
</tr>
<tr>
<td>WS-Security</td>
<td>Web Services Security</td>
</tr>
<tr>
<td>WS-Trust</td>
<td>WS-* specification hat provides extensions dealing with the issuing, renewing, and validating of security tokens.</td>
</tr>
<tr>
<td>XACML</td>
<td>Extensible Access Control Markup Language. The standard defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate authorization requests according to the rules defined in policies.</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language.</td>
</tr>
</tbody>
</table>

© ComVantage Consortium – 2013
6 REFERENCES

AllegroGraph, n.d. AllegroGraph RDFStore Web 3.0’s Database. [Online] Available at: www.franz.com/agraph/allegrograph/


Karagiannis, D. et al., 2012. D7.2.1 – Adaption of Secure Information Model Concept. s.l.:s.n.

Karagiannis, D. et al., 2012. D8.2.1 – Adaptation of secure information model concept. s.l.:s.n.

Lázaro, O. et al., 2012. D3.2.1_Secure Information Model Definition.

Notation3, 2013. [Online] Available at: http://www.w3.org/TeamSubmission/n3/
DISCLAIMER

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law.