SEVENTH FRAMEWORK PROGRAMME
Challenge 1
Information and Communication Technologies

Trusted Architecture for Securely Shared Services

Document type:	Deliverable
Title:	Software Documentation: System: Back Office Services
Work Package:	8
Deliverable Number:	8.2
Editor:	Dr. Ingo Dahn, University Koblenz-Landau
Dissemination Level:	Public
Preparation Date:	1 June 2009
Version:	1

Legal Notice
All information included in this document is subject to change without notice.
The Members of the TAS3 Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the TAS3 Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Information Society Technologies

The TAS3 Consortium

Nr	Participant name	Country	Participant short name	Participant role
1	K.U. Leuven	BE	KUL	Coordinator
2	Synergetics nv/sa	BE	SYN	Project Manager
3	University of Kent	UK	KENT	Partner
4	University of Karlsruhe	DE	KARL	Partner
5	Technische Universiteit Eindhoven	NL	TUE	Partner
6	CNR/ISTI	IT	CNR	Partner
7	University of Koblenz-Landau	DE	UNIKOLD	Partner
8	Vrije Universiteit Brussel	BE	VUB	Partner
9	University of Zaragoza	ES	UNIZAR	Partner
10	University of Nottingham	UK	NOT	Partner
11	SAP research	DE	SAP	Partner
12	Eifel	FR	EIF	Partner
13	Intalio	FR	INT	Partner
14	Risaris	IR	RIS	Partner
15	Kenteq	BE	KETQ	Partner
16	Oracle	UK	ORACLE	Partner
17	Custodix	BE	CUS	Partner
18	Medisoft	NL	MEDI	Partner

Contributors

	Name	**Organisation**
1	Dr. Ingo Dahn	University Koblenz-Landau
2	Philip Feuchtner	University Koblenz-Landau
3	Thomas Kirkham	University of Nottingham
4	Quentin Reul	VUB
5	Sascha Zimmermann	University Koblenz-Landau

Table of Contents

Table of Figures

Executive Summary

This document provides the description of the back office services These services neither store person related data nor serve the user directly. They provide ontologies and metadata, perform search and aggregation operations and transform data into specific formats. Task T8.3 contributes to this deliverable. The back office services are a component of the TAS3 Trusted Application Infrastructure but not of the core TAS3 Trust and Security Infrastructure. This document describes in particular

- The Generic Data Format used to store data in TAS3 repositories
- Services to transform data from the IMS ePortfolio format to the Generic Data Format and from the Generic Data Format to the Europass CV Data Format.
- Aggregation Service and Policy Aggregation
- A Request Logger Service to store information on requests issued and responses received by TAS3 web services for rapid prototyping and debugging purposes

The TAS3 Trusted Application Infrastructure is the application dependent part of the TAS3 infrastructure. Its purpose is to provide the services needed to guide the implementation of the pilots of WP9 in the fields of employability and eHealth. The TAS3 Trusted Application Infrastructure depends on the requirements collected in WP1, the architecture design provided by WP2 and the business process models developed in WP3. It allows the application independent services developed in WP4, WP5 and WP7 to be used in the WP9 pilots.

Within the TAS3 Trusted Application Infrastructure the back office components serve as parts of the Application Dependent Policy Enforcement Points of service providers and service requesters or as stand-alone components. This document describes

- the assumptions on which the development of back office services is based
- the services provided
- the high level architecture of the services
- installation and usage of the services
- limitations and known issues
- plans for the further development of the back office services in the next phase of the TAS3 project.

NOTE: The software components produced in WP 8 implement application specific adaptors that are required to use the application independent TAS3 infrastructure in the TAS3 pilots in eHealth and eEmployability. The overall architecture, semantically enriched executable business process models with an XForms user interface and the design of the core TAS3 services are the pre-requisites for WP8. The TAS3 architecture has been finalized only recently (see D2.1). While all our results are consistent with and usable for the current TAS3 architecture it cannot be a surprise that some alignments and refinements will be required. Moreover new service needs, for example the request for a service bus, are emerging from the architecture document, which still need to be detailed before they can be implemented and documented in any of the deliverables of WP8.

Technical Note:
All produced components (web services, libraries and clients) of Deliverables D8.1., D8.2. and D8.3. can be found in a binary version at this location:
http://citrix.uni-koblenz.de:9000/homepage/tas3/default.aspx

To access this page, please use the following login data:
Login: tas3
Password: z65rf5

Revision	Date	Author	Changes
0.1	17.10.2008	Ingo Dahn	Initial version
0.2	4.11.2008	Ingo Dahn	Draft for Internal review
0.3	16.12.2008	Ingo Dahn	VUB Contribution on DOGMA/MESS integrated
0.4	01.04.2009	Ingo Dahn	Revised according to reviewers' feedback, integrated University of Nottingham contribution on aggregating policies
0.5	10.04.2009	Ingo Dahn	Revised according to recommendations at Koblenz Application Cluster Workshop of April 9, 2009
0.6	29.04.2009	Ingo Dahn	Revised according to internal review
1.0	15.05.2009	Marc Santos	Adapting to TAS3 template

1 Introduction

1.1. Scope, Objectives and Context

The TAS3 project aims to develop a service oriented architecture for the secure handling of personal data. According to the architecture description in D2.1 [3], this architecture can be divided into

- the core trust and security related services to be developed by work packages 4, 5 and 7
- the application dependent policy enforcement points that connect clients and repositories with the TAS3 secure infrastructure which are developed by work package 8 and described in deliverables D8.1 [1] and D8.3 [2] respectively
- a number of generic auxiliary services that can be used by other services, called back office services

The latter kind of services – the back office services – is described in this document. According to the Description of Work of the project, the TAS3 trusted infrastructure to be provided by work package 8 is the basis for carrying out the pilots in work package 9. Semantically enriched models of the business process models for the pilots are the basis for the development of the infrastructure. The position of the services within the architecture is explained in Section 2 of Deliverable D8.1 [1].

In contrast with other projects, TAS3 goes for an infrastructure that can be used by European citizens lifelong, in particular in contexts of employability and healthcare as particular examples. Therefore the TAS3 infrastructure has to be developed in a domain independent way.

These general settings of the TAS3 project suggested the demand for a number of back office services which are described in this document. It is understood that the list of these back office services will grow as the requirements of the pilots and the available trust and security services pose new challenges.

It is emphasized that the main parts of the Deliverables of work package 8 are the implemented software solutions themselves which are described in the Deliverable documents. Work package 8 explores the potential of existing technologies to integrate the innovative components developed in the core trust and security work packages with security aware business processes developed in Work Package 3 in order to realize the overall TAS3 architecture developed in Work Package 2. The development of new technologies is not a primary concern of Work Package 8.

1.2. Structure of this Document

This document describes the services which are available from work package 8 by PM15. This document is structured according to the different kinds of services.

We start with a description of transformation services. The domain independent character of the TAS3 project faces work package 8 with the need to handle a variety of domain specific data formats described in various specifications. To approach this problem in a specification neutral way, as sketched in the project's Description of Work, a simple generic data format has been developed. This format is described before introducing the individual transformation service and their underlying technologies.

From a methodological point of view, having a generic data format that can handle eHealth as well as employability data offers the possibility to develop a single uniform lab setting for testing generic infrastructure components.

As described in the DOW, TAS3 is dedicated to support purposeful aggregation of data objects. It has been envisaged that these data objects have sticky policies attached which have been set by the data owner to control data access and usage. The available use cases show a wide variety of data aggregation services. These range from loose combinations of XML fragments over zipped archives to fully integrated pdf documents. While structurally quite different, all these aggregation services share the problem to construct the policies that are to be sticked to the data aggregation. This problem is discussed in Section 2.

Already early versions of the TAS3 architecture showed the need to log service requests for auditing purposes. Therefore a service has been developed to log service requests. This service is of key importance to track service requests in the TAS3 infrastructure in order to maintain the consistency of the infrastructure and for auditing the behaviour of the infrastructure to verify its correctness. This service, which is bound to be modified as the audit data formats are defined, is described in Section 3.

Annex 1 describes the technology used to link Java code, automatically generated as described in this Deliverable, into the services. Annex 2 gives a more detailed overview of the architecture of the request logger service mentioned above. In addition to the services developed in the project, VUB provides ontology and metadata services. These services allow capturing and editing of semantic information in ontologies. Another semantic service supports searching for optimal solutions in a semantically specified sense. Those services, which allow the use of semantic technologies in the work of WP8, are described in Annex 3.

Annex 4 has a research paper discussing the problem of aggregating policies which is of concern for the development of aggregation services.

1.3. Reader's Guide

The introduction to Sections 2.1 – 2.3 provide information about the status of development of the back office services. As these are ad hoc developments for the needs of the project they are bound to be extended and modified as those needs develop.

The theoretically minded reader may be most interested in the discussion of the Generic Data Format in Section 2.1.1 and in the considerations on aggregating policies in Section 2.2.9.

Users of the services should consult the sections on Installation and Usage. For developers the sections on Interfaces and on Technologies Used will be of main interest. They may also wish to read the Annexes in detail.

2. Available Back Office Services

In this Section we describe the back office services that have been developed or are under development until mid 2009. These services are needed to manipulate data. Due to the lack of detail of the descriptions of the trust and security services and of the intended use cases, in particular w.r.t. the trust and security specific data to be handled, generic services have been selected for implementation, leaving placeholders to insert those data when they become available without deprecating the work described in this document. These are in particular:

- Transformations between data formats. These are currently used to prepare input/output of repositories but could be applied as separate transformation service anywhere in the service orchestration
- Aggregation of complex data from simple data objects
- Request logger to log data communication for rapid prototyping and debugging purposes

We have included in Annex 3 a description of semantic ontology services which are provided to the project by VUB. It is expected that those services can be applied to describe semantic issues in the TAS3 infrastructure, in particular when data are to be re-used cross-domain or when best-match data selections are to be made. These descriptions have been moved to an Annex since they have not been developed within the project.

It is understood that this list of services is not comprehensive. More services will be needed when the data manipulation needs have been defined in greater detail. Also these services have to be extended to correctly handle policies that stick to the data.

The back office services are designed to be used as part of the Application Dependent Policy Enforcement Points (ADPEP) or connected to the Audit Bus. As such they are assumed to be protected by the ADPEP they are in. If they are to be used as stand-alone services, they will need to be connected to the TAS3 infrastructure through their own PEP.

2.1. Transformation Services

2.1.1. Introduction

The TAS3 project aims to support the lifelong re-use of personal data. During their lifetime European citizens produce and use their personal data in a broad variety of contexts – education, work, health care, social life and finance ... to name just a few. For many of those contexts specific formats have been developed to describe the particularly relevant data. From context to context these formats differ not only in the data set they can cover, but also, for data relevant for different context, in the way these data are encoded.

The TAS3 project is following a specification-neutral approach as explained in [10], and [12]. An essential feature of this approach is that TAS3 services will communicate with services from outside the project **respecting their technical standards and specifications**, in particular HR-XML for the employability sector and IMS LIP for the educational sector. Further specifications, for example from the eHealth sector will have to be considered. Even within the same sector several specifications and their application profiles may be used for different purposes.
Clearly, re-use of data, in particular in a cross-domain setting, requires data format transformations. Providing transformations between any two relevant formats would require rapidly increasing efforts to link a domain with any new format to the TAS3 infrastructure.

The only feasible alternative is to use a single core data format and to provide translations to/from each relevant format to this core data format. Each of the existing data formats have been designed to suit the needs of a particular domain. Hence none of them is capable of storing all the personal data that may occur in any domain.
Transformation services will be used in the TAS3 architecture in support of the ADPEP of repositories. They are necessary to meet the requirement of cross domain data re-use.

2.1.2. The Generic Data Format

Therefore the TAS3 repositories, need to store data in a universal format that is capable to take up data from all particular formats and to translate them into these formats to the extent to which the respective format is capable to express them. This is called the specification neutral approach. The specification neutral approach intends to store data from any relevant format so that they can be recreated without loss of essential information, i. e. from the Generic Data Formats the information provided in a source format can be reconstructed in that source format. This does not mean that, for a given source format S a source file, providing data for translation into the Generic Data Format and a target file in format S produced from these data, will be identical. Formatting, white spaces and generic comments may be changed or lost. However it is expected that all proper information re-appears in a semantically correct place.
When data from a format S are translated to the Generic Data Format and then to a different target format T, it is admitted that loss of essential information can occur if T does not have provisions to take up these data from S. For example a file in HR-XML created for employability purposes cannot include some information from an HL7 eHealth file. This is not considered a deficiency: If the community designing the target format T did not provide possibilities to store particular data, then it can be assumed that these particular data are irrelevant for the respective target domain.
We add that personal data may contain data in proprietary formats which cannot be translated, for example pdf files or images. These occur embedded into a personal context assigning a meaning to these data. Such data are not touched by transformation to the Generic Data Format. Only their metadata, which describe the meaning of the data, are translated
Data which are signed or encrypted to protect them from changes are handled the same way as these proprietary data and are not affected by the transformations. Only when a policy applicable to such a document allows their disaggregation, a service provider may compile subsets of those data into a new format which she may eventually sign and encrypt anew.
The important difference of the specification neutral approach from the futile attempt to create one universal language to be used by everyone is that this universal format is for project internal purposes only. In fact, the specification
neutral approach allows domain specifc services to use their established domain specific formats for communication. In addition, the Generic Data Format allows for services supporting domains where several specifications are in use and for cross-domain re-use of personal data. For the experimental work within TAS3 it offers the additional advantage that we can use it in one repository for all experimental settings, whether they are motivated by employability or eHealth.
XML is generally accepted as lingua franca of the web for data formats. The broad tool support available for handling, analyzing, validating, transforming and rendering XML data make it unrivalled for development purposes compared with other formats like CSV, text based formats, PDF or proprietary formats. We are well aware that proprietary data formats can be handled often more efficiently than XML based formats. It is foreseen that systems with proprietary internal formats can be used with the TAS3 infrastructure, in particular for applications in restricted domains where being generic is not a requirement.
The TAS3 project uses Fedora repositories (see http://www.fedora-commons.org/) as standard repositories for experimental purposes. Reasons for this choice are explained in Deliverable D8.1 [1]. From those advantages it is particularly valuable for the TAS3 development process, that Fedora natively supports storage of XML data, sticky data policies, distributed data storage, semantic relations between data, web access to data and data transformations.

Nevertheless, making the format also easy to store in widely used relational data bases is a major concern of the following design in order to ease the integration of legacy relational data bases into the TAS3 infrastructure as described in Deliverable D8.1 [1]. Therefore we start the design of the domain independent Generic data format with some general considerations.
As minimal general requirements for the intended format we need to express

1. Name-Value pairs
2. Data collections – this is important to re-use groups of data for efficiency and for ease of data maintenance
3. Relations between data and references to other data – note that this includes support for symmetric relations (for example "is married with") and possibly cyclic relations (for example "knows")

Beside these requirements to the expressive power of the language it is important that the format is easy to extend in order to adapt it to express information from other formats which may not yet be known. Simplicity of the format is considered as a convenient prerequisite for adaptability.
Exploring various approaches to data format specifications (including Europass, various CV formats, HR-XML, IMS LIP, IMS ePortfolio, HL7), we found the UN/CEFACT Core Components approach ([6], [7]) particularly promising because of its simplicity and flexibility which makes it quite generic and well prepared to adapt to future modifications. Due to its simplicity, such a format is easy to store in any kind of data base. Relations between data objects, which are central to this approach, are well-suited to be exploited by semantic tools based on RDF.
We are aware that the price to pay for these features is the need to handle large sets of data objects and complex procedures to collect all objects needed for a particular purpose. Nevertheless we believe that in the current experimental stage of the project, flexibility and extensibility of the format take precedence over efficiency considerations. As mentioned above, the TAS3 architecture does not mandate any specific data storage format, hence we feel free to choose one which supports research and development best.

The UN/CEFACT approach has essentially two constituents.

1. A set of data naming conventions based on the Oxford English Dictionary
2. A particular way to build data objects (called Business Information Entities)

While having sympathy for the Core Component naming conventions, we do not consider naming as a major concern of the TAS3 project in its current early stage. Moreover, the Core Components approach considers Business Information Entities in their Business Context while it is impossible for the TAS3 project to foresee all the business contexts in which personal information can be used. Therefore we are interested only in the context independent features of the Core Component approach.
An inspection of the strengths and weaknesses of the Core Components approach to structuring data revealed some feature that make it promising to adopt and adapt the Core Component approach for the design of the TAS3 project internal information models. Before describing the proposed information model and its binding to XML data structures we will sketch what we consider to be the most important features of the Core Component information model.
The following figure is adapted from [6].

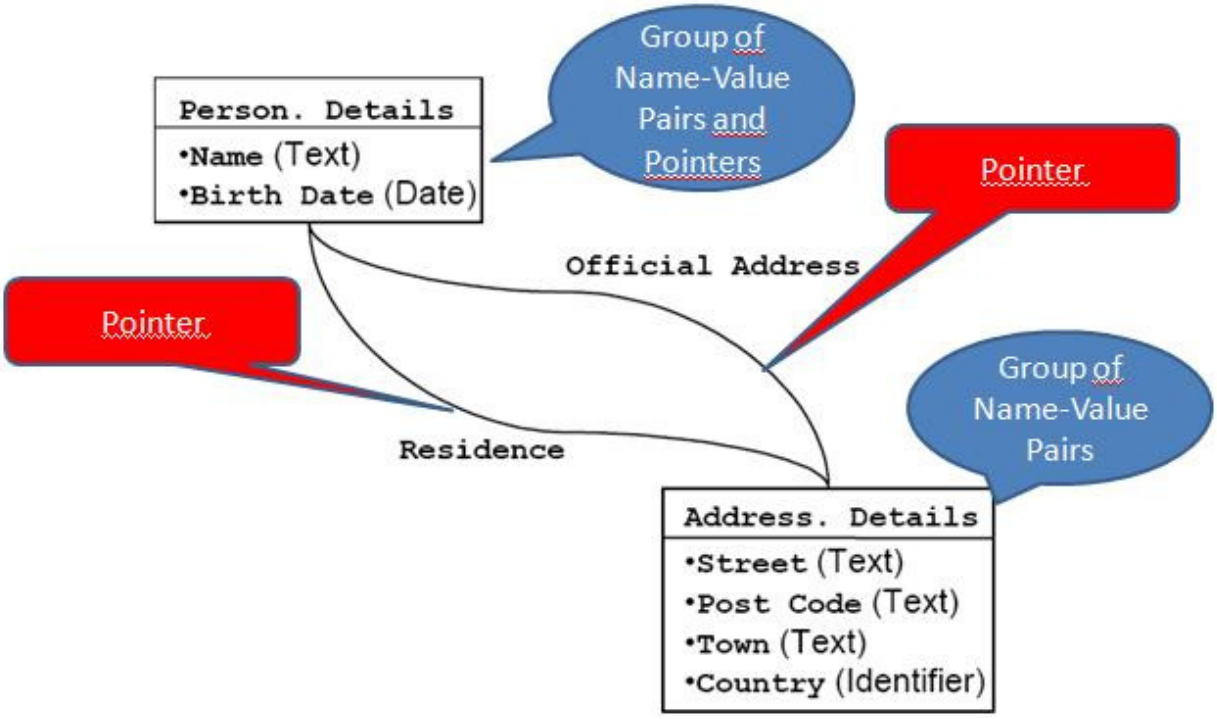


Figure 1: Core Components

Data objects in the Core Component approach (called Aggregate Core Components) consist of name-value pairs and pointers (called Association Core Components -ASCC) to other data objects only. These objects are simple, easy to extend by new fields and they are well suited to be represented in relational data base tables. The Core Components approach supports the re-use of data collections by allowing the same object to be targeted from different other objects. The specification [7] says on p. 19 "Because ASCCs represent hierarchical structures…" – this indicates that the Core Component approach is not intended to represent symmetric and cyclic relations. Therefore it needs to be modified to suit the needs of the TAS3 project.

When we generalize the Core Component ASCCs to be indeed labeled pointers, it is possible to support the implementation of arbitrary networks of binary relations. Relations with more than two arguments can be modelled with reification techniques, for example "A certifies that B has competency C" can be represented as a binary relation "A certifies D" where D is a competency possession data object with pointers to the competency C and to the individual B.

As mentioned above, disadvantage of the Core Component approach is that, due to the simplicity of the data structure, a large network of data objects[1] will be required to express data of any sizable complexity.

There are some issues with the Core Component approach to data structuring that need further clarification:

An element in an XML document can contain several subelements with the same tag. For example in the current HR-XML 3.0 draft the element *PersonName* can have an unlimited number of string values for subelements all named *GivenName*. Therefore it seems necessary to support re-use of the same field name within one data object.

Related to the aforementioned issue is the question whether the order of fields is relevant. The example of the *GivenName* of a person also suggests that the order of data fields matters in some cases. The Generic data format does not prescribe a particular order of fields. Therefore, if this order is relevant because it affects the semantics of the content, the

[1] Fortunately for the TAS3 project the Fedora repository has been designed with a particular concern to handle large sets of data objects.

data need either to be stored in fields with different names reflecting this semantics or a numeric parameter needs to be attached stating the position of this element.
More generally, parameterized field names are used when an element contains many closely related fields, in particular when the number of these fields is potentially large or cannot be determined in advance. Parameterized field names are distinguished in XML by using parameters from a predefined parameter set. They occur in two forms.
In the simple form, parameters are given as an attribute of an XML element, the tag of which carries the field name like in <city **lang=„de"**>Aachen</city> or <city **lang=„fr"**>Aix-la Chapelle</city>. In this example the lang attribute specifies the language in which the value is given. The possible values of the attribute are usually defined by a simple type in the respective specification. The more complex form, used for example in IMS specifications, ties the field name to a complex structure which points to an external vocabulary and to an item in this vocabulary:

```
        <partname>
                <typename>
                        <tysource>imsdefault</tysource>
                <tyvalue>Given</tyvalue>
        </typename>
        <text>My Given Name</text>
</partname>
```

>
This has the advantages that the vocabulary can be replaced without changing the core specification, that the vocabulary can be structured and/or multilingual.
The Core Components approach does not foresee the use of parameterized fields. However, in particular for the support of multilingual representations of data which is so important in a European context, we consider the use of parameters, in particular of a language parameter, as indispensible. Unfortunately, structured vocabulary based approach described above is incompatible with the previously made requirement to have simple, unstructured data objects only.
Therefore we are led to use the attribute based approach only. When needed, the structured approach and the use of external vocabularies can be simulated in the attribute based approach by applying restrictions to the structure of possible attribute values. For instance, the example above could be re-written as

```
<partname type="imsdefault/Given">
        <text>My Given Name</text>
</partname>
```

Having decided to group personal data into simple, re-usable, interrelated data objects, we need to decide how this grouping should be defined. We have adopted the following four guiding principles. These principles are based on semantics and on expected usage. We relate them to principles of Data Normalization below.

1. **Joint Access Principle:** Collect data into one object only if they share the same access rights. Observing this principle will allow data to be released object-by-object without further inspection of the object data. This principle requests that data should not be put into the same data object if they do not share the same access rights.

2. **Joint Concern Principle:** Collect data into one object only if they concern the same real or abstract items. For example data concerning father and mother of a person should not go collectively into one data object. In this way, each item is represented by a cloud of interrelated data objects which represents the facets that the item can present to the world. Also this principle serves to restrict the size of the data objects.

3. **Joint Usage Principle:** Put data into one object if they are mostly used together. This principle serves to restrict the number of data objects. In case of doubt it is recommended to keep as many data as permitted by the first two principles in one

object and leave it to the application layer to remove the data that are not needed for a particular use case. Should this turn out to be inconvenient, such an object may later be replaced by another object having pointers to smaller objects which contain the data needed for the individual use cases.

Joint Semantics Principle: Each data object, each data object field and each relation should have a unique semantics. This implies that data fields which are distinguished by the use of attributes must have the same semantics; in particular attributes must not be used to differentiate between different semantics of the field. The use of the language attribute is considered to be in-line with this requirement. Semantic information is provided by reference into the TAS3 Subjectory at https://wiki.tas3.eu/bin/view/Subjectory.

When implemented in relational data bases, these principles are to be combined with principles of Data Normalization. The objectives of Data Normalization are avoidance of redundancy, ease of maintenance and efficiency of access. While avoidance of redundancy and ease of maintenance are equally valid in the TAS3 context, efficiency of access has to follow different rules in TAS3 since TAS3 has to do with distributed data and with protection of access.

When a new data format is to be handled by a cross-domain repository, the Generic Data Format needs to be extended in a backwards compatible way. To achieve this, a first step is to locate the nodes of the format that keep the information to be stored. Next, to satisfy the Joint Semantics Principle, we have to locate for each data node the context[2] that determines its semantics.

From this semantics it should be clear which real or abstract item it concerns. If there is already an object for such an item, we shall consider, according to the Joint Concern Principle, to store the respective information in a field of this object. Otherwise we have to define a new object for this kind of item. In the further case, i.e. if an appropriate data object is already available, we ask whether the new information is, if available, used together with the other information in this object (Joint Usage Principle). If not, or if we cannot expect it to share the same access rights (Joint Access Principle) with the fields already present, we again introduce a new data object and add to the existing object a relation that relates it to the new object. In this way we can find that object when needed and we can define specific policies for the new object.

In this way we can extend the scope of the Generic Data Format without deprecating existing data stores or transformations.

A straightforward implementation of the Generic Data Format in a relational data base will create a separate table for each type of objects with columns for each data field. Each object is then represented as a row in that table. When data are kept in multiple repositories targets of relations between those objects can point from a row in a table in one data base to a row in a table in a remote data base. A central table keeping all those relations is not only unrealistic, but would even restrict the possibilities for privacy protection: In some cases it is intentional that object B can be accessed from object A but not conversely, for example if A is the name of a person and B is an artistic alias name of that person.

Such a straightforward implementation would obey principles 1NF, 2NF and 3NF of Data Normalization but not the principle of the Boyce-Codd Normal Form: Related items will be grouped together in one data object and each data object gets a unique URI attribute which can serve as a key. Avoidance of data redundancy is achieved by the fine grained structure of the Generic Information Model. Whenever information is to be re-used, it can be put into different data objects which are referenced through relations. The Joint Concern Principle

[2] This step to determine the context which uniquely determines the semantics of a data field is also an essential pre-requisite for defining semantically correct transformations.

implies that only related data are grouped together in one table and hence enables the 3NF principle of avoiding irrelevant table columns. As mentioned above, following the Boyce-Codd principle of separating relations from data objects is not always wanted in the TAS3 context. Nevertheless, since relations are clearly distinguished from ordinary data fields by the presence of a target attribute, such a separation can be easily introduced. If a data base does this for efficiency reasons, it needs to make sure that the interface it exhibits to the outside world does not allow illegal aggregation of data through relation tracing.

The Generic Information Model[3] can be inspected at http://iwm.uni-koblenz.de:8080/orbeon/tasData/. At this URL a web based tool for the further development of the information model is maintained.

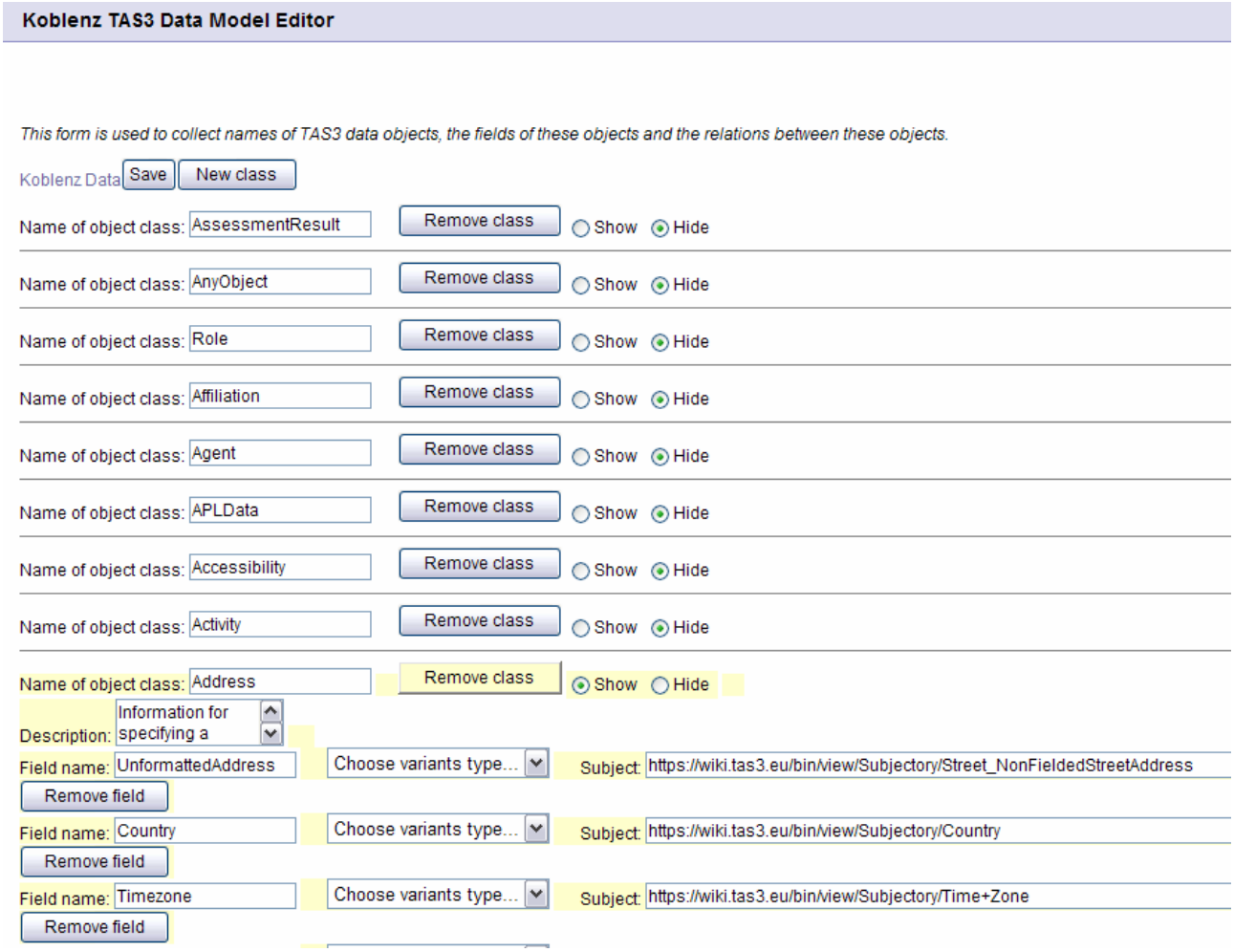


Figure 2: Screenshot of the Data Model Editor

The Generic Information Model is augmented by an XML binding. The XML schemas defining this binding are automatically generated from an XML file produced by the aforementioned web based tool.

Relations between data objects are the key way to model complex data systems. In order to enable this, the TAS3 XML binding requests that each data object gets a unique identifier. To support the exploration of collections of data objects, each such collection is augmented with an *anchor* element pointing to an entry point, usually an object of type *identity*.

[3] The Generic Information Model defines the way information is structured. The Generic Data Format defines an XML binding of this structure, i.e. appropriate XML tags and data types are defined by XML Schemas.

Particular domain specific data formats may contain data which make sense only in this particular domain where this format is used. In such cases these data will always have to be delivered in this format so that it is not mandatory to convert them to a generic format. Therefore the Generic Data Format contains an exceptional data type named *AnyObject* which can contain structured XML data, augmented with metadata according to the Unqualified Dublin Core metadata standard. *AnyObject* can also be used to store data for which a conversion to the generic data format is not available yet.

The personal data to be handled in TAS3 are frequently augmented with data in legacy formats, for example images, MS Word files, pdf certificates etc. These data are provided in a special data object type named *Attachment*. The *Attachment* object has a data field *Location* which points to the place where the file respectively the files are stored.

Besides the proper data content data objects may optionally contain sticky policies that govern how the data in these objects are to be used. These policies may be set by the data owner and they can – under certain conditions (see below) – be augmented or modified by institutions and users who have received such objects for their own processing needs.

We collect policies sticked to data objects in a single tag <policies> which is to be optionally added as the last child of the name tag of the respective object.

TAS3 supports the use of a variety of policy languages (for example XACML and PERMIS).. However it can be assumed that all policies will be encoded in XML. Since policies for a particular object may be collected from various sources, one object may have attached sticky policies in several languages. Also when an object is obtained as an aggregation of existing objects with their particular sticky policies, it may be an option to pile up all policies of the component object in order to define a policy for the aggregated object (see Section 3). Thus we arrive at the conclusion that each data object may have attached a variety of policies expressed in a variety of policy languages.

In order to decide access based on such a mix of policies, a Master PDP will analyze this mix and delegate the policies to language specific PDPs. To support this we group policies below the <policies> tag within tags <policy> and request that type and version of the respective policy group are given with a mandatory attribute *type* and an optional attribute *version*.

2.1.3. Functions Realized

Transformation services translate data from one format into the Generic data format or conversely. Currently the following transformations are available.

1. From IMS ePortfolio format to the Generic data format
2. From Generic data format to Europass Data Format (see [21])

The main problem to be handled by the transformations into the Generic data format is the decomposition of complex data structures into sets of simple data objects. For example an address is described in IMS ePortfolio as follows:

```
<address>
        <typename>
            <tysource sourcetype="imsdefault"/>
            <tyvalue>Private</tyvalue>
        </typename>
```

```
<contentype>
<referential>
<indexid>address_01</indexid>
</referential>
</contentype>
<street>
<nonfieldedstreetaddress>Lindelaan 4a</nonfieldedstreetaddress>
<streetnumber>4a</streetnumber>
<streetname>Lindelaan</streetname>
</street>
<city>Rommeldam</city>
<country>Nederland</country>
<postcode>7522 MD</postcode>
</address>
```


Note that this not only defines the address but also classifies the address as private. The transformation service has to extract the data, build the respective TAS3 data objects, fill the appropriate fields with data and link the data objects. In this example two linked objects are created: *ContactMethod* and *Address*:


```
<tas3:ContactMethod uri="7B2E0372805E2E43A0D879F867EF9091">
<Contexts>Private</Contexts>
<Address target="BDFB2CF46FFDC1459832431A26EED338" />
</tas3:ContactMethod>
<tas3:Address uri="BDFB2CF46FFDC1459832431A26EED338">
<UnformattedAddress>Lindelaan 4a</UnformattedAddress>
<Country>Nederland</Country>
<StreetName>Lindelaan</StreetName>
<StreetNumber>4a</StreetNumber>
<PostCode>7522 MD</PostCode>
<City>Rommeldam</City>
</tas3:Address>
```

For the inverse direction – transforming data from the Generic Data Format into another format – the appropriate structures must be built from simple data objects. In this example the value of the *tyvalue* element below the *address* element cannot be determined from the TAS3 *Address* object without following the link from the *ContactMethod* object. This is achieved by starting the transformation from the *ContactMethod* object and traversing the relevant link to collect further data. Similarly the total ePortfolio or Europass CV data set is built from a set of TAS3 objects by recursively following relevant links starting from an *Identity* element to which an *Anchor* points.

Handling of policies is not implemented yet. It is planned that for the transformation from any format to generic data objects to attach any policy sticked to the incoming data to each of the generated data objects. For the converse a default may be to collect all policies that have been sticked to data objects joining them into a new policy for the new object in a specific format (see below for alternatives).

2.1.4. Installation Guidelines

Software Prerequisites

Java Version 1.6 or later

Installation Instructions
No installation required

2.1.5. License Information

The software is released under BSD[4] License.

2.1.6. Usage

The program provides a standalone and a library version. The standalone version can be found in the source package and is called tas mapping fat jar.

Command line call

The standalone version accepts the following command line options:

-i <input filename or input directory>

-o <output filename>

-h a short help

-v <true/false> if set true, messages are displayed

It is called by:

java -jar tas_mapping_fat.jar -i portfolio.zip -o \

tas_mapping_result.xml

Library call

The class `Run_Me.java` *in* `de.uniko.iwm.tas3.tas_mapping` applies the mapping.

[4] http://www.opensource.org/licenses/bsd-license.php

It contains a method

```
public File find_and_map(File source_directory)
```

It takes a file, the source directory containing the unzipped e-portfolio package, applies the mapping and returns the resulting file.

2.1.7. Technologies Used

The mappings of data structures underlying the implementation of the transformation services are developed graphically using the software Altova MapForce. The following screenshot shows the mapping from an IMS ePortfolio to the TAS3 ContactMethod and Address object.

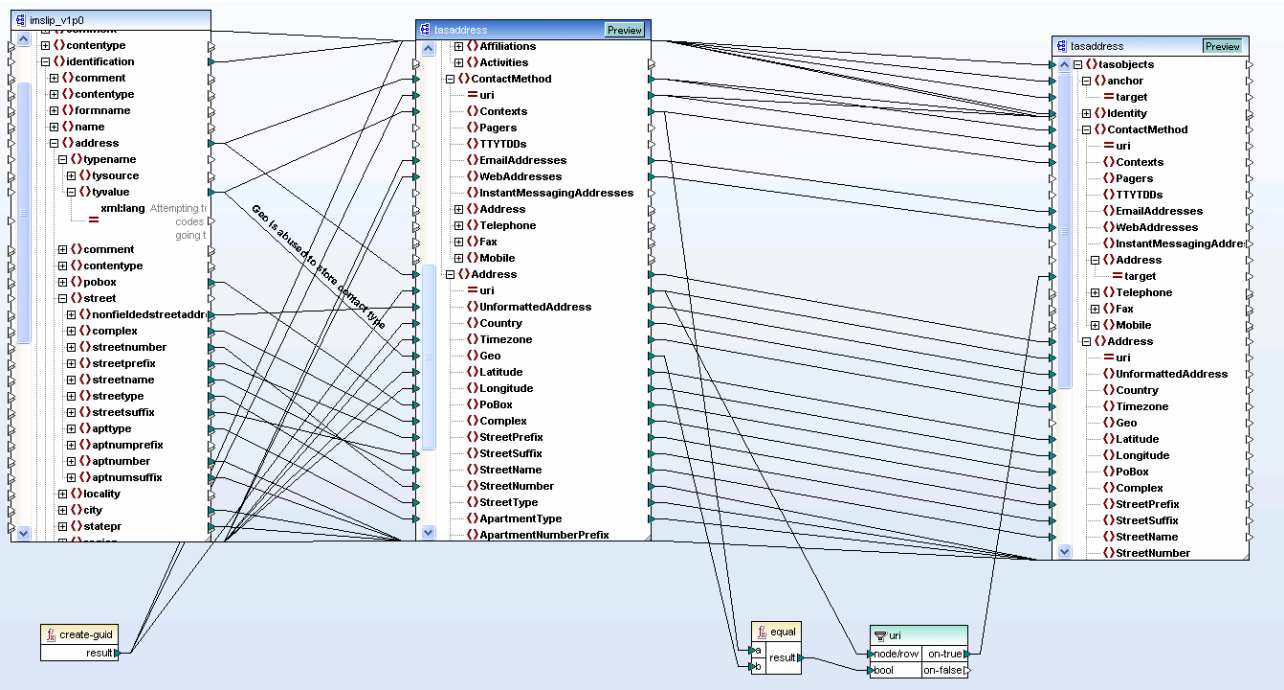


Figure 3: Screenshot of the Generic Data Format mappings

The mapping is constructed in two stages. In the first stage data fields from the source are mapped to appropriate fields in appropriate TAS3 target objects. In cases more complex than the one shown in the screenshot, the selection of the target field may depend on the context structure in which the source field occurs. In this first stage also URIs are created for all TAS3 objects produced.

In the second stage (the right part of the screenshot shown) the links between the objects are set. When this is complete, Altova Mapforce generates Java Code that implements the transformation defined by this mapping.

To build the mapping from IMS ePortfolio to the Generic data format, for each target object a mapping is constructed. This mapping usually does not only build the target TAS3 object, it also builds related objects (like the *ContactMethod* object in the example above related to the *Address* object); in particular it creates an *Identity* object from where all other objects can be reached. In a final step the generated transformations are integrated into the Java function which realizes the integrated transformation. The technology used for this integration is described in Annex 2. The resulting transformation

1. Unzips the source package IMS ePortfolio zip file

2. Analyzes the package manifest file to determine which transformation needs to be applied to which file in the package

3. Applies the appropriate transformations to the individual files

4. Merges the main generated *Identity* objects into one,

5. Collects all other generated objects,

6. Augment these objects with the policies that had been attached to the incoming portfolio,

7. Sets the additional, non-structural relations that are explicitly given in the relations part of the source portfolio

8. Adds all accompanying files in proprietary formats and

9. Packages the generated target XML file with those files into a new package for transmission.

2.1.8. Tests Done

The Dutch standard ePortfolio provides a variety of independent XML files with partially overlapping data types and with files in proprietary formats (for example images). The partial transformations have been tested using these files. The complete transformation has been tested using the complete portfolio. The integration of the transformation into the Fedora ADPEP has been tested with the client described in Deliverable 8.1.

2.1.9. Adapting the Transformations to Further Needs

While the form of the Generic data format, which follows the Core Components approach, is likely to stay, the precise definitions of the data objects may change during the further course of the project. In particular the release of the HR-XML specification 3.0 may give reason to align the TAS3 definitions with the HR-XML definitions. Since the HR-XML specification follows a similar Core Components based approach, it should not be a major problem to either adapt the transformations to/from the Generic Data Format to the data structures used by HR-XML or to develop only transformations of the current Generic Data Format to/from any aligned version of it.

Whenever a new data format is to be handled, the Generic data format needs to be extended for storing new types of information. That will leave intact all systems working with the previous version of the Generic Data Format. Then the respective transformations from/to the new data format to/from the Generic Data Format must be provided. Once this is completed, re-use of uploaded data in other supported formats is possible without further efforts.

Any change to the current Generic Data Format will be done in the web based editor for the Generic Data Model Editor which has been developed within Workpackage 8. These changes will be reflected in the XML Schema set generated from the output of the Data Model Editor. Altova MapForce will detect any changes that have been made to the schemas and will highlight those parts of the mappings which need to be reconsidered. After a manual correction of the graphical representations of the mappings, Altova MapForce generates new Java Code. Then the code providing the integrated mapping needs to be changed as documented in Annex 1.

2.1.10. Limitations and Known Issues

The current transformation from IMS ePortfolio format to the Generic Data Format handles all data that occur in the Dutch standard ePortfolio. This standard ePortfolio covers all parts of the IMS ePortfolio specification and some extensions specific for the Dutch labour market, but does not use all fields of the IMS specification. Nevertheless it is straightforward to extend the transformation to cover any missing field.

2.2. Aggregation Service

2.2.1. Introduction

Personal profiles, as defined in the Description of Work, are collections of personal data made up for a particular purpose. Such profiles are built up in TAS3 in two steps.

1. Determine from the repositories which data objects are relevant for particular purposes

2. Format those data in a form suitable for the particular purpose.

3. Assign policies to the aggregated object

The second function is similar to the translation from the Generic data format to a specific format. The Aggregation Service allows an integration of both steps, supporting the first step by the implementation of declaratively specified collections.

Within TAS3 the Aggregation Service will be used in support of the Service Responder ADPEP for repository services. There it serves to realize the domain independent specification neutral approach of TAS3.

2.2.2. Functions Realized

In detail, the knowledge engineer needs to formally describe the data needs for a particular purpose, referencing the Generic data format. For example the following could be declared:

- To get an address for a CV start from the anchor element and follow the relations *ContactMethods* to find a contact method and the *Address* relation to find an address for that contact method. If that is possible for a *ContactMethod* object with *Context* value "*private*" use *street, city, zip code* as number from the corresponding address, else try to find in a similar way address data for a *ContactMethod* object with *Context* value "*work*".

This could have been split up into two declarations – one defining what are acceptable contact methods and another one defining which fields of the related address objects are relevant. The aggregation service processes such declarations to determine which data objects are needed to build a personal profile. In a formalized way, the declaration above looks as follows.

```
get_address(Address) :- anchor(Id),relation('ContactMethods',Id,C),

        relation('Address',C,Address).

get_cv_address(Address) :- get_address(Address),
fieldValue(Address,'ContextType',_,'private'),!.
```

get_cv_address(Address) :- get_address(Address), fieldValue(Address,'ContextType',_,'work').

Currently realized functions include get_address and get_name to determine name and address(es) of a profile owner. Extending the aggregation service by new functions is extremely easy since it requires only writing declarations similar to those above.

In order to define rules for data collection the following primitive relations are made available:

- anchor(Id). This defines the Ids of concerned Identity objects
- fieldValue(ObjectId, FieldName, ParameterList, Value). This provides values of data fields.
- type(TypeName, ObjectId). This can be used to determine the type of an object or to find all objects of a given type.

Note that in these predicates all arguments can be used as input or output parameters as well, depending on whether they are instantiated at run time or not.

In order to **format** the collected data in a particular way, templates are used. These templates are XML files which have calls to the aggregation service inserted as special attributes. When the templates are parsed, these calls are executed to collect the required data. Other calls to the aggregation service let it insert the data found into the template.

For example, to insert address data the template contains

```
<address tasGet = "get_cv_address(address)">

  <addressLine tasGet = "get_street(address,street)">?- street</addressLine>

  <municipality tasGet="get_city(address,city)">?- city</municipality>

  <postalCode tasGet = "get_zip(address,zip)">?- zip</postalCode>

  <country tasGet = "get_country(address,country)">

    <label>?- country</label>

  </country>

</address>
```

In case the aggregation service delivers more than one solution, they are inserted all. For example, in the case above, if *get_cv_address* would yield more than one identifier for convenient *address* objects the xml document produced would contain more than one address element.

2.2.3. Installation Guidelines

Software Prerequisites

The Eclipse Prolog compile Version 6.0 of the Eclipse Constraint Logic Programming System is required. It can be downloaded from http://www.eclipse-clp.org for MS Windows, Unix/Linux and MacOS. It is released under Cisco-style Mozilla Public License Version 1.1.

The aggregator software can be ported to other Prolog systems by adapting the representation of strings and the calls to the *xml* library to the requirements of the target system. Since the same *xml* library is also available for B-Prolog, SICStus Prolog Version 3.11 or later and Quintus Prolog Release 3.5 no adaptation of the *xml* calls are necessary for porting to those systems.

Installation Instructions

The package contains the following files.

- aggregatorV1p0.pl – the source file to be compiled into Eclipse Prolog
- aggregationDefinition.pl – this file contains the definitions of the aggregation rules. This is executable Prolog code as well
- CVTemplate.xml – a template for test purposes to generate a Europass CV file from a file in Generic Data Format
- tasPiet.xml – a sample input file for test purposes. It contains the data of the Dutch Standard ePortfolio in Generic Data Format.
- A Europass CV file generated from the aforementioned input file and template

Within Eclipse Prolog call *?- compile('aggregatorV1p0.pl').*

The installation can be tested by calling *testInstallation*.

2.2.4. License Information

The software is released under BSD License.

2.2.5. Usage

Currently the aggregation program can be called from within an Eclipse Prolog interpreter. The file to be compiled is *aggregator.pl*. Eclipse Prolog offers a possibility to translate the Prolog source code into a Java module for ease of integration into a Java based repository ADPEP.

?- get_tasData(Filename) imports the data from an XML file which must be in the Generic Data Format,

?- fill_template(Infile,Outfile) parses a template from file *Infile*, calls the aggregation service to collect the required data during the parsing and writes the template filled with data to the file named *Outfile*.

The Aggregation Service can be modified to support other output format and other aggregation principles by modifying the *CVTemplate.xml* template file and the *aggregationDefinition.pl* file

2.2.6. Technologies Used for Implementing an Aggregation Service

Unlike many other data formats, the Generic Data Format is not based on deeply structured objects but on large sets of simple interrelated objects. Therefore XPATH as a basic technology to select data is hardly applicable. As an alternative we use Prolog technologies which follow a declarative programming paradigm. The particular application of the aggregation service, avoiding deep structures, provides a case which is known to be very efficiently handled by Prolog. Prolog is also used inside the semantic technology services used in TAS3, offering a further point of integration.

2.2.7. Tests Done

For test purposes the Dutch standard ePortfolio in the Generic Data Format, as produced by the Transformation Service as described above, has been used. It is provided, for testing the installation, as file tasPiet.xml.

2.2.8. Limitations and Known Issues

Though the aggregations service is functional it is not yet available as a web service. That may require use of the code with a different Prolog interpreter. For use in the future TAS3 infrastructure the program should be extended by a direct access to TAS3 enabled repositories of data objects.

The power of this aggregation service depends to a large extent upon the future modelling of the data needs of the individual use cases.

When data are aggregated, we shall have to consider the case where those data have sticky policies attached.

2.2.9. Aggregating Policies

This section presents approaches to policy aggregation in a user centric employability demonstrator. It reflects the current status of the discussion. Its results are not implemented yet. A more detailed explanation will be given in [16] which has been submitted to the 1st International Workshop on Business Security BPS09.
We suggest a two stream approach of policy based aggregation. The first of these involves an automated mathematical formulation of a new policy for the aggregated object using the sticky policies of the composite data objects. The other approach takes into account the context of the aggregation and uses supporting services such as an Aggregation Service (with associated PDP) to overwrite the policy rules and use a specially selected meta-policy for the new aggregated object.
The rapid development of service computing over resent years has seen the use of policy specifications become more distributed. In distributed service orientated computing environments the policy management environment has to be clearly defined and fine grained [17]. This refinement has led to the development of service infrastructures to support policy management in distributed systems and a whole range of associated policy standards [11], [20], [14], [8].
The support for policy based access to distributed applications and services is well supported in these frameworks and standards. However this has led to the creation of web service frameworks that can be seen to be bias towards the service provider perspective. The TAS3 project aims to readdress this balance by looking to define a policy based security mechanism for user data as it passes through distributed service based frameworks.

Central to this user centric focus on security is the association of data objects with specific policies. A popular method of achieving this is by the use of sticky policies [18], [9].

Sticky policies will allow the user to specify contexts and conditions around the usage of the data object as it passes between services. The use of sticky policies in a trusted network therefore has the ability to ensure that the user's policy constraints are respected when the data is out of their domain of control.

Automated / Local Aggregation

In this model all the policies are aggregated along side the data. The process is logical following on from the basic aggregation concept and the policies are all respected and merged into an overall policy for the aggregated object.
The advantages of this approach are that the transformation can be achieved in an automated way by the aggregator that does not rely on negotiation or retrieval of data from other sources. The disadvantages stem from the fact that the large amount of policy statements are likely to severely restrict how the new data object can be used.

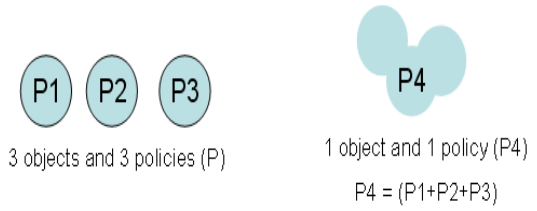


Figure 4: Automated Aggregation

This figure shows automated aggregation in a simple case. Each sticky policy could be designed so it can fit as part of an aggregation of similar object or could include a set of aggregation contexts to allow aggregation in specific circumstances.

The main driving force behind the aggregation is a mathematical formulation to check all policy statements are respected. Other aggregation equations may be introduced to better align policy definitions.

Trusted Aggregation

Problems aggregating sticky policies are linked to various factors ranging from problems integrating different policy formats, scalability of aggregating multiple policies to individual policy rules that could prevent certain aggregations. In these cases the sticky policy has to be over ruled.

To achieve this as in the case of a 'break the glass' scenario the TAS3 project uses policy contexts and supporting trusted infrastructure. It is envisages that when a data provider signs up to a TAS3 application they will agree to certain terms of use. These will include certain levels of service that they can expect from the application as well as guarantees over security of data. In addition the user will also agree to specific operations that the TAS3 application owner can conduct on their data.

Central to these operations will be the ability for the TAS3 infrastructure to overrule the sticky policy on the data objects. This process will be controlled by a set of specific circumstances in when this can be done and a clear explanation of the data that will be both aggregated and presented as a result. All aggregations will also be fed back to the user.

In order to standardise the methods of aggregation and the way they are presented in TAS3 the concept of meta-policies is introduced.

The meta-policies are be generated from the terms and conditions agreed to in the joining process by data providers. As discussed some of these terms will allow the owner of the TAS3 application certain access privileges on the data objects and will be set as the sticky policies are also associated with the data.

Meta-Policies

The sticky policies are stored with the data objects and the meta-policies are stored in the central TAS3 architecture. The meta-policy and data objects are linked via a shared identifier created when the user add the data objects to the application. It is in the interest of the TAS3 owner to make the meta-policy definition and use public to the data providers in the system the same way as terms of agreement are shared in everyday contracts.

The meta-policy specifies what can be done to the sticky policy associated with the data object. The meta-policy definition will specific to each TAS3 instance, this will ensure that meta-policies are application specific and can only be used in specific domains. Using meta-policies cross application / domain maybe possible using a higher entity but this is not yet defined and discussed as future work.

The application of meta-policies in the TAS3 employment demonstrator will be via services provided by the TAS3 application owner. However it is possible that as such an application increases in size this approach may become a bottleneck. In such cases it is envisages that by the use of trusted third parties and the use of trust ranking and reputation mechanisms the services that use and manage the meta-policies may be provided by third parties. The data provider could specify the level of reputation that the service provider must possess that handles their meta-policy data.

The ability to overrule the use of sticky polices by the use of meta-policies will allow distributed applications to scale better and support greater dynamicity. However this is dependent on the Meta-polices being clearly defined and linked specifically to the sticky policies in application centric way and ultimately under the control of the user.

2.3. Message Logger Service

2.3.1 Introduction

The purpose of the message logger service is to extract the necessary logging information from incoming and outgoing messages and to send:

i) the complete set of logging information to a secure audit log
ii) a summary of the logged information to the Audit Bus

when these components become available within the TAS^3 infrastructure. The logger service needs to be configurable so that the repository security administrator can determine what information should be logged (both complete and summary information). Furthermore the message logger should provide a service interface to allow the logger to be dynamically adapted to fit the current security context. For example, if an intrusion detection system suspects that the repository is being attacked then the logging might be switched to High level, whilst under normal operations it might be switched to Low level. The Message Logger Service is supposed to be closely connected to the (set of) service(s) the communication of which it is logging. It should enjoy the same access protection when integrated. The initial implementation of the logger service is described in this Section.

2.3.2. Functions Realized

In order to rapidly explore data flow and data access in a TAS^3 integration framework (a simulated TAS^3 environment where components delivered by other workpackages can be gradually integrated), early in the project the need was realized, to keep track of requests and responses sent over the infrastructure. The functional module *RequestLogger* provides a web service with a core functionality: Storing information about requests and responses sent by the requesting and responding applications. Using this service it is possible to save the following data for auditing purposes.

Attribute	Type	Description
Id	Integer	Primary key used to identify the dataset.
Name	String	The name of the person or application that sends the request / response.
Date	Date	The date and time of the request / response.
Ip	String	The IP-address of the requestor / responder.
Freetext	String	Free text field for additional information. This will be replaced by more complex structures after it has been specified which information has to be preserved for audit or maintenance purposes. According to the description of the audit bus in D2.1 [3] it can be expected that this will be at least a

		descriptor for the respective business context that led to the issuing of the respective call, in particular for the particular business task and the involved roles.

2.3.3. Interfaces

According to the functional requirements for the first release of the RequestLoggerService web service, the two functions saveRequestData and saveResponseData have been implemented. Due to upcoming additional requirements in the future, functional extensions will be implemented in later releases.

2.3.4. Technologies Used

Data describing the handling of requests and responses are stored in a MySQL database. MySQL has been chosen because of the available tool support to rapidly obtain a system usable for monitoring data flows. It is acknowledged that this is not sustainable for a tamper-proof auditing system. Storing the data is handled by using the Hibernate object/relational persistence and query service by implementing an object relational mapping (ORM). Through this, the two classes *Requestor* and *Responder* are mapped to their respective tables in the MySQL database.
The *RequestLoggerService* web service will be deployed on an Apache Tomcat web server and can be accessed through the Apache Axis2 web service engine installed on the web server.
Technical details about the implementation of the Request Logger service are provided in Annex 2.

2.3.5. Using the Service

The generation of the web service is done by using the Apache Ant build tool. Therefore the buildfile build.xml (in folder "RequestLogger/") is used to specify the ant tasks to be executed during the build process.

Executing the buildfile will compile the source files and create the directory structure (in folder "RequestLogger/build") where all required configuration files, compiled classes and used libraries will be stored. It also creates one single archive file *RequestLoggerService.aar* which is used to deploy the web service on the Apache Tomcat web server.

2.3.6. Limitations and Known Issues

The service needs to be augmented by a component determining which data are to be logged. A first set of messages worth logging is described in Section 6.3.1 of D2.1 [3]. These concern trust and security relevant events which will occur once the respective trust and security components have been integrated. Depending on the particular security-aware business process models and on the actual security situation, more information may be found worth logging.

The logged data are not secured yet. This becomes an issue when logs are not only kept for debugging purposes but for fulfilling contractual obligations. To achieve this, MySQL is to be replaced by a database which is protected so that only an appropriately restricted set of authorized data modifications is possible.

Inspection of the logged data is currently possible only by inspecting the tables in the data base. This needs to be augmented by web service access to the respective data base which needs to be strongly protected as well.

Also pushing summaries of the logged data to the audit bus is requested. This becomes possible with the implementation of the audit bus. It will make use of the library for creating standardized log summaries which is announced in D2.1 [3].

3. Recommendations for Further Development

Depending on the needs of the pilots, more transformation, search and aggregation services need to be implemented. Search and aggregation services need to be augmented with methods to attach policies to the data they return.

Subject to confirmation in a revised DOW, it is suggested to concentrate in the next period on the needs of the eHealth pilot as defined in D9.1. This requires

- Transformations between the relevant eHealth metadata format accompanying the attachments which hold the medical content and the Generic data format. The respective format has no deeply nested structures and is more close to the Generic Data Format than the IMS ePortfolio format for which the transformation was discussed in Section 2
- Preserving specific sticky policies during the transformation from the ehealth format to the Generic Data Format
- Providing a simple policy aggregation service that can be used to attach a policy to any selection of data objects that may collected based on restricted access rights.
- Refining the request logger service to keep the correct amount of data for auditing and maintenance purposes.

4. References

[1] Santos, Marc: TAS3 Deliverable D8.1 – Repository Services. 2009
[2] Kutscher, Michael. Deliverable D8.3 – Client / ServiceRequester ADPEP. 2009
[3] Kellomäki, Sampo. Deliverable D 2.1 - TAS3 Architecture. 2009.
[4] Abrams, M.D. Renewed Understanding of Access Control Policies. In Proceedings of 16th
National Computer Security Conference. 1993. Baltimore, Maryland, U.S.A.

[5] Allen, JH: The CERT Guide to System and Network Security Practices. Addison-Wesley. 2001

[6] 15.11.2003: Core Components Technical Specification – Part 8 of the ebXML Framework.

[7] 16.4.2007: Core Components Technical Specification - 2nd Public Review.

[8] Chadwick, D. W. and Otenko, A. The PERMIS X.509 Role Based Privilege Management Infrastructure. 7th ACM Symposium on Access Control Models and Technologies, 2002

[9] Casassa Mont Marco, Pearson Siani and Bramhall Pete. Towards Accountable Management of Identity and Privacy: Sticky Policies and Enforceable Tracing Services. DEXA Workshops, pages 377–382, 2003.

[10] Dahn, Ingo: Specification Neutral ePortfolios and Their Implementation With a Fedora Database: Proc. ePortfolio Conference, Maastricht 2007.

[11] Damianou, N., N. Dulay, E. Lupu, and M. Sloman. Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. The Language Specification Version 2.2. Research Report DoC 2000/1, Imperial College of Science Technology and Medicine, Department of Computing, London, 3 April, 2000.

[12] Dahn, Ingo; Vervenne, Luk (2007): Specification-neutral employability portfolios using digital repositories. In: Beishuizen, Jos; Carneiro, Roberto; Steffens, Karl (Hg.): Proc. of the Kaleidoscope-Taconet Conference. Aachen: Shaker , S. 54–63.

[13] Ferreira A, Cruz-Correia R, Antunes L, Farinha P, Oliveira-Palhares E, Chadwick DW, and Costa-Pereira A. How to break access control in a controlled manner. Proc IEEE CBMS 2006: pp. 847-54.

[14] Foster, I. and Kesselman, C. editors. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, 1999

[15] Iannella, R.: "Digital Rights Management (DRM) Architectures," D-Lib Magazine, vol. 7, no. 6, June 2001

[16] Kirkham Thomas, Chadwick David and Dahn Ingo: Aggregating Policies in User Centric Real-Time and Distributed Applications, Paper submitted to 1st International Workshop on Business Process Security (BPS09) – see also Annex 4

[17] Karjoth, G. and Hunter, M.: A Privacy Policy Model for Enterprises, IBM Research, Zurich - 15th IEEE Computer Foundations Workshop, June 2002

[18] Karjoth Gunther, Schunter Matthiasand Waidner Michael: Platform for Enterprise Privacy Practices: Privacy-Enabled Management of Customer Data. In Privacy Enhancing Technologies, pages 69–84, 2002.

[19] Rockafellar, R.T., and Wets, R.J.-B., "Scenarios and policy aggregation in Optimization under uncertainty", Mathematics of Operations Research v. 16 (1991), 119-147

[20] Sloman, M.S., Policy Driven Management for Distributed Systems. Journal of Network and Systems Management, 1994b. 2(4): p. 333-360.

[21] Europass: See http://www.europass-info.de/EN/start.asp

5. Annex 1: Adapting MapForce Generated Java Mappings

This annex describes the way to integrate new versions of data transformations generated by Altova MapForce. It is intended for future developers.

If a new mapping is created or a mapping is changed, the source-code needs to be adopted. The mapping is defined with Altova MapForce and exported as java package.

Following steps need to be carried out:

- Change package from `com.mapforce` to `com.`*`new_package`* to avoid name clashes. One may use eclipse to accomplish this task and may change *new_package* to a more appropriate name.
- Copy the resulting folder new_package/ to libs/com/ in the java project. The created folder com/altova should not be copied, because it is only be needed once.
- Find the file PortfolioConsole.java in libs/com/new_package/ and copy it into a temp-folder. Depending on the mapforce program setting this file can also be named MappingConsole.java.
- The program worker_fat.jar is part of the distribution. It needs to be applied to the file PortfolioConsole.java to create a usable mapping.
  - Cd to the temp-folder,
  - type
    java -jar worker_fat.jar -i MappingConsole.java -p \
    de.uniko.iwm.tas3.tas_mapping -c TasNewMappingMapping
  - copy the file TasNewMappingMapping.java into the folder sources/de/uniko/iwm/tas3/tasmapping/ of the java project.
- Now some minor changes must be applied to the source of sources/de/uniko/iwm/tas3/tasmapping/Run_Me.java in the java project.

  - Insert the constructor:

  Within this file find the comment

```
/*

* Important
*
* Codepoint register new mappings here !
*/
```

  ~line 203.

Insert

```
TasNewMappingMapping tasNewMapping = new TasNewMappingMapping();
```

  in the following line.

  - Insert the method

    In the same file find the comment

```
/*
* apply mappings
*/
```

  ~line 236.

  - insert

```
message("-> newMapping");
tasNewMapping.runMe(cleaner);
```

  in the following line.

- Save and rebuild tas_mapping_fat.jar.

6. Annex 2: Architecture of the Request Logger Service

This Annex describes technical details of the implementation of the Request Logger Service. It is provided as reference for future development.

6.1. Data Organization

6.1.1. Object Relational Mapping (ORM)

As mentioned before, an object relational mapping is implemented by using Hibernate, which maps the attributes of the two classes *Requestor.java* and *Responder.java* (package de.uniko.iwm.tas3.model) to the data fields in the respective tables of the MySQL database. These attributes are described in Table 1 (since both classes have the same attributes, they are described in only one table).

Attribute	Type	Description
Id	Integer	Primary key used to identify the dataset.
Name	String	The name of the person or application that sends the request / response.
Date	Date	The date and time of the request / response.
Ip	String	The IP-address of the requestor / responder.
Freetext	String	Free text field for additional information.

Table 1: Attributes of classes Requestor and Responder

In order to configure the ORM, the two Hibernate mapping files *requester.hbm.xml* and *responder.hbm.xml* (saved in the folder "RequestLogger/src") are needed. In these XML files, the attributes shown in Table 1 are assigned to the respective columns in the MySQL database. To complete the configuration of hibernate the file *hibernate.cfg.xml* (saved in the folder "RequestLogger/src") is needed. In this file, general settings are specified, such as the URL of the database, the connection driver used and the specified hibernate mapping files.

6.2. MySQL database

According to the above mentioned ORM, two tables in a MySQL database are required to store the data. The following list gives information about this database:

- **Server address:** mysqlhost.uni-koblenz.de
- **Database name:** tasss
- ***_data* and *res_data*:** Tables for storing attributes of the classes Requester and Responder.

Both of the tables req_data and res_data contain columns to save the data about requests and responses. The names of these columns (and the names of the tables) are the same as configured in the hibernate mapping files.

6.3. Program logic

The functionality is separated into two classes implemented in the following files:

- HibernateUtil.java in package de.uniko.iwm.tas3.util and
- RequestLoggerService.java in package de.uniko.iwm.tas3.services.

6.3.1. Class HibernateUtil

The class *HibernateUtil* contains basic functionality to initialize the connection to the MySQL database and to create the *SessionFactory* object which is used to open a new connection session required to access the database and to store data.

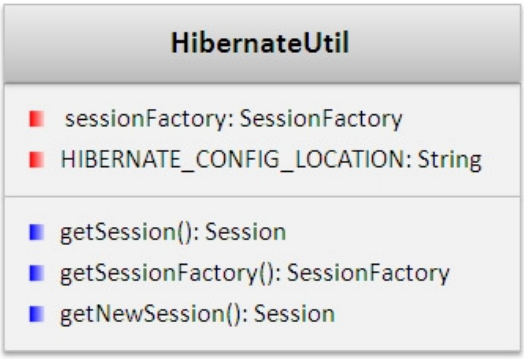


Figure 5: HibernateUtil class diagram

Attributes:

- **sessionFactory: SessionFactory** – The global SessionFactory object.
- **HIBERNATE_CONFIG_LOCATION: String** – The filename of the configuration file hibernate.cfg.xml.

Methods:

- **getSession(): Session** – Returns the current session opened by the SessionFactory.
- **getSessionFactory(): SessionFactory** – Returns the sessionFactory object.
- **getNewSession(): Session** – Opens a new session and returns the session object.

The class also contains a static initializer, which is automatically called when the class is loaded, that sets up the configuration of the hibernate configuration file and creates the *sessionFactory* object.

6.3.2. Class RequestLoggerService

RequestLoggerService is the service class of the web service. It contains two functions which store data specified by their parameters in the database:

- **saveRequestData(String name, Date dateTime, String ip, String freetext)** – Saves the data about a request handed over by the parameters in a MySQL database.
- **saveResponseData(String name, Date dateTime, String ip, String freetext)** – Saves the data about a response handed over by the parameters in the MySQL database.

7. Annex 3: Ontology and Metadata Services

An *ontology* is commonly defined as: *"a [formal,] explicit specification of a [shared] conceptualization"*[5] [Gr93]. More specifically, an ontology explicitly defines a set of entities (e.g. classes, relations and individuals) imposing a structure on the domain that is readable by both humans and machines. A variety of representation schemas have been proposed to encode this ontological conceptualisation, such as CML, CycL, KIF, Loom and the Web Ontology Language (OWL).

Given the ontological concepts and relationships, the business objects can be described explicitly as an ontological abstract. The ontological abstract is a summary of semantic properties of the business objects and contains the semantic data with which any semantic-driven processing can take place. The added value of the ontology here is to express the meaning explicitly either for human operation on clear and precise understanding or for automatic processing and resource management.

According to the double articulation philosophy a DOGMA ontology consists of an lexonbase of plausible fact types (called lexons), which holds (multiple) intuitive conceptualizations of a domain, and a layer of reified ontological commitments [SYM08]. The latter essentially are views and constraints that within a given context allow an application to commit to the selected lexons. This setups allows the construction and application of semantic patterns, which can be compared at various levels of granularity. In this way, scalable ontological solutions for eliciting and applying complex and overlapping collaboration patterns can be built, and semantic alignment can be achieved [Co08].

7.1. Functions Realized

Ontology Engineering

The DOGMA ontology engineering approach is composed of two main steps: semantic reconciliation and semantic application (see Figure 1) [DLC08].

The semantic reconciliation is composed of five phases (Figure 1). The user (or knowledge engineer) first elicitates the scope of the ontology by collecting abstract facts (e.g. logical schemas and natural language sentences). These facts are created and then formalized into

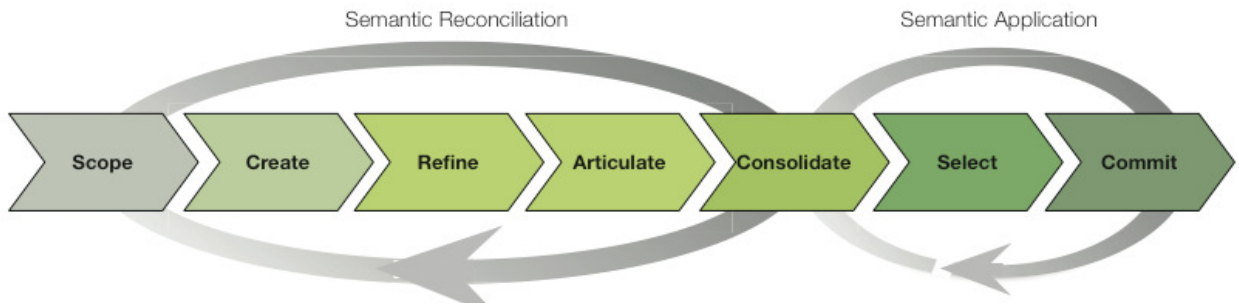


Figure 6: Agile ontology engineering methodology

lexons. A Lexon is a quintuple constructed from, and uniquely identified by the combination of its (i) context from which it was elicited (identified by a URI), its (ii) head- and (iii) tail-terms, and its (iv) role and (v) co-role terms. An example of a hypothetical lexon elicited from a packaging standard is <driving, person, drives, driven by, car>. Finally, redundant elements are removed. This first step results in a number of reusable language-neutral patterns for constructing semantics that are grounded in informal meaning descriptions.

During the *semantic application*, existing information sources and services are committed to a selection of semantic patterns. This is achieved by selelecting the appropriate patterns, constraining their interpretation and finally mapping (or committing) the selection on the existing data sources. In other words, a commitment creates a bidirectional link between the existing data sources and services and the business semantics that describe the information assets of an organization. The existing data itself is not moved.

Ontology Evolution

DOGMA's collaborative ontology evolution process is driven by the current semantic interoperability requirements that emerge from the community dynamics. For example, semantic interoperability requirements identify the conceptions that are needed in the common ontology in order to ensure *semantic interoperability*. This process consists of four main phases [DL08]:

1. **Community grounding:** The core domain expert interprets the domain and identifies the conceptions that are needed to enable semantic interoperability. This leads to the identification of a meta-ontology framework. The resulting *community representation* is an initial common ontology that includes the conceptions identified in semantic interoperability requirements, and that is hooked into the chosen meta-ontology framework.
2. **Perspective rendering:** All stakeholders interpret the semantic interoperability requirements and the initial common ontology, and render stakeholder perspectives. The result is a set of subjective representations that contextualise the conceptions identified in the semantic interoperability requirements. Perspective rendering can be formally restricted by perspective policies (see [DLM08]) . Furthermore, these perspectives are stored and versioned in the DOGMA ontology framework.
3. **Perspective unification:** All stakeholders interpret the set of stakeholder perspectives and collaboratively agree on a unified perspective on the initial ontology. The result is a minimal inter-subjective representation of the conceptions identified in the semantic interoperability requirements that are socially accepted. The initial ontology is updated with the unified perspective accordingly.
4. **Community Commitment:** All stakeholders interpret the new ontology version and commit their applications (e.g., web service interfaces or business process models) to it. The result is a formal representation of the semantic interoperability in the community.

This four-phase cycle is repeated until satisfying consensus on the relevant conceptions is achieved with minimal effort. The explicit rendering of stakeholders perspectives allows us to capture the ontology evolution process completely, and validate the ontology against these perspectives respectively. Ultimately, co-evolving communities with their ontology will increase overall stakeholder satisfaction. This makes DOGMA-MESS convenient for this project given the continuously evolving business processes within and between stakeholders.

Calculating Semantic Distance

The matcher calculates semantic distance in order to find optimal marche for a given request. It can be classified as a hybrid matcher: it uses a number of existing and newly developed matching algorithms to compute a similarity score between two entities [5]. The matching methodology comprises four steps; namely Feature Engineering, Building the Search Space, Similarity Iterations, Computing the Final Score.

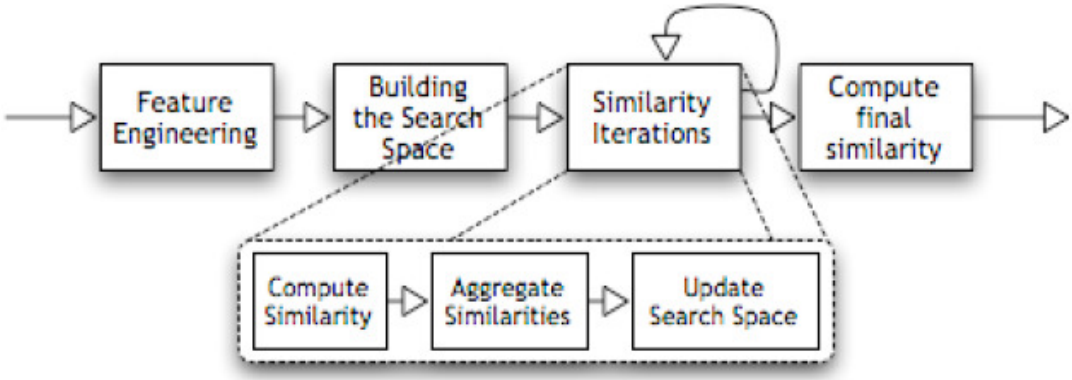
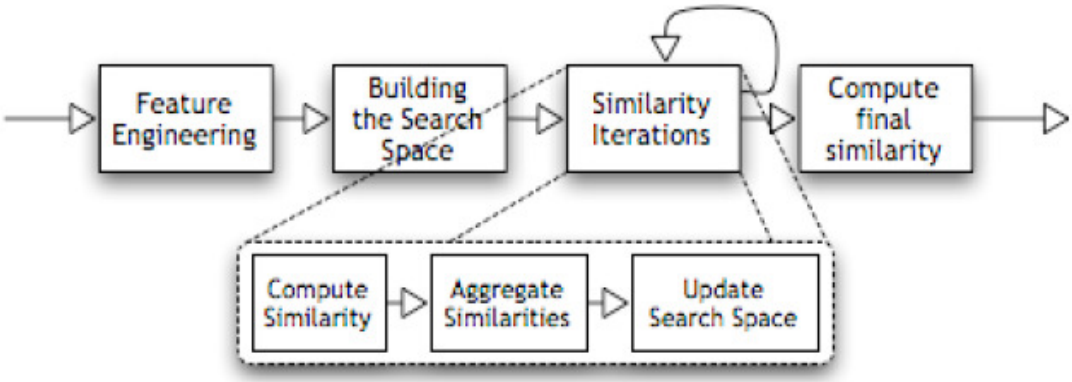


Figure 7: DMatch methodology



The first step, the feature engineering, transforms the initial representation of both commitments into a format that can be used by the different matching algorithms. We define the structure that is used to compare the concepts from the two commitments as a commitment rule. The rule maps one concept form the source ontology to one concept from the target ontology. While the denomination commitment rule complies with the DOGMA terminology, from here on, we call it mapping element to be consistent with the literature on ontology integration.

Secondly, the derivation of the mappings takes place in a two-dimensional grid, the search space, consisting of all the mapping elements. During the matching process, the similarity scores of the mapping elements are updated. In the end, the search space is used to compute the final relevance score between the two ontology commitments.

Thirdly, our hybrid matching approach uses several matching algorithms and combines their results to efficiently compute the similarity score of each mapping element. This is done in an iterative manner where the similarity score and possibly the linguistic relation of every mapping element is updated with each iteration. This approach allows us to easily adapt the matching process when the limitations and constraints set by the matching context changes. Each iteration is composed of three phases, i) the algorithm computes the similarity score

between two entities from the search space and is normalised (i.e. in the interval [0,1]), ii) the new similarity is aggregated with the existing score for the two entities using an aggregating function, and iii) the updated mapping element is stored in the search space.

The DMatch matching framework currently supports several matching algorithms, combining several different levels of matching. Initially, the terms representing the concepts are matched using well-known string matchers such as Levenshtein distance and Soundex similarity. In the next step, the concepts are matched on a linguistic level, using the WordNet thesaurus [Fe98]. The concepts are mapped to (several) WordNet synsets (if possible) and the linguistic relations between these synsets are then extracted. Also the descriptions of the different synsets are compared using the Jaro algorithm. On the semantic level, the contexts of the concepts are compared, which can solve the problem of homonyms. We also compare the gloss descriptions of the concepts using the Jaccard similarity coefficient.

Finally, DMatch extracts one-to-one mappings from the resulting set of mapping elements. Thus, one gets an average score based on that subset of mappings.

Semantic Integration

The DOGMA tools offer a run-time server that allows parties to semantically integrate their different data sources scattered around their different infrastructures and services. It enables them to publish (publicly or privately) at real-time all these different data sources as high quality and integrated intelligence. The semantic possibilities of the run-time server allow the parties to transform ambiguous, scattered data to well-understood and well-defined information in the model of their choice.
The run-time server delivers the following services:

- automatic transformation: instead of manually writing transformation programs, the run-time server automatically transform information from one format to another when both these formats have been committed to the ontology.

- scalability & manageability: the generated transformations greatly diminish the problem of proliferation of data formats and custom transformation code. The management is centralized through the commitments and ontologies, while the operational transformation is still distributed, point-to-point.

The functionality of the semantic integration server is exposed as so-called data services. This is a new integration paradigm, which fits well in a Service Oriented Architecture.

Semantic Annotation

Using the tools, the end user can add semantic annotations to his/her existing resources (e.g., an existing database, web service, web page, …). These annotations are commitments on the semantics, and leverage existing resources with semantics, without the need for internal change.

7.2. Interfaces

User Interfaces

DOGMA Studio Workbench is able to edit lexons, edit lexon term definition, link lexon term to WordNet, browse lexons, visualize lexon NORM trees, visualize some ORM constraints that are applied to the lexon NORM tree at the commitment layer. There is also full versioning and change log storage at Server side. The GUI contains five different areas dealing with specific functionalities. The user interface provides a lot of easily accessible search functionality, as well as drag and drop so as to facilitate and stimulate ease of use and reuse.
The Context Navigator lists, in a tree-like manner, all the contexts that are present in the lexonbase. These serve as containers for a variety of semantic patterns, and can be manipulated in this plugin.

The Ontology Viewer is responsible for retrieving ontologies and templates from the server and list them in a tree view for the user. Items are sorted first by domain, then by ontology and finally by version. When a user double-clicks on an ontology, it will be opened in the Ontology Editor. Also, this plugin serves as a pattern browsers to provide a graphical representation of accessing and manipulating semantic patterns.
This Ontology Editor is used to browse and edit a selected ontology or template. Its a multipage editor with four views for viewing/editing the taxonomy, viewing/editing the relations, viewing/editing the whole ontology and finally a small log for displaying information about the process to the user. When the user clicks on an item in one of the three views, a message is sent to the Ontology Operations plugin and the Concept Viewer plugin. The first will display all of the available operations on that object and if the selected object was a concept then the latter will try to fetch a concept definition from the Concept Definition Server.

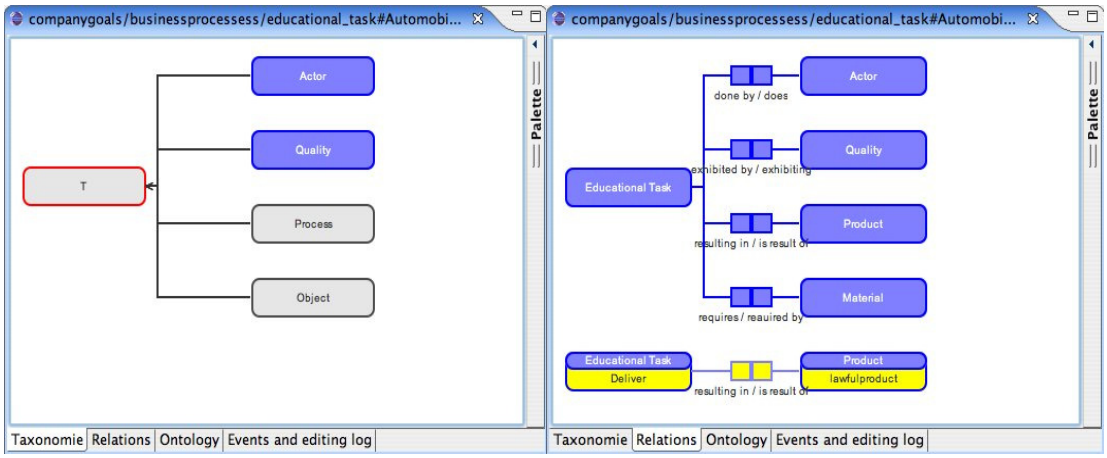


Figure 8: The left-hand side shows a view of a taxonomy, while the right-hand side shows how we can view/edit relations in DOGMA Studio Workbench.

The Ontology Operations plugin receives messages when something is selected in the Ontology Editor. The plugin is then responsible for displaying the possible operations on the selected object. The user is presented a different set of operations when he selects the ontology (by clicking on the background) or a concept or a differentia (a non-taxonomic relation).

The Commitment Editor plugin is used to create commitments that consists of a selection of semantic patterns, a number of constraints on these patterns, and the mappings to the application data sources. There are two ways to add constraints to a commitment. The first way is to select the Unique or Total tool and click on a role or co-role.

The second way to add constraints is to select the roles you want to put a constraint on, right click, and select the constraint you want. For example, you can select two roles, right click, and put a uniqueness constraint on both roles. The following screenshot depicts the example.

The Concept Viewer retrieves the concept definition from the Concept Definition Server whenever the user clicks on a concept in the editor. If the concept has no concept, it will inform the user that it is not hooked to a concept. The same happens when the user clicks on something that is not a concept. When the concept has a definition, it will fetch the glosses and synonyms of that concept and display them in their appropriate boxes. If something wicked happened while retrieving concepts from the server (server problems, network failure, etc.) a message will be displayed in the plugin.

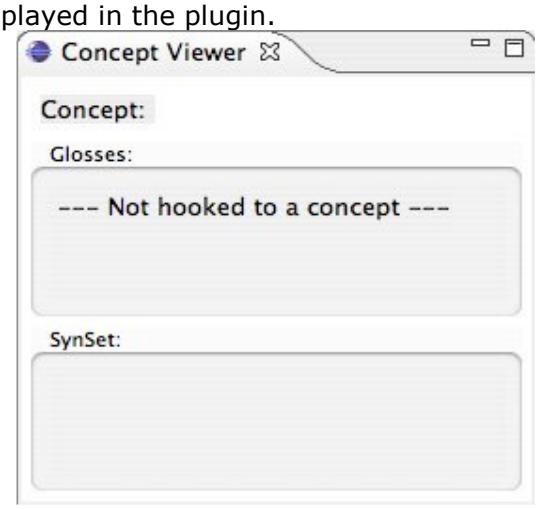
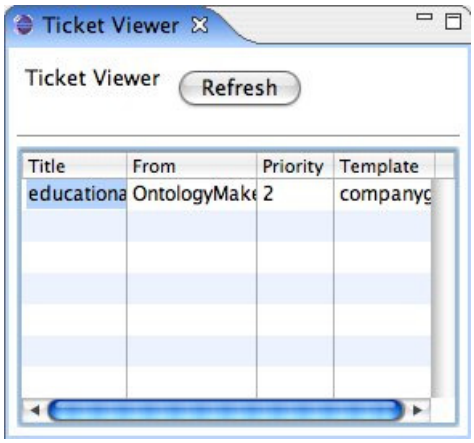


Figure 9: A concept viewer

Finally, the Ticket Viewer *will fetch tickets from the server and list them to the user. Tickets contain a title, a motivation or extra information, a priority code, the ontology related to the ticket, the stakeholder who created the ticket, and the stakeholder who receives the ticket. When the user double clicks on a Ticket, a dialogue box is opened asking the user if the ontology needs to be opened in the editor.*



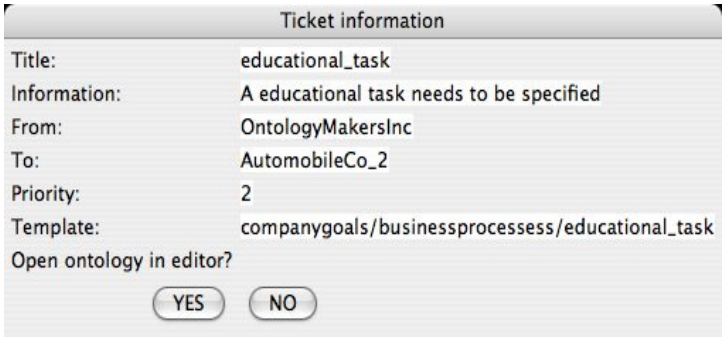


Figure 10: The left-hand side represent the ticket viewer, while the right-hand side shows the dialogue box.

In addition, several plugins have been developed to maintain other aspects of the ontology. The mapping editor creates (or selects) ontological commitments by mapping parts of an ontology to the data structure if a given application. The administration of users, user groups, systems and modules is done by the user management plugin. The Workbench also provides several tools that are not immediately accessible by a user. The Template Maker, which is used to make a template from an ontology after several non-taxonomical relations have been introduced. The Ticket Maker, which sends a ticket containing a reference to a template as well as additional information to several organizations. Those organizations are then able to open those tickets.

The Search plugin allows the user to search through all available semantic patterns in the server, retrieving terms, roles, concepts as well as larger semantic patterns. Based on the results from the search, the user can drag and drop the relevant patterns to speed up his/her own ontology development process.

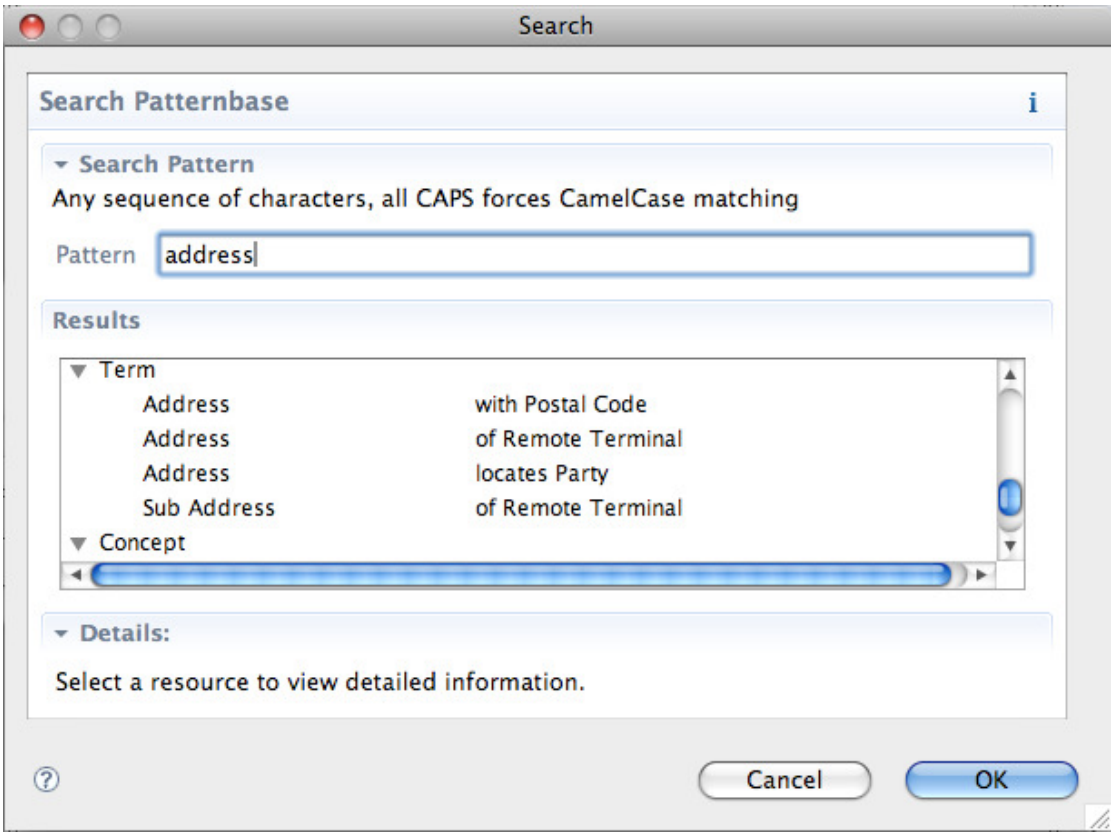


Figure 11: The Search plugin

Also, there are several Eclipse plugins that are reused to support the ontology engineering process. For instance, the Navigator View provides functionality to create projects, which is useful for collecting different sorts of information related to the process. Next to this, there are also a variety of import/export plugins which allow the user to migrate the semantic patterns (in or out).

Component Interfaces

The DOGMA tools offer a run-time server that allows parties to semantically integrate their different data sources scattered around their different infrastructures and services. It enables them to publish (publicly or privately) at real-time all these different data sources as high quality and integrated intelligence. The semantic possibilities of the run-time server allow the parties to transform ambiguous, scattered data to well-understood and well-defined information in the model of their choice.

The DOGMA server is composed of different layers. The Lexon Base layer stores uninterpreted, extensive and reusable pool of elementary building blocks, called lexons, for constructing an ontology. Although ontologies can differ in syntax, semantics, and pragmatics, they all are built on these shared vocabularies in the Lexon Base.

The Commitment layer stores the semantic constraints, and rules applied to a subset of lexons. An ontological commitment constitutes an axiomatisation in terms of a *network of* lexons logically connected and corresponds to the intended meaning of the application (-domain). An important difference with the underlying Lexon Base is that commitments are internally unambiguous and semantically consistent. Once elicited, ontological commitments (i.e. Ontologies) are used by various applications such as information integration and mediation of heterogeneous sources.

The Concept Definition Server (CDS) layer is a database based on the structure of WordNet. The user searches for a term, and gets a set of different meanings or concept definitions (called synset in WordNet) for that term. A concept definition is unambiguously explicated by a gloss (i.e. a natural language (NL) description) and a set of synonymous terms.

The Versioning layer provides all the functionality for identifying and storing ontology versions, using change logs and evolution operators.

The User Management layer contains all the specifics related to administration of users, user groups, systems and modules.

7.3. Technologies Used

STARLab uses the Eclipse PDE (plugin development environment) to develop Dogma Studio, a tool aiding the ontology engineer in the different stages of creating and maintaining ontologies.

The DOGMA Studio Server is a robust ontology server, developed in J2EE and deployed in a JBoss application[6] and backed by a PostgreSQL[7] relational database. The database hosts the relational model of the DOGMA framework, whereas the J2EE application layer provides the functional model for manipulating the ontologies and hosting annotations.

7.4. User Manual

As part of TAS3, VUB shall deploy an ontology engineering platform, called DOGMA-MESS Studio, to enable the different partners to browse and model ontologies. The functionality offered by this platform will be extended based on the requirements. The resulting platform will be available from the VUB server via remote login.

---

6 http://labs.jboss.com/portal/jbossas
7 http://www.postgresql.org/

After launching the workbench and activating the perspective, the workbench should look something like this. For a description of the different components (or plugins) we refer to Section 5.1.2.

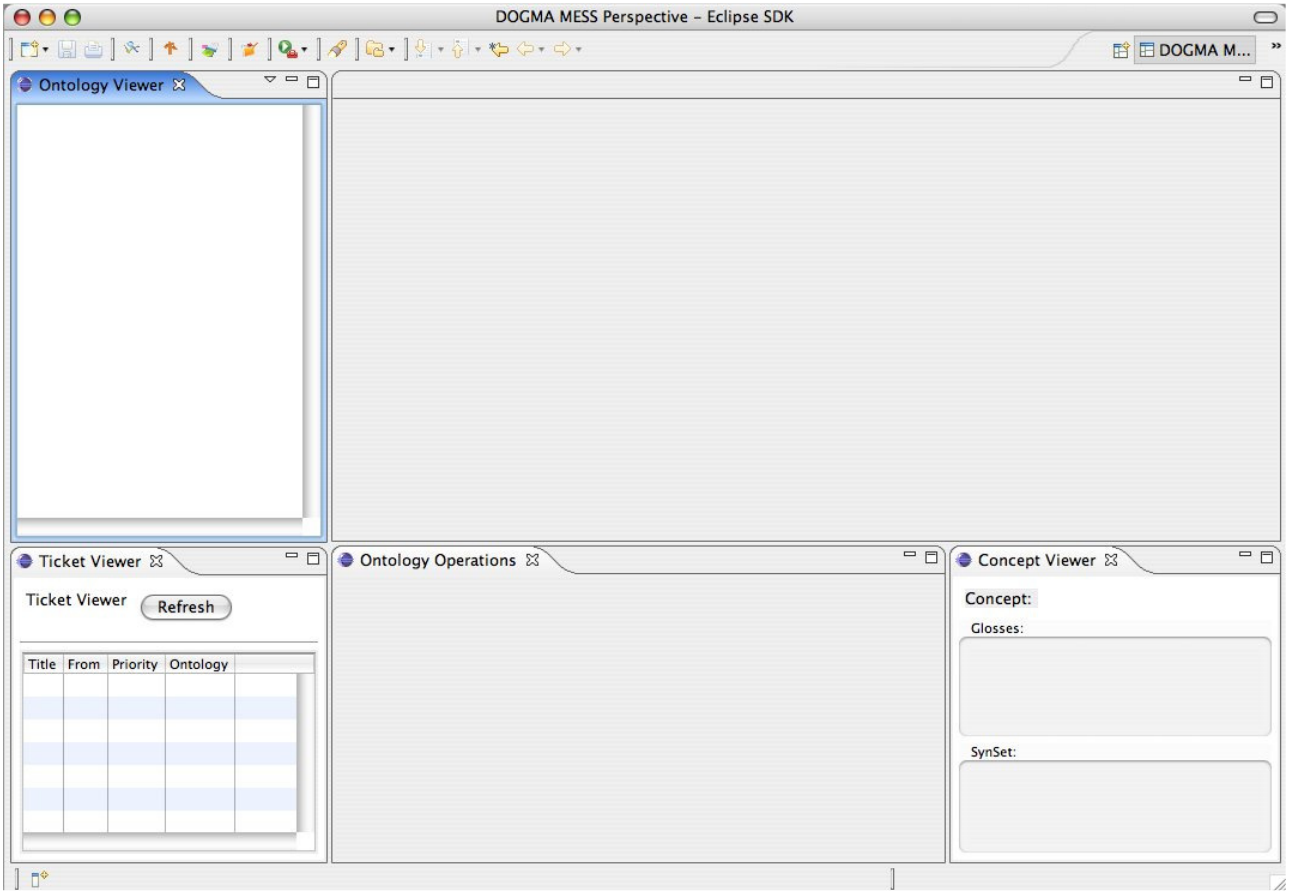


Figure 12: The DOGMA Studio Workbench perspective.

Opening a Ticket

Now the user can go to the ticket viewer and double click on a ticket. A dialog window will pop up displaying information about the ticket and asking if she wants to open the ontology or template in the editor. Clicking yes opens it in the editor. A new pane will show up in the perspective with an entry T, which is the most general type.

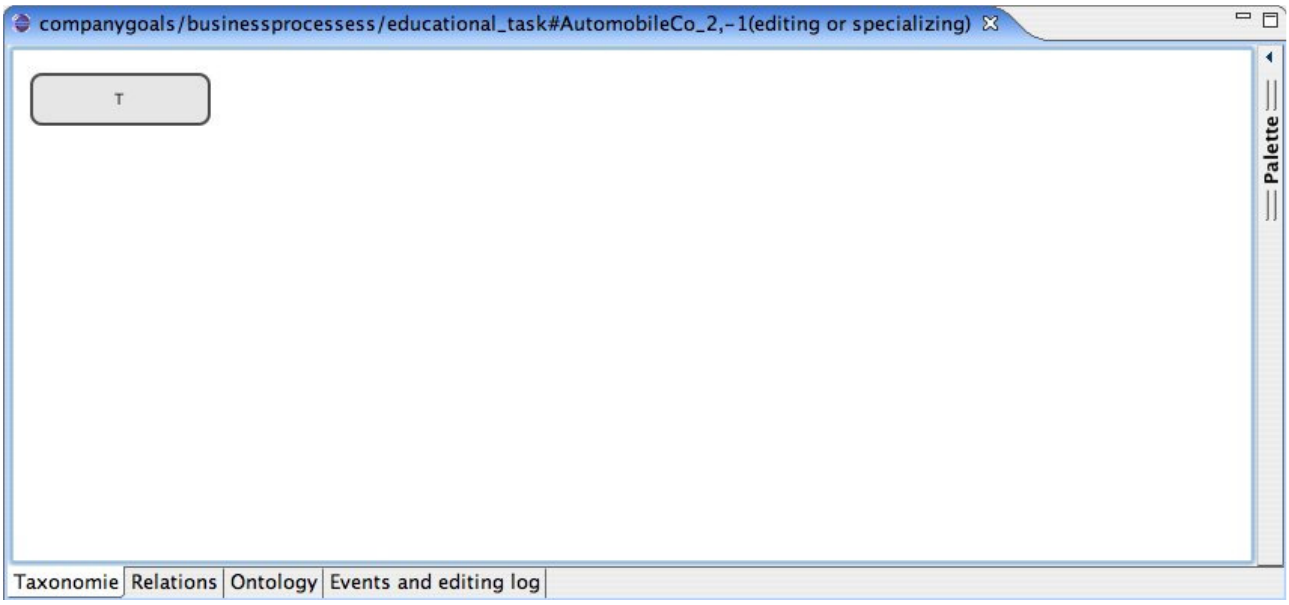
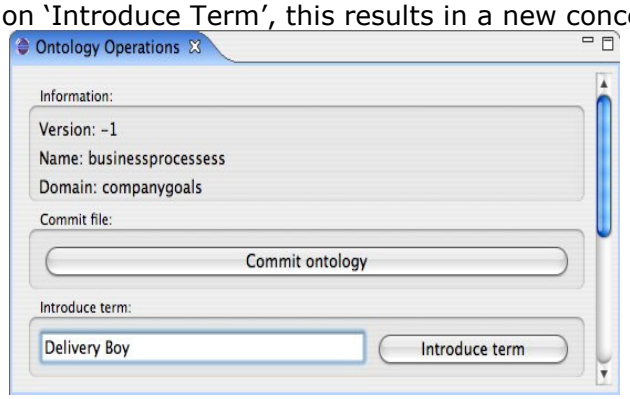


Figure 13: A new ontology in the Ontology Editor

Double clicking on a concept expands it, double click it again collapses it. When the user clicks on something in the editor, a message is sent to the Ontology Operations plugin and the Concept Viewer plugin. When clicking on a concept, the Concept Viewer plugin will try to look up the definition in the concept definition server. If it is not hooked to a definition, it will notify the user of that. Otherwise the glosses and synonyms of that concept are displayed in the appropriate boxes. The Ontology Operations plugin will display operations the user can perform on the selected item.

Editing an Ontology

To introduce a term, we click on the ontology (the white background) in the editor and then look for the introduce term operation in the Ontology Operations plugin. We give it the name 'Delivery Boy' and click on 'Introduce Term', this results in a new concept added in the editor.



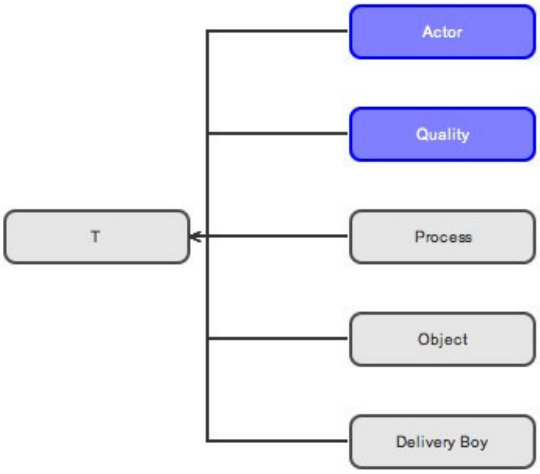


Figure 14: Panel to introduce a new term to the ontology, while the right-hand side show the resulting taxonomy.

'Delivery Boy' is put right under the root of the taxonomy, 'T'. This is too general and we want to change its genus to 'Actor'. We click on the concept 'Delivery Boy' and look for the operation 'Change genus'. There we choose 'Actor' and click on the button to execute the operation (Figure 9).

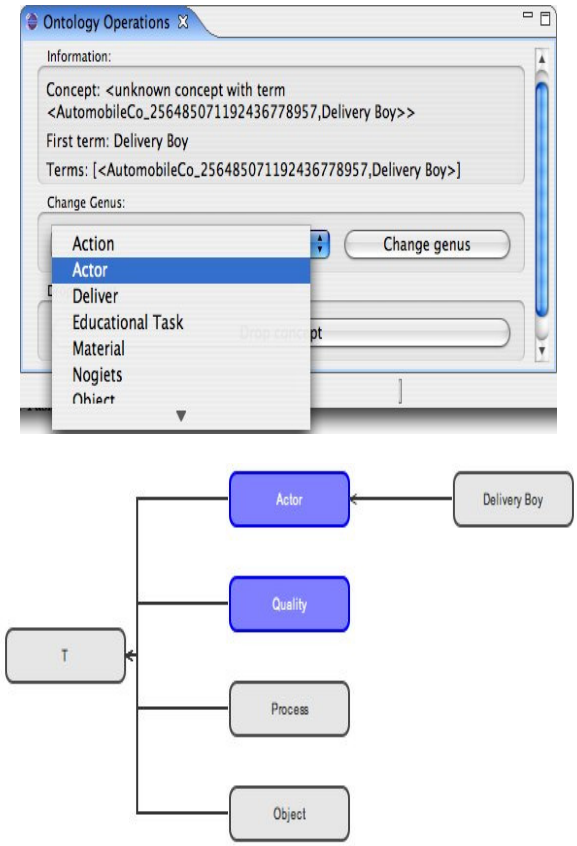


Figure 15: Pane to modify the genus of a concept.

The view in the editor changed as a result of this. The ticket informed us we had to specialize a template. If a template is provided, its root (a template involves specializing the relations around a certain concept) can be found in the relations editor.
When you click on a relation (the line between to concepts) the operations will show up in the Ontology Operations plugin. We will now specialize the relation. <Educational Task, done by,

does, Actor>. Click on that relation and look for the operation 'Specialize Differentia'. Choose two concepts (specializations of those concepts) and click on specialize differentia.

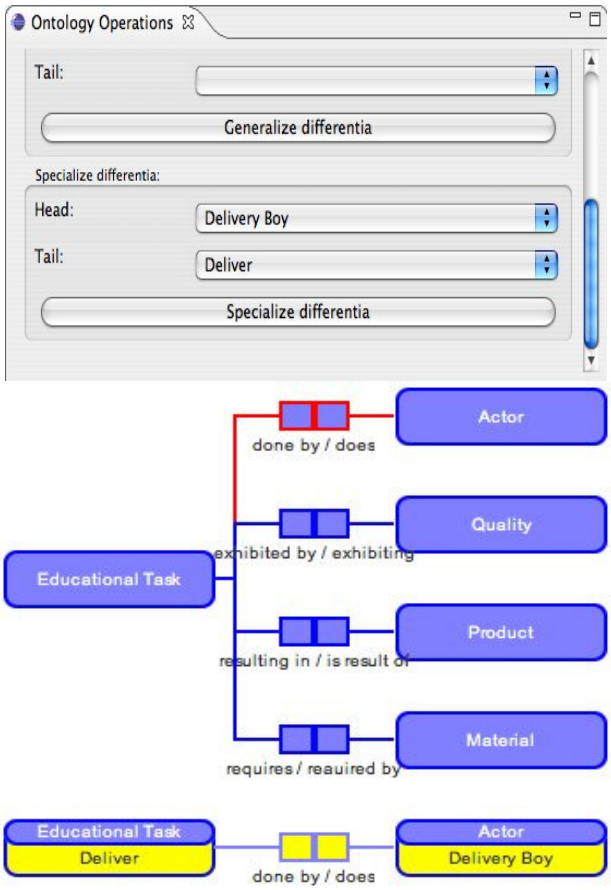


Figure 16: Pane showing how to specialize differentia and the resulting ontology.

Below the new specialized relation appears and the concepts are indicated with a different figure. When non-taxonomic relations are specialized, the concepts have their taxonomical parent indicated above them.

7.5. Limitations and Known Issues

The Dogma Studio tool suite needs improvement in terms of user functionality based on further user experience studies. It needs to cover the various application functionalities and its performance for the service of ontology consultation and reasoning may be improved. Both user experience and online application functionality consequently can make the current theoretical assumptions, design structure and implementation inappropriate, thus motivating system overhaul.

The tool suite (its design and code) is pre-existing knowledge which has not been developed within TAS3 and is not open source. It is part of the intellectual property of STARLab's commercial spinoff, Collibra, under the name of Collibra Studio. We have included this description here since it is a basis for achieving Task T8.4 (Metadata tool implementation). This tool suite will be made available to the Consortium on a VUB server.

References

[Co08] Collibra (2008). Semantic Alignment: why it matters. Collibra white paper (http://www.collibra.com/download/CollibraSemanticAlignment.pdf).

[DL08] P. De Leenheer, P. (2008) Keynote: Towards Ontological Foundations of Agent Community Evolution. In *Proceedings of 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2008),* pages 523-528, 2008.

[DLC08] De Leenheer, P. and Christiaens, S. (2008) Challenges and Opportunities for More Meaningful and Sustainable Internet Systems. In Proc. of Future of the Internet Symposium 2008 (Vienna, Austria), LNCS, Springer.

[DLM08] P. De Leenheer and A. de Moor, and R. Meersman. Context dependency management in ontology engineering: a formal approach. In *Journal on Data Semantics*, 8:26-56.

[Fe98] Fellbaum, C., ed.WordNet, an Electronic Lexical Database. MIT Press, 1998

[Gr93] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation*, pages 907–928, Deventer, The Netherlands, 1993.

[SYM08] P. Spyns, Y. Tang, and R. Meersman. An Ontology Engineering Methodology for DOGMA. In *Journal of Applied Ontology*, 3:13-39, 2008.

8. Annex 4 Aggregating Policies in User Centric Real-Time and Distributed Applications

By Thomas Kirkham (University of Nottingham), David Chadwick (University of Kent) and Ingo Dahn (University Koblenz-Landau)

Paper submitted to 1st International Workshop on Business Process Security (BPS09)

8.1. Abstract

Securing individual data objects using sticky policies in trusted networks is essential in user centric distributing computing applications. However aggregation of data objects presents a challenge in terms of sticky policy integrity for the new object. A possible solution is based on a mathematical merger of sticky polices associated with all aggregated data objects that respects all the individual policy rules in a new sticky policy for the data object. Whilst another approach is an aggregation using the policy enforcement framework of the trusted network to bypass the sticky rules. This process is enabled by the use of meta-polices that are introduced in this paper that explores the application both approaches in a employability demonstrator from the EU Framework 7 project TAS.

8.2. Introduction

Secure interaction between web services and their providers in composite applications is an ongoing area of research. The TAS3 project is adding to this debate by focusing on the role of the individual as a data and policy provider in these collaborations. Individual data security is vital for future of service based computing as applications evolve from more computer based roles such as business supply chain automation to user centric functions such as finding employment.

This document presents approaches to policy aggregation in a user centric employability demonstrator. It suggests a two stream approach of policy based aggregation. The first of these involves an automated mathematical formulation of a new policy for the aggregated object using the sticky policies of the composite data objects. The other approach takes into account the context of the aggregation and uses supporting services such as an Aggregation Service (with associated PDP) to overwrite the policy rules and use a specially selected meta-policy for the new aggregated object.

8.3 Related Work

The use of policy information is integral to the data sharing process in computing applications. Policy use can range from the definition of privileges for users and user groups in desktop computing environments to network access rules issued to students when they join an institution [1, 2].

The rapid development of service computing over resent years has seen the use of policy specifications become more distributed. In distributed service orientated computing environments the policy management environment has to be clearly defined and fine grained [3]. This refinement has led to the development of service infrastructures to support policy management in distributed systems and a whole range of associated policy standards [4, 5, 6, 7].

The support for policy based access to distributed applications and services is well supported in these frameworks and standards. However this has led to the creation of web service frameworks that can be seen to be bias towards the service provider perspective. The TAS3

project aims to readdress this balance by looking to define a policy based security mechanism for user data as it passes through distributed service based frameworks.

Central to this user centric focus on security is the association of data objects with specific policies. A popular method of achieving this is by the use of sticky policies [8, 9]. Sticky policies will allow the user to specify contexts and conditions around the usage of the data object as it passes between services. The use of sticky policies in a trusted network therefore has the ability to ensure that the user's policy constraints are respected when the data is out of their domain of control.

The model of sticky policy use provides a view on how a system of user defined security around data objects can be realised. However in practical terms this use of sticky policies in complex and dynamic distributed systems is not perfect. It is easy to envisage circumstances where access to the data is needed beyond the terms of the sticky policy. A good example of this is during aggregation of the data with data from other providers, or in the case that the policy needs to be over ruled in a 'break the glass' scenario [10].

In these scenarios the sticky policy has to either support specific contexts where an exception can be made, or the supporting framework will have a higher authority by which the sticky policy can be overwritten. In the break the glass scenario the policy is respected by using a specific contexts and the exceptional scenario is supported by a detailed logging and alerting process in the supporting framework. This process can be seen as a large overhead in terms of resource and not practical to repeat on a regular basis.

Data and therefore policy aggregation scenarios are likely to in some applications on a regular basis. It is fair to say that aggregation of data is likely to be a far more common task that the supporting framework has to support than 'break the glass'. In order to avoid the use a break the glass system of contexts and alerts, a common approach to policy aggregation could be deployed using mathematical algorithms. This approach is well established in previous research and can satisfy all the constraints of the merged policies [11].

However the mathematical approach is not likely not to suit all scenarios, particularly in increasingly real-time and dynamic applications. Using the mathematical approach is likely to create a resultant policy that is likely to restrict the use of the new data object, which could render it useless for an application. Thus another aggregation approach is needed to bypass the sticky policy in a similar way as the 'break the glass' but without the overheads. Building this approach along side a mathematical aggregation in a user centric distributed application is the focus of this paper.

8.4. Employability Scenario

The application scenario involves the user looking to present a set of resources to support their search for work. This personal data will be evidence supporting the users experience and qualifications in the workplace and within learning institutions. This data is owned by a variety of users in the scenarios and sticky policies defined by its owner are associated with each data object.

The demonstrator aims to initially address the challenge of how these policies are set and enforced in the framework. As users share their data through the trusted infrastructure with parties that can aid them in finding employment. The other focus of work is on the aggregation of data by services in the trusted network, for example in the process of seeking employment and / or skills matching.

In simple terms the TAS3 UK Employability demonstrator will involve four main groups which have to collaborate and share this data. In simple terms they are the learner, Small to Medium Enterprise (SME), University and placement co-coordinator. However, the use cases can be seen as more fine grained than the use of four separate groups. For example, the

learner can consist of a worker linked to the SME or a student linked to the University. The University may consist of data sources from a central Information Services Dept, or a distributed collection of data sources from different HE institutions. The SME may mirror this distribution of data and the placement co-coordinator may be part of a specific learning institution or a separate outside service provider.

Thus within each group individuals may have different rights to data and ability to define polices associated with their data in different ways. This flexibility is addressed in the project using the central TAS3 trust infrastructure as a means to define common understanding and policy definition.

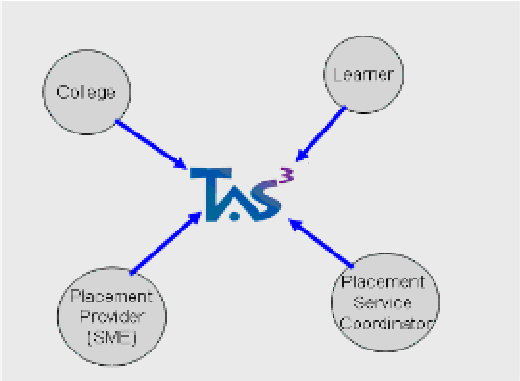


Figure 17: Main stakeholders in the student placement employability scenario.

Data use in the application example can be simply achieved via an upload by the user to an appropriate web service. Or via more complex presentation, for example (depending on the specific legal codes) the data may belong to the user but may be located elsewhere such as in a University database.

In these scenarios user authentication and authorisation in terms of access to the data is an important process. Once accessed the appropriate policies need to be associated with the data and these have to be both enforced and understood in the trusted framework. Understanding and respecting the policies defined by data owners when used in applications (provided by a wide range of service providers) is a technical challenge. This challenge is directly linked to the trust framework of TAS3.

8.5. Implementation

8.5.1. Trust and Policies

Data can be shared globally and personal data either via personal sharing (Social networking) or via organisational policy (credit report) can now be accessed freely via the internet. The only restriction placed on the data sharing is the policies associated with the data which can be defined managed in a variety of ways. These can range from data polices defined by the data provider and enforced by trusted third parties i.e. DRM or defined by third parties and enforced by the user (sticky policies) [12 ,8].

Overall these policies have to define what elements of the data the user can access and what also the user can do with the data once it is received. Polices associated with the data are enforced by trusted parties as the request for the data is made commonly referred to as Policy Enforcement Points (PEP). In terms of DRM the PEP would be a trusted third party in an un-trusted environment, but in the case of sticky policies the PEP is a trusted third party in a trusted environment.

In trusted systems sticky policies are used to link policy to specific data objects. The sticky policy is defined by the data owner and associated with the data as it is sent (unlike DRM sticky policies are more fine grained and give access to the data rather than preventing access to the whole).

The trust is influenced in the framework often by negotiated legal SLA and terms of use agreements between the user and data provider. This ensures that the sticky policy is respected and any forwarding of the data in the network maintains this link. However, as applications scale and trusted systems become larger the uses of sticky policies do not scale well, particularly in the case of data aggregation.

8.5.2. Aggregation

As discussed aggregation of data in large scale distributed applications is often needed to present application results. In terms of a basic aggregation process the merger of objects can be seen to create a new object. For example if a report is needed that consisted of a users photograph, English grade and passport number. The process of forming the report would be an aggregation of these 3 objects. This process can be seen in Figure 2.

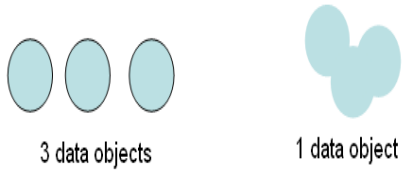


Figure 18: Basic aggregation

When sticky policies associated with data objects are added to the process the basic aggregation it is logical to view the aggregation process as a merger of policies in the same way as figure 2.

This approach can be seen as an automated or local aggregation that can be done using an algorithm and the sticky policies associated with the data, no outside authority is needed to perform it.

4.2.1 Automated / Local Aggregation

In this model all the policies are aggregated along side the data. The process is logical following on from the basic aggregation concept and the policies are all respected and merged into an overall policy for the aggregated object.

The advantages of this approach are that the transformation can be achieved in an automated way by the aggregator that does not rely on negotiation or retrieval of data from other sources. The disadvantages stem from the fact that the large amount of policy statements are likely to severely restrict how the new data object can be used.

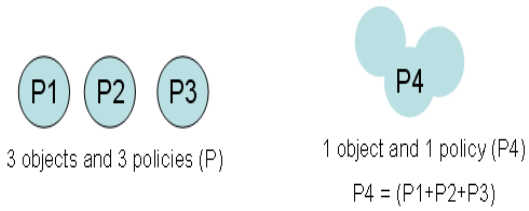


Figure 19: Automated Aggregation

Figure 1 shows automated aggregation in a simple case. Each sticky policy could be designed so it can fit as part of an aggregation of similar object or could include a set of aggregation contexts to allow aggregation in specific circumstances.

The main driving force behind the aggregation is a mathematical formulation to check all policy statements are respected. Other aggregation equations may be introduced to better align policy definitions.

4.2.2 Trusted Aggregation

Problems aggregating sticky policies are linked to various factors ranging from problems integrating different policy formats, scalability of aggregating multiple policies to individual policy rules that could prevent certain aggregations. In these cases the sticky policy has to be over ruled.

To achieve this as in the case of a 'break the glass' scenario the TAS3 project uses policy contexts and supporting trusted infrastructure. It is envisages that when a data provider signs up to a TAS3 application they will agree to certain terms of use. These will include certain levels of service that they can expect from the application as well as guarantees over security of data. In addition the user will also agree to specific operations that the TAS3 application owner can conduct on their data.

Central to these operations will be the ability for the TAS3 infrastructure to overrule the sticky policy on the data objects. This process will be controlled by a set of specific circumstances in when this can be done and a clear explanation of the data that will be both aggregated and presented as a result. All aggregations will also be fed back to the user.
In order to standardise the methods of aggregation and the way they are presented in TAS3 the concept of meta-policies is introduced.

The meta-policies are be generated from the terms and conditions agreed to in the joining process by data providers. As discussed some of these terms will allow the owner of the TAS3 application certain access privileges on the data objects and will be set as the sticky policies are also associated with the data.

4.2.3 Meta-Policies

The sticky policies are stored with the data objects and the meta-policies are stored in the central TAS3 architecture. The meta-policy and data objects are linked via a shared identifier created when the user add the data objects to the application. It is in the interest of the TAS3 owner to make the meta-policy definition and use public to the data providers in the system the same way as terms of agreement are shared in everyday contracts.

The meta-policy specifies what can be done to the sticky policy associated with the data object. The meta-policy definition will specific to each TAS3 instance, this will ensure that meta-policies are application specific and can only be used in specific domains. Using meta-policies cross application / domain maybe possible using a higher entity but this is not yet defined and discussed as future work.

The application of meta-policies in the TAS3 employment demonstrator will be via services provided by the TAS3 application owner. However it is possible that as such an application increases in size this approach may become a bottleneck. In such cases it is envisages that by the use of trusted third parties and the use of trust ranking and reputation mechanisms the services that use and manage the meta-policies may be provided by third parties. The data provider could specify the level of reputation that the service provider must posses that handles their meta-policy data.

The ability to over rule the use of sticky polices by the use of meta-policies will allow distributed applications to scale better and support greater dynamicity. However this is dependant on the Meta-polices being clearly defined and linked specifically to the sticky policies in application centric way and ultimately under the control of the user.

8.6. Applying Aggregation

8.6.1. Policy set up phase

Meta-policies are stored in the TAS3 infrastructure in the central Policy Store. This store is directly linked to the configuration tools that the data providers use to both create sticky polices and the meta-policies when the join the TAS3 application.

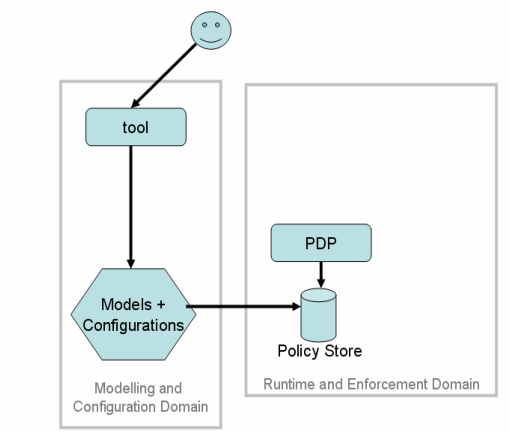


Figure 20: Main interactions in policy setup phase.

A typical interaction will stem from a user invoking a specific service in the TAS3 infrastructure, for example a skills matching service may be chosen. This service could be provided by a third party and may not be provided by the TAS3 application owner. The authorisation of access to the service will be provided by a PDP in the infrastructure.

8.6.2. Aggregation

The requesting web service may be a workflow engine and as part of the workflow an aggregation may be needed on several data objects. The aggregation could be performed by the engine locally using the automated approach, however if this is not possible or the resultant object is not usable an aggregation using meta-policies may be requested.

The Master PDP is present in the TAS3 infrastructure and handles requests for aggregation using meta-policies. In order to perform this service will contact the Master PDP for authorisation and if this is granted the Master PDP will provide the TAS3 Aggregator Service with the meta-policies to perform the transformation.

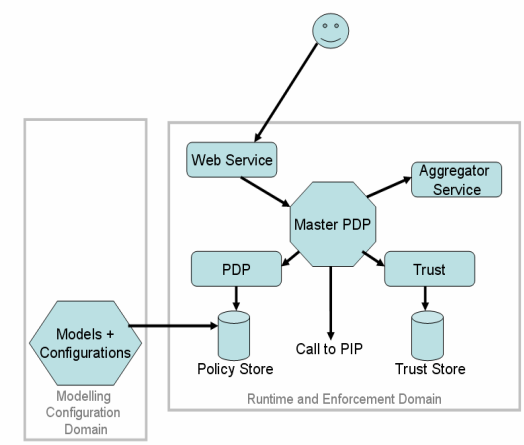


Figure 21: Aggregation Service approach

After the aggregation occurs a new sticky policy will be created with the new data object formed from the aggregation. This will be returned to the requesting service, the new policy will be limiting in the use of the new object to the service that requested the object and could also be secured by time to live (TTL) to the life of the workflow.

8.6.3. Disaggregation

In both aggregation cases any disaggregation process will return the aggregated data objects to their original state with their associated sticky polices intact, as far as is possible. But this may not always be possible, especially when "lose" aggregation has taken case, for example in the case of medical records when anonymised aggregated data is retrieved from a database. The aggregation service in both cases should be able to perform this function. Support for diagrregation into new data objects not present previous to aggregation will not be possible as this will be a breach of policy integrity.

8.7. Future Work

Communication and collaboration between TAS3 application specific domains will enable the wider use of meta-policy. However to date there is no clear vision or international structures in place to enable both reliable trust and information. For example EU level privacy and identity legislation could cater for such applications for example in cross border co-operation.

The framework also needs a means by which sticky policy language can be translated. This will make the system more flexible by handling sticky policies created externally to the TAS3 application domain.

8.8. Summary

The use of sticky polices and a trust network is a popular way of securing individuals data in distributed service orientated architectures. However this approach is not flexible for the demands of dynamic real-time distributed services when applied to the domain. The 'break the glass' method of policy bypass illustrates a way around this problem. However, aggregation of data needs to be supported by more lightweight and transparent mechanisms.

This paper presented two methods using automatic aggregation of policies and an approach using meta-polices agreed between users and service providers. The use of these polices in

well defined trust frameworks enables aggregation to take place to support the demands of more dynamic and real-time applications.

8.9. References

[1] Abrams, M.D. *Renewed Understanding of Access Control Policies*. In Proceedings of 16th National Computer Security Conference. 1993. Baltimore, Maryland, U.S.A.

[2] Allen JH: *The CERT Guide to System and Network Security Practices.* Addison-Wesley. 2001

[3] G. Karjoth, M. Hunter, A Privacy Policy Model for Enterprises, IBM Research, Zurich - 15th IEEE Computer Foundations Workshop, June 2002

[4] Damianou, N., N. Dulay, E. Lupu, and M. Sloman. *Ponder: A Language for Specifying Security and Management Policies for Distributed Systems*. The Language Specification Version 2.2. Research Report DoC 2000/1, Imperial College of Science Technology and Medicine, Department of Computing, London, 3 April, 2000.

[5] Sloman, M.S., *Policy Driven Management for Distributed Systems*. Journal of Network and Systems Management, 1994b. 2(4): p. 333-360.

[6] I. Foster and C. Kesselman, editors. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, 1999

[7] D. W. Chadwick and A. Otenko. The PERMIS X.509 Role Based Privilege Management Infrastructure. *7th ACM Symposium on Access Control Models and Technologies*, 2002

[8] Gunter Karjoth, Matthias Schunter, Michael Waidner. *Platform for Enterprise Privacy Practices: Privacy-Enabled Management of Customer Data*. In Privacy Enhancing Technologies, pages 69–84, 2002.

[9] Marco Casassa Mont, Siani Pearson, Pete Bramhall. *Towards Accountable Management of Identity and Privacy: Sticky Policies and Enforceable Tracing Services.* DEXA Workshops, pages 377–382, 2003.

[10] Ferreira A, Cruz-Correia R, Antunes L, Farinha P, Oliveira-Palhares E, Chadwick DW, and Costa-Pereira A*. How to break access control in a controlled manner.* Proc IEEE CBMS 2006: pp. 847-54.

[11] Rockafellar, R.T., and Wets, R.J.-B., "Scenarios and policy aggregation in Optimization under uncertainty", *Mathematics of Operations Research* v. 16 (1991), 119-147

[12] R. Iannella, "Digital Rights Management (DRM) Architectures," *D-Lib Magazine*, vol. 7, no. 6, June 2001