SEVENTH FRAMEWORK PROGRAMME
Challenge 1
Information and Communication Technologies

Trusted Architecture for Securely Shared Services

Document Type:	Deliverable
Title:	**TAS³ Lower Common Ontology**
Work Package:	WP2
Deliverable Nr:	D2.3
Dissemination:	PU
Preparation Date:	31 December, 2011
Version:	Final

The TAS³ Consortium

Benef. Nr	Beneficiary name	Benef. short name	Country	Date enter project	Date exit project
1	Katholieke Universiteit Leuven (Proj. Coord)	KUL	BE	M1	M48
2	Synergetics nv/sa	SYN	BE	M1	M48
3	University of Kent	KENT	UK	M1	M48
4	University of Karlsruhe	KARL	DE	M1	M48
5	Technical University Eindhoven	TU/E	NL	M1	M48
6	Consiglio Nazionale delle Ricerche	CNR	IT	M1	M48
7	University of Koblenz-Landau	UNIKOLD	DE	M1	M48
8	Vrije Universiteit Brussel	VUB	BE	M1	M48
9	University of Zaragoza	UNIZAR	ES	M1	M48
10	University of Nottingham	NOT	UK	M1	M48
11	SAP research (S&T Coord.)	SAP	DE	M1	M48
12	Eifel asbl	EIF	FR	M1	M30
13	Intalio Ltd	INT	UK	M1	M18
14	Risaris Ltd	RIS	IR	M1	M48
15	Kenteq	KETQ	NL	M1	M48
16	Oracle	ORACLE	NL	M1	M48
17	Custodix nv/sa	CUS	BE	M1	M48
18	Medisoft bv	MEDI	NL	M1	M12
19	Karlsrhue Institute of Technology (KIT)	KARL	DE	M1	M48
20	Symlabs SA	SYM	PT	M18	M30

Contributors

	Name	Organisation
1	Ioana Ciuciu	VUB
2	Cristian Vasquez	VUB
3	Quentin Reul	VUB
4	Gang Zhao	VUB
5	David Chadwick	KENT
6	Lei Lei Shi	KENT

Contents

Lists of Figures

Lists of Tables

Executive Summary

In this last iteration of Deliverable D2.3, we present the TAS^3 lower common security policy ontology based on the DOGMA (Developing Ontology-Grounded Methods and Applications) framework and its application to the TAS^3 scenarios via specialized tools. In particular, we define conceptual models associated with authentication, credential validation, access control, obligation, privacy, delegation, and audit policies. More specifically, we represent security policies as a declarative model by defining a set of concepts and the relationships between them rather than describing the explicit sequence of steps required to apply them.

Given this ontology, PDPs can interoperate with each other by interpreting policy attributes and their values from the Service Requester to those of the Service Provider through annotations. This removes the impractical restriction on all PEP/PDP in a TN to use an identical vocabulary to describe the conceptual model of their respective security domains. For instance, the credentials of an employee in System A together with their ontological annotation can easily be evaluated by the PEP in System B. As a result, the PDP in System B can call on ontology-based interpretation and translation (mapping) services to understand whether the presented credentials and their values are those required by System B or are equivalent to the conditions in its policies. In this respect, we have implemented the ontology-based interoperation service (OBIS), which has been validated in the Nottingham and e-Health use cases. The OBIS service has been extended by the OBIS Domain Mapper which maps attributes contained in access requests between different domains. This aspect is demonstrated in the health care pilot.

Regarding the semantic support for the security-annotated business process models, the annotator tool has matured to the point where it allows the process designer to retrieve security annotations from the dedicated knowledge base according to his/her query specification. This achieves the objective of the annotator, which was to support the user into creating correct security annotations to be enforced on business processes.

Note that the concepts defined in this document draw upon those defined in the Descriptive Upper Ontology and the TAS^3 Common Upper Ontology in Deliverable D2.2. The LCO presented here has been gradually refined over the last year of the project from the service (application) ontologies derived from the use cases and from the ontology of security constraints designed for secure business processes.

1 Introduction

One approach to ensure *secure* and *trusted* access to personal information is to develop security policies to control access to this information. A security policy is the primary way in which management's expectations for security are translated into specific, measurable, and testable goals and objectives [1]. One advantage in using security policies is that the security of a trust network can be managed dynamically without changing the underlying structure of the network [6].

Over the years, many languages have been developed to define security and privacy policies enabling secure access to personal information in distributed environments. For example, eXtensible Access Control Markup Language (XACML) [2] provides a lexicon and vocabulary to represent access control policies while exchanging data in service-oriented architecture. In particular, XACML provides a XML-based syntax enabling a Policy Decision Point (PDP) to determine whether a request to access a resource should be granted, and to return an answer to a Policy Enforcement Point (PEP), which allows or denies access to the resource. However, policy interoperability can only be achieved if the policy language used is interpretable by every system. An additional barrier may result from the fact that these languages do not represent the content of a security policy (i.e. the value for the conditions and actions associated with a policy) in the same way.

Semantic Web technologies provide the means to address this problem. The Semantic Web [3] extends the current World Wide Web (WWW) with resources in machine-understandable format. For instance, the conditions and the actions to be performed by agent in access control policies can be given meaning through the use of ontologies. An ontology is commonly defined as: "*a formal, explicit specification of a shared conceptualization*" [4]. More specifically, an ontology is a server stored unambiguous and flexible meaning agreement on the semantics of concepts and the relations between them in a given domain.

In this document, we present a security policy ontology based on the DOGMA (Developing Ontology-Grounded Methods and Applications) framework [5]. In particular, we define conceptual models associated with authentication, delegation validation, access control, obligation, privacy, audit and delegation policies. More specifically, we represent security policies as a declarative model by defining a set of concepts and the relationships between them rather than describing the explicit sequence of steps required to apply them.

Given this ontology, we demonstrate how it can be used in TAS3 to allow different PDPs to interoperate with each other by interpreting credentials and values from the Service Requester to those of the Service Provider through annotations. For instance, the credentials of an employee in System A together with their ontological annotation can easily be evaluated by the PEP in System B. As a result, the PDP in System B can call on ontology-based interpretation and translation services to understand whether the presented credentials and their values are those required by System B or are equivalent to the conditions in its policies. Thus, this ontology removes the impractical restriction on all PEP/PDP in a TN to use an identical vocabulary to describe the conceptual model of their respective security domains.

Tonti et al. [6] introduce five general requirements that any policy representation should satisfy regardless of its domain of applicability. The policy representation requirements are:

- *expressiveness* to handle a wide range of policy requirements arising in the system being managed;
- *simplicity* to assist custodians, resource owner and data subjects[1] to define policies based on their expertise;
- *enforceability* to ensure that formally defined policies can be implemented in different platforms;
- *scalability* to ensure adequate performance of the overall system;
- *analysability* to allow reasoning about policies.

Different security policy specification languages (e.g. Ponder [7], REI [8] or KAoS [9]) have been proposed to reduce policy conflicts and facilitate interoperability. The aim of these specifications is to enable system administrators and data custodians to easily define security policies. In contrast, our security and privacy policy ontology is grounded in natural language. The use of natural language enables technical (e.g. custodians, data controller) and non-technical users (e.g. data subject) to easily specify policies to control the access to data (e.g. a CV). This does not have an impact on the enforceability of these policies by agents as the DOGMA framework provides mechanisms to convert DOGMA-inspired ontologies into description logic paradigms [10]. As a result, our ontology provides a framework that (i) can be implemented to respond to policy requirements of a system, (ii) can be extended through agreements between technical and non-technical user, (iii) can be enforced by different systems, and (iv) can be analyzed by both human and software agents. Note that the concepts defined in this document draw upon those defined in the Descriptive Upper Ontology and the TAS^3 Common Upper Ontology [12] and have been continuously refined from the TAS^3 service ontologies (e.g. from the employability and health care pilots).

[1] A data subject is the agent whose personal data is collected, held or processed by a data controller.

1.1 Reading Guide

The rest of the document is structured as follows:

- Section 2 describes the motivation behind the use of ontologies in policy-based security systems.
- Section 3 presents several scenarios where ontologies can be used to manage policies. This section concludes with a concrete scenario for the application of the technology to TAS^3.
- Section 4 describes the model of security policies encountered in the TAS^3 project. Moreover, it provides the conceptual representation for the core elements of these policies; namely condition, subject, action, and target.
- Section 5 presents the conceptual model for different types of security policies; namely authentication, credential validation, access control, obligation, privacy, audit and delegation policies.
- Section 6 presents a lower common ontology for secure business processes used to define security policies.
- Section 7 presents the application of the TAS^3 Lower Common Ontology in the employability and health care pilots.
- Section 8 provides a conclusion on the current status of our work and presents future directions.

2 Motivation

This section describes the motivation behind the use of ontologies in policy-based security systems.

2.1 Security Control

Security control consists of five phases; namely *Identification*, A*uthentication*, *Authorization*, *Obligation*, and *Auditing* [13]. *Identification* is the process of establishing the identity of a user, process, or device, usually as a prerequisite for granting access to resources in an IT system. *Authentication* is the process of corroborating a claimed identity with a specified or understood level of assurance, while *authorization* focuses on the granting and enforcing of permissions based on the evaluation of applicable policies (such as access control policies). An *obligation* is an operation specified in a policy that should be performed in conjunction with the enforcement of a security policy decision. The *auditing* process comprises a review of the system records and activity logs to ensure that established policies and operational procedures have in fact been complied with.

The authorization phase focuses on the granting or denying of actions based on the evaluation of policy rules which specify the trust relationships between the service provider, identity provider and service requester. In a federated system (e.g. TAS³ Trust Network), credentials are issued by appropriate authorities (e.g. the British Medical Association) which service providers may trust sufficiently in order to rely on these credentials when granting access to their resources.. The process of assigning credentials to users is controlled by a credential issuing policy at the assigning site, whilst the process of validating credentials at the resource site is controlled by a credential validation policy. Once credentials have been issued, a service provider will determine whether these are sufficient for granting access to the service requester. The process of determining if access can be granted, based on the validated credentials, is controlled by an access control policy. When the policies are satisfied and trust is established between the service provider and the service requester, the access is granted to the requester.

2.1.1 Policy Languages

Policies are rules governing the behaviour of a system [14]. More specifically, a security policy defines a set of conditions, which are evaluated to determine whether a set of actions may be performed. Over the years, many languages have been developed to define security and privacy policies to control the behaviour of users. In general, these languages describe the *conditions* for an *action* to be performed by a *subject* on a *target*. The Condition states the environment for an action to be executed, while an Action is something that is done or performed on a target. A Target is any type of entities (e.g. personal data) that can be acted upon by an agent. In the eHealth domain, for example, an audit policy could state that any access to a patient record should be recorded in a log. In this case, the condition is the access of a patient record, while the action is to record the access in a log.

A security policy must be expressed in a formal language to be processed by computers. For instance, XACML is an access control policy language describing how to interpret access control policies while exchanging data in service-oriented architectures. Although policy languages vary in representation (e.g. XML, predicate logic), they consist of a lexicon and syntax to write well-formed expressions understandable by machines. The XACML lexicon provides a list of reserved terms (e.g. `Policy`, `PolicySet`), while the syntax specifies the grammar by which terms can be combined and computed (see Section 2 in [2]).

In a secure Service-Oriented Architecture (SOA) [15], service requesters (SRs) and service providers (SPs) tend to use the same syntax and lexicon (e.g. XACML) as well as vocabulary to express values when exchanging messages. In this case, the conditions and actions are expressed using the same vocabulary. However, it is unlikely that the SR will use the same vocabulary as the SP to represent values. For example, the action requested by the SR might use a different vocabulary than the SP and would lead to non-interpretation of the policy. This type of policy interoperability is normally scoped out of formal languages.

Thus, communication across different security services needs to respect not only a shared lexicon and syntax of the message, but also shared semantics. The shared semantics is the conceptual model of the "*details of the application*" with respect to which policies are created, managed, interpreted and evaluated. Ontologies can be built as an abstraction layer of common semantics about who, what, how, why, when and where for peer-to-peer distributed access control and trust management.

2.2 Conceptual Model

A security domain is a sphere of trust, where the behaviour of an agent is controlled by a set of policies. It is situated in a context comprising many variables such as: information systems, resources, actors, business processes, administrative regulations and legal procedures. Each security domain has its own conceptual model of who to access what, how, when and why. Examples are information systems or networks of businesses, governments, public and private services.

The collaboration and interoperation between organisations on a trust network is likely to cover different security domains and involve different conceptual models. For instance, the evaluation of the credentials associated with an agent in Security Domain A by a policy defined in Security Domain B requires the application not only to speak the same policy language, but also to have the same semantics for the value of these credentials. However, it is hardly true in reality as each security domain may impose its own interplays of actors, information systems, resources, constraints, and procedures. This means that a PDP validating an access control policy in Security Domain B depends on an interpretation service, which renders the semantics defined in Security Domain A into that of Security Domain B.

The semantics of values in different security domains need to be captured in the form of an ontology. This ontology can either be adopted and extended, or used to annotate the conceptual model of a security domain to achieve policy interoperability. Thus, the ontology enables PEPs in different security domains to enforce access control policies based on the credentials of an agent as well as their associated values.

2.2.1 Requirements

A policy language specifies how to formulate a policy, its attributes and values, structure and format of expression. The information specified in a policy (i.e. its meaning) is based on the semantics of the security domain in which the policy is to be applied. These domain semantics are not only confined to security semantics, but is also related to particular types of applications or systems, domains of application (such as health care, human resource management), business practices, etc.

The domain semantics are expressed by policy languages through the use of pre-defined vocabularies. However, these policy languages do not aspire to represent the entire domain semantics, instead it is intended to provide the minimal lexical and syntactic means to convey the domain semantics. As medium of representation, it leaves out the issue of interpretation. In other words, it does not address how the semantic of a security domain is managed, and shared to facilitate the policy creation process, nor does it define the process by which a security policy is interpreted during its evaluation. We propose to use an ontology for the purpose. Such an ontology not only provides the conceptual model of a specific security domain, but also describes the 'common meaning' in different conceptual models of the security domain underlying the specific rules and policies.

The application semantics of the policies can be summarised by the following questions:

- ***Who*** is being controlled by the policy?
- ***What*** is being controlled by the policy?
- ***How*** is an entity controlled by the policy?
- ***Why*** is an entity controlled by the policy?
- ***When*** is an entity controlled by the policy?

The meaning of the policy (i.e. its elements) is expressed in terms of these questions. More specifically, these elements are concerned with who, what, how, why, when depending on the security domain (e.g. organisation boundaries, system contexts and applications).

2.2.2 Ontology of Security Policy

The ontology represents the concepts and the relations between them to express the "*common meaning*" of different security domains, in which a security policy is evaluated. It serves as a vocabulary of shared semantics, with which service providers and service requesters can 'understand' each other across different security domains, regardless of whether they 'speak' the same policy language or not.

The ontology does not model the "*trust*" as such, but rather the basic concepts of security policies in the process of trust establishment. The conditions, subjects, actions, and targets specified in policies can give rise to different conceptual models depending on the security domain.

The security policy ontology has two major functionalities. Firstly, the ontology is used to define the vocabulary to express attribute-value pairs in a policy. Secondly, the concepts in the ontology are used to annotate elements in security policies. The next sections describe these functionalities in more detail.

2.2.2.1 Vocabulary

Under an access control policy, an agent shall be authorized to access a resource based on a list of attribute-value pairs in terms of the conditions, subjects, actions, and targets. Thus, the access to a target is granted if and only if certain conditions have been met. The ontology described in this document specifies the semantics of the names (e.g. condition, action) as concepts and the relations between them, while domain specific ontologies will be used to define their values. This makes it possible to import the content of security policies from existing ontologies and thus apply them to many domains.

2.2.2.2 Semantic Annotation

In Service-Oriented Architecture, each service provider and service requester sets up their security domain with necessary and sufficient policies governing its behaviour and trust management. In order to effectively interact with other services on the network, a service needs to expose its security policies in terms of concepts in an ontology. In other words, the conditions, subjects, actions and targets of a security policy need to be annotated with concepts defined in the common and local ontologies. These semantic annotations can then be used to assess the equivalence between entities (and their values) defined in different security domain.

Kim et al. [16] present the NRL Security Ontology, which complements domain ontologies, to enable the discovery of resources that meet security requirements. The concepts defined in these ontologies are used to annotate resources or security aspects of Web Services description and queries. The NRL ontology includes concepts about mechanisms, protocols (e.g. `SAML`), objectives (e.g. `Trust`, `Authorization`), and credentials (e.g. `X.509Certificate`, `RBACCertificate`). However, this ontology does not represent the semantics of security and privacy policies, which is required in the context of TAS³.

As a result, TAS³ has developed a security policy ontology based on the DOGMA framework. In this framework, lexons are used to express the relationships between concepts (see Section 5 in D2.2 [12]). More formally, a lexon is a bi-directional graph, with nodes being the concepts and edges the relationships. Based on these lexons, we can annotate elements in a policy with concepts (i.e. sets of lexons). These annotations serve as a means to evaluate the equivalence between attribute-value pairs in policies expressed in different languages. Thus, the annotation of policies by an ontology not only imposes the 'common sense' necessary for collaboration, but also provides additional flexibility for policies to be fit for particular needs of a given security domain.

3 Scenarios

This section presents several scenarios where ontologies can be used to manage policies and concludes with a description for the application of the security policy ontology in TAS^3.

3.1 Use of Ontology in Policy Creation

Suppose we have an ontology specifying the semantic of security policies with regard to their conditions, subjects, actions and targets. One way of using this ontology is in the policy creation and its management. In this case, the ontology provides a set of attributes and their types of values to define security policies. It can serve as the extended vocabulary of the policy language, with which policy rules can be directly formulated (e.g. POLIPO [17]). Or it can be used to annotate the vocabulary of the policy language, respecting the concerns of using a particular policy language other than semantic modelling and processing. Similarly, the ontology enables the representation of different security policies regardless of the underlying rule engine used to enforce and evaluate them.

Use of an ontology for either policy specification or policy annotation will open up the prospects of managing policies by their meanings, since the ontological attribute and values can be used as indexing for knowledge management during policy creation.

An annotator is currently being developed within TAS^3 to assist business modelers into specifying secured business processes. This tool is based on a LCO of security constraints described in Section 6.

The use of ontology during policy creation is in fact in line with well-proven software engineering practices: encapsulation for changes multi-disciplinary development, management of viewpoint of multiple stakeholders and domains, and reuse of resultant models in different context and abstractions.

3.2 Use of Ontology for Policy Interpretation

During trust negotiation, two parties exchange among other things information about their respective credentials (for service requester) and policy obligations (for the service provider). This information can be represented as a list of attribute-value pairs. Suppose the two parties use the same policy language for the information exchange. While the language governs the structure of the message, it would be unrealistic to assume that any two parties from different security and application domains would be prepared to adopt identical terms for the disclosure of credentials and obligations in the policy evaluations, such as `<SubjectMatch>`, `<ResourceMatch>`, `<ActionMatch>` and `<EnvironmentMatch>` in XACML. In reality, these attribute-value pairs in the condition of a rule need to be interpreted prior to matching. In other words, semantic matching removes any restrictions on the use of identical terms for credentials and values across different security domains.

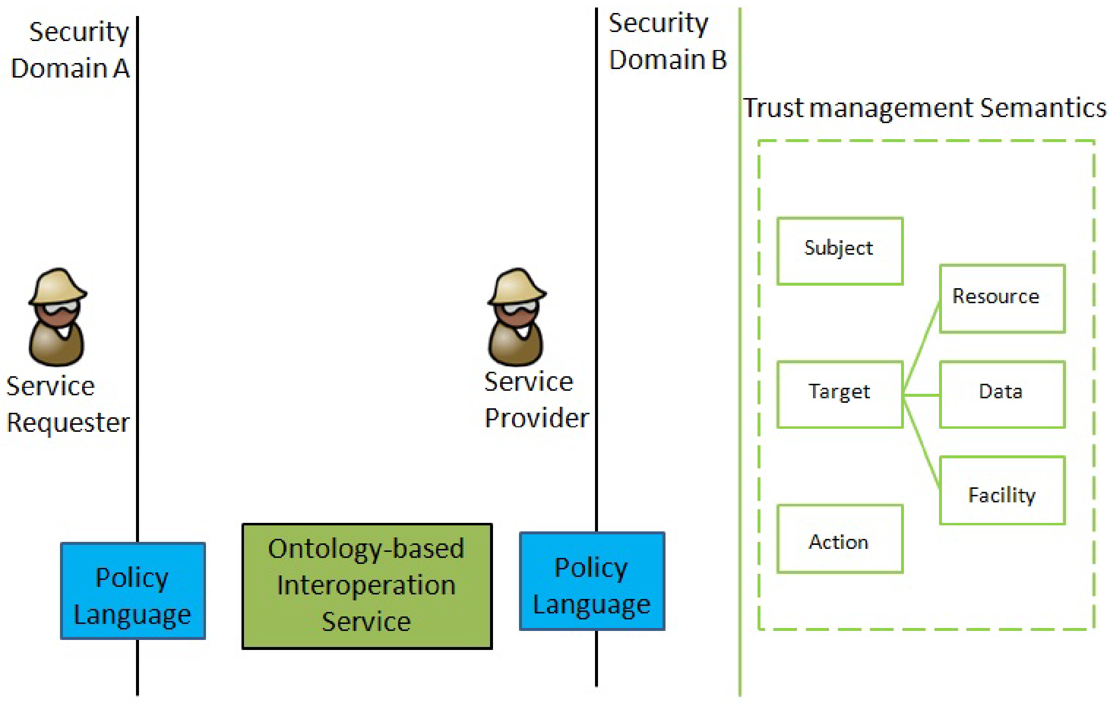


Figure 3.1: Interoperation Example

Figure 3.1 shows the different topics considered when interpreting a policy from Security Domain A in applications in Security Domain B. If the named instances (i.e. values for the different topics) have been annotated with concepts specific to each security domain, then we interpret as identical or different the values of the attributes (i.e. denotations) for the different parties. In this approach, an annotation is a set of lexons (from the Lexon Base) representing the relationship between concepts and serves as semantic definition for the credential or value. Thus, annotations can be used during the policy evaluation process (i) to disambiguate different denotation of the same named instance, (ii) to recognise identical denotation for different named instances, and (iii) to detect the generalization/specialisation between named instances (e.g. the generic requirement can be satisfied by a more specific fulfilment).

This approach has been applied in the authorization and authentication process in the Nottingham pilot, for the interoperability between service requesters and service providers, based on the ontology-based interoperability service (OBIS).

3.3 Use of Ontology for Policy Interoperability of PDP

Given a shared ontology representing access control policies as well as a commonly adopted policy language, the PDPs of different security domains shall be able to interoperate with each other with the ability to interpret of credentials and values in the input (i.e. information provided by the service requester). This removes the impractical restriction of all PEP/PDP in the network using an identical vocabulary to describe the conceptual model of their respective security domains.

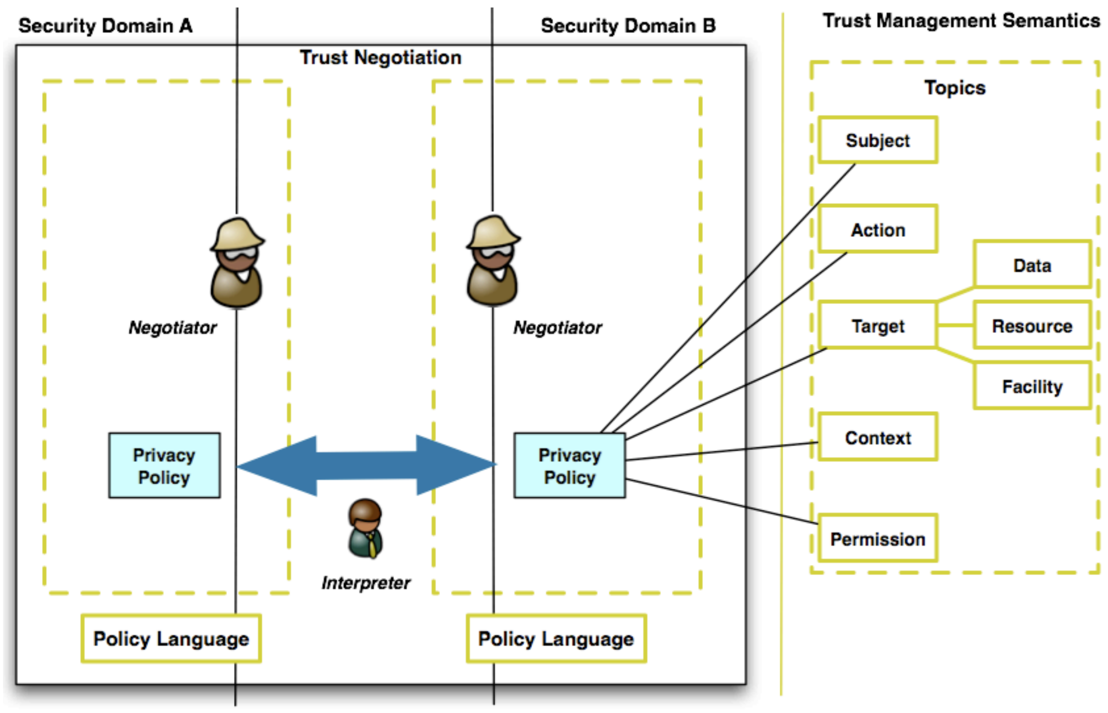


Figure 3.2: Cross-domain Policy Interoperability.

Since PDPs are distributed across different participants, PDPs are likely to use decision mechanisms based on different policy languages. In this case, an ontology can be used to bridge the different policy languages by using of ontological annotations. This Translation service based on the ontology presupposes the presence of annotations/mappings between the local vocabulary used by a specific PDP or of a commonly agreed and shared vocabulary; i.e. the security policy ontology. In Figure 3.2, the interpreter translates the privacy policy from Security Domain A to a policy suitable for Security Domain B by focusing on the concepts described in Section 2.1.1 (i.e.subject, action and target). In the employability scenario, this service is used to calculate the matching of security concepts extracted from access requests (made for example by a placement provider) and local authorization policies (set up by a student).

3.4 Ontology-based Interoperation Service in TAS^3

The security policy ontology provides a common 'conceptual model' for different security domains. This "*common meaning*" is modelled as entities and relations in the form of lexons. The previous sections identified three scenarios for the use of this ontology in TAS^3:

- Use of ontology in policy management
- Use of ontology in policy interpretation
- Use of ontology in policy in translation

The ontology can be adopted directly as the local conceptual model of a specific security domain in the policy specification. Where different conceptual models exist, different policy languages are used and different implementations exist, the information exchanged across security domains can be annotated by the ontology. For example, the credentials of an agent in Security Domain A can be sent to a PEP in Security Domain B together with their ontological annotations. The PDP in Security Domain B will then interpret the attribute-value pairs of the policies based on their ontological annotations. As a result, the PDP will be able to assess whether the information provided is of the expected type expected or synonymous or equivalent of conditions for the evaluation of a policy.

The Ontology-based Interoperation Service (OBIS [31]) was designed as a web service providing an interface to perform relation lookups between two terms originating from different security policies. The main method of the web service interface is summarized in the Figure 3.3.

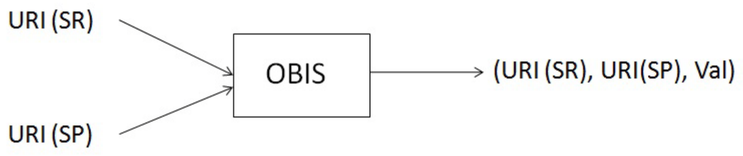


Figure 3.3: OBIS invocation method.

Where URI(SR), URI(SP) are two input values, representing the URIs of the two terms to be matched, and Val represents the returned value, i.e. the level of dominance between URI(SR) and URI(SP). There are six possible return values:

- -3: "I don't know SP"
- -2: "I don't know SR"
- -1: "SR is not related to SP"
- 0: "SR is less general than (is dominated by) SP"
- 1: "SR is equivalent to SP"
- 2: "SR is more general (dominates) SP".

OBIS is invocated during the authorization process, to calculate the dominance relationship between two security concepts inferred from an access request term and a local authorization policy term. Based on the value returned by OBIS, the access request is either granted or denied.

During the last year of the project, the OBIS service has been enhanced with an extension, the OBIS Domain Mapper [32], which maps attributes (e.g. XACML name value pairs) from one security domain to another (here from a "local" repository into the vocabulary used as reference in the distributed environment).

The mapping of security attributes is done using the Ω-RIDL Mapping Generator tool [33,34]. Ω-RIDL takes in input an XML file representing the access decision request and an ontology file (lexons and commitments) representing the access control policy ontology and returns a Ω-RIDL mapping file which maps the security attributes to concepts in the ontology, as illustrated in Figure 3.3.

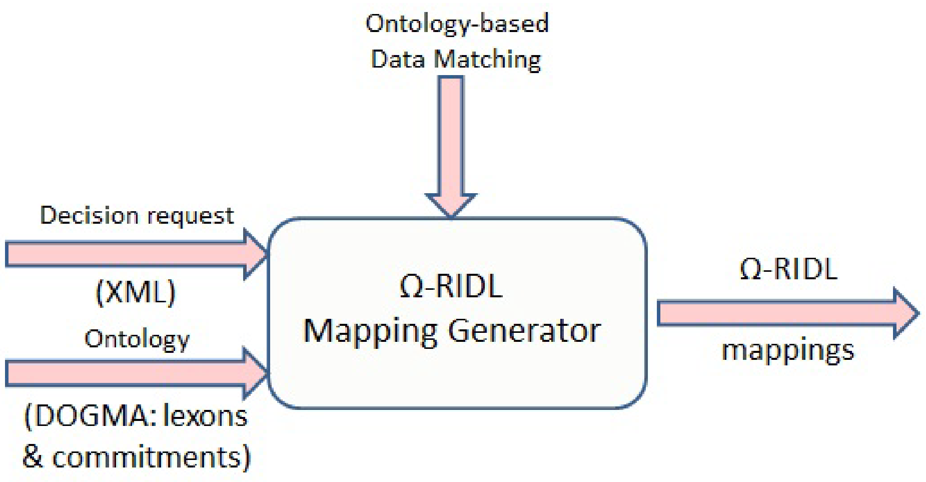


Figure 3.3: Ω-RIDL Mapping Generator architecture.

The Ω-RIDL mappings are obtained by applying ontology-based data matching strategies at (1) string level (fuzzy literal similarity, e.g. JaroWinkler); (2) lexical level (synonymous similarity, e.g. based on WordNet[2]) and (3) ontology (lexon graph) level (semantic similarity) in this order (we refer to [35] for details on ontology-based data matching strategies).
Ω-RIDL is designed as a web service which is called by the OBIS Domain Mapper service in order to infer the mapping of security attributes between two domains. OBIS Domain Mapper sends a bag of security attributes (name-value pairs representing the subject, resource and action) corresponding to a Security Domain Dom1 in input to Ω-RIDL which performs semantic inference and ontology-based data matching operations and returns another bag of attributes corresponding to another Security Domain, Dom2, as shown in Figure 3.4. Previous to calling Ω-RIDL, OBIS performs an explicit translation (mapping) from the local terminology (Dom1, Dom2) to the core ontology (lexon graph), based on the user-defined dictionaries. Then Ω-RIDL performs semantic inference operations on the ontology graph in order to infer the mappings from Dom1 to Dom2. For the moment being we only consider one-to-one mapping of attribute-value pairs. The one-to-many and many-to-one mappings of attribute-value pairs are ongoing work.

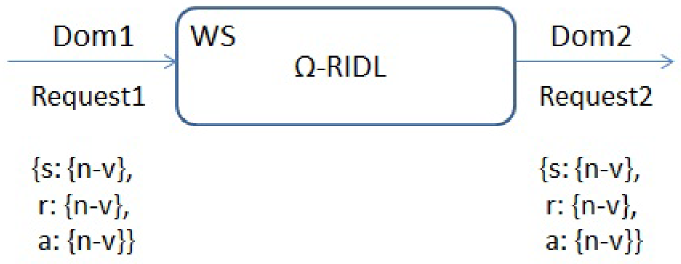


Figure 3.4: Ω-RIDL invocation.

[2] http://wordnet.princeton.edu/

The security policy ontology, unlike any other common APIs and processes in TAS^3, is a contract of conformance for collaboration and information exchanges. It is the pivot, which enables service providers and consumers from an open-ended list of security domains to agree in order to interact with each other, while allowing them to disagree on certain elements to satisfy local requirements of their respective security domains.

4 Security Policy Core Elements

A *security policy* imposes rules to control or guide behavior in an attempt to reduce risk to resources by accidental (e.g. natural disasters) or deliberate actions (e.g. viruses) [18]. More specifically, a security policy is a *countermeasure* defined to address *vulnerabilities* identified during the security *risk analysis activity* (Table 4-1). From now on, concepts with the prefix "DUO:" are described in the Descriptive Upper Ontology (see Section 6.1 in D2.2), while those with the "UCO:" are defined in the TAS[3] Upper Common Ontology (see Section 8 in D2.2).

Table 4-1: Security Policy

Context	Head term	Role	Co-role	Tail term
TAS3Policy	SecurityPolicy	is-a	subsumes	UCO:Countermeasure
UCO	UCO:Countermeasure	addresses	addressed-by	UCO:Vulnerability
UCO	UCO:RiskAnalysis	identifies	identified-by	UCO:Vulnerability
TAS3Policy	SecurityPolicy	defined-by	defines	UCO:RiskTreatment

A security policy includes the *conditions* to be respected to perform a certain *action* on a *target* in a given context (Table 4-2). For example, an audit policy could state that if the security level is orange (i.e. the condition) then all user access must be logged (i.e. the action) in an audit log (i.e. the target). Similarly, an authentication policy could state that if a user provides the correct username and password (i.e. condition), then the user is logged in (i.e. the action) to the system (i.e. the target). The *effect* of a security policy provides a statement committing the system to enact the action articulated in the policy. For example, the effect of an access control policy would be to grant or deny access to a target, while the effect of an audit policy would be to commit the system to add an entry to the audit log. Note that every security policy is given an *identifier* (e.g. URN) to uniquely refer to it within an information system as well as a natural language *description* of what the policy actually does. Note also that recording its *author* provides the provenance[3] of a policy.

Table 4-2: Lexons for the `SecurityPolicy` concept

Context	Head term	role	co-role	Tail term
TAS3Policy	SecurityPolicy	specifies	specified-by	Condition
TAS3Policy	SecurityPolicy	controls	controlled-by	Action
TAS3Policy	SecurityPolicy	Has	of	Target
TAS3Policy	SecurityPolicy	Has	of	Effect
TAS3Policy	SecurityPolicy	Has	of	Identifier
TAS3Policy	SecurityPolicy	Has	of	Description
TAS3Policy	SecurityPolicy	written-by	writes	Author
TAS3Policy	SecurityPolicy	enforces	enforced-by	Confidentiality
TAS3Policy	SecurityPolicy	enforces	enforced-by	Integrity
TAS3Policy	SecurityPolicy	enforces	enforced-by	Availability
TAS3Policy	SecurityPolicy	enforces	enforced-by	Traceability

[3] Provenance means the origin, or the source of something.

A security policy aims to enforce four goals; namely *confidentiality*, *integrity*, *availability*, and *traceability* [19]. Confidentiality provides protection against unauthorised access to an information system, while integrity ensures the correctness and completeness of information and the correct functioning of an information system. Traceability provides verifiability in terms of all actions performed in an information system. Finally, availability is concerned with ensuring that the information system is accessible to its users when it should be.

The rest of this section focuses on describing the concepts of *Condition*, *Action*, and *Target*.

4.1 Condition

A *condition* determines whether or not an action (or a set of actions) should be performed. In other words, conditions define the requirements (i.e. the *environment*) for an action to be executed. For example, an access control policy could state that students can only enter the library if it is a weekday and that the time is between 9am and 5pm.

Table 4-3: Lexons representing conditions.

Context	Head term	role	co-role	Tail term
TAS3Policy	Condition	Has	of	Environment
TAS3Policy	Environment	equivalent	equivalent	DUO:Descriptor
DUO	DUO:Descriptor	subsumes	is-a	DUO:Time
DUO	DUO:Descriptor	subsumes	is-a	DUO:Location
DUO	DUO:Descriptor	subsumes	is-a	DUO:Attribute

The *environment* provides a *descriptor* to express the condition of a security policy. In the example above, the *location* would be the library. A complete definition of descriptor is found in Section 6.1.3 in D2.2.

4.2 Subject

A *subject* is an *agent* performing an *action* on a *target* (*resource*) (Table 4-4.4). A subject can access a target according to the access *rights* granted to the subject in the security policy. Examples of subjects (data subjects) are a patient in the eHealth domain, an employee or a student in the employability domain, a home organisation or an id provider in both domains.

Table 4-4.4: Lexons for the `Subject` concept.

Context	Head term	Role	co-role	Tail term
DUO	DUO:Event	is-a	Subsumes	DUO:Predicate
DUO	DUO:Event	performed-by	Performs	DUO:Agent
TAS3Policy	Subject	is-a	Subsumes	DUO:Agent
TAS3Policy	Action	is-a	Subsumes	DUO:Event
TAS3Policy	Target	Equivalent	Equivalent	DUO:Entity
TAS3Policy	Subject	Performs	preformed-by	Action
TAS3Policy	Subject	Has	Of	Right
TAS3Policy	Subject	Accesses	Accessed-by	Target

4.3 Action

An *action* is an *event* that an *agent* seeks to perform (Table 4-5). In an audit policy, for example, the action could state that access to any resource must be recorded by the system. Note that an action can either be a simple operation, or a bundle of complex operations provided as an integrated set. Note also that an action is given a list of *parameters* (i.e. attributes) defining how the action must be performed.

Table 4-5: Lexons for the `Action` concept.

Context	Head term	Role	co-role	Tail term
DUO	DUO:Event	is-a	Subsumes	DUO:Predicate
DUO	DUO:Event	performed-by	Performs	DUO:Agent
TAS3Policy	Action	is-a	Subsumes	DUO:Event
TAS3Policy	Action	controlled-by	Controls	SecurityPolicy
TAS3Policy	Action	Has	Of	Parameter
TAS3Policy	Parameter	is-a	Subsumes	DUO:Attribute

4.4 Target

A *target* is an *entity* on which an *action* is performed and is described by *descriptors* (Table 4-6Table 4-6). A target can therefore be anything e.g. a resource, data, an agent, or a permission. For example, in TAS^3, security policies (in the form of privacy policies) are imposed to protect inappropriate behaviour on personal data (the target). Personal data refers to any information relating to an identified or identifiable natural person (see art. 2 (a) in EU directive 95/46/EC[4]). Note that an identifiable person is one that can be directly or indirectly identified by reference a identification number or by one or more characteristics specific to his physical, physiological, economic, mental, cultural or social identity. Examples of resources containing personal data are a patient record in the eHealth domain, or a CV in the employability domain.

Table 4-6: Lexons for the `Target` concept.

Context	Head term	role	co-role	Tail term
TAS3Policy	Target	equivalent	equivalent	DUO:Entity
DUO	DUO:Entity	subsumes	is-a	DUO:Agent
DUO	DUO:Entity	subsumes	is-a	DUO:Object
DUO	DUO:Entity	described-by	describes	DUO:Descriptor
TAS3Policy	Resource	is-a	subsumes	DUO:Object
TAS3Policy	Resource	subsumes	is-a	PersonalData
TAS3Policy	PersonalData	identifies	identified-by	Person
TAS3Policy	Person	is-a	subsumes	DUO:Agent

[4] http://ec.europa.eu/justice_home/fsj/privacy/index_en.htm

5 Type of Policies

A *security policy* imposes rules on the behaviour of users of a system to ensure that resources are used properly (Table 4-2). We have identified different types of policies in TAS^3; namely *authentication*, *credential validation, access control*, *obligation*, *privacy*, *audit* and *delegation* policies. The rest of this section represents the conceptualisation of these types of policies.

5.1 Authentication Policy

An *authentication* policy determines the process of verifying the claimed *identity* (i.e. a set of attributes) of an agent. It specifies how an entity's claim of a particular identity shall be verified. For example, an authentication policy could state that if a user provides the correct password (i.e. condition), then she is allowed to log in (i.e. the action) to the system (i.e. the target).

Table 5-1: Lexons representing authentication policies.

Context	Head term	role	co-role	Tail term
TAS3Policy	AuthenticationPolicy	is-a	subsumes	SecurityPolicy
TAS3Policy	AuthenticationPolicy	controls	controlled-by	Identity
TAS3Policy	Identity	of	has	DUO:Agent
TAS3Policy	AuthenticationPolicy	has	of	AuthnMechanism
TAS3Policy	AuthnMechanism	is-a	subsumes	UCO:Countermeasure
TAS3Policy	AuthenticationPolicy	has	of	RegistrationProcess
TAS3Policy	AuthenticationPolicy	has	of	LevelOfAssurance
TAS3Policy	LevelOfAssurance	has	of	Value
TAS3Policy	LevelOfAssurance	has	of	Type

An authentication policy has a level of assurance (LoA) [20], which provides a value (e.g. between 1 and 4 in NIST [21]) specifying the confidence in the asserted identity of an agent, which the service provider thinks it has (i.e. which is claimed by the agent). This level of assurance is not only impacted by the *authentication mechanism* (e.g. usernames and passwords vs. public key certificates and private keys), but also by the *registration process* that preceded the authentication. We can identify three types of LoA; namely *registration*, *authentication*, and *session* LoA. The registration LoA measures the strength of the registration process (i.e. documents required and procedure) occurring prior to any computer-based authentication taking place, while the authentication LoA measures the strength of the authentication mechanism itself. Finally, the session LoA is equivalent to the authentication mechanism chosen by a user for logging in to a session. However, no Session LoA can be higher than the Registration LoA when attributes are going to be asserted, since it is the latter that provided the strength of authentication of the user to which the identity attributes are now attached. If no attributes are to be asserted, then the Authentication LoA can be used as in D7.1 [22].

5.2 Credential Validation Policy

A *credential validation* policy defines the mechanisms for the evaluation of credentials (see Section 6.4.2 in D7.1) and mapping their *assigned attributes* into internally recognised *security attributes*. Note that these mechanisms are countermeasures enabling external credentials to be validated according to internal policies, and then mapped into internally recognized security attributes suitable for passing to an access control PDP. The policy also provides an *authoritative source* of the issued credentials.

Table 5-2: Lexons representing credential validation policies.

Context	Head term	role	co-role	Tail term
TAS3Policy	CredentialValidationPolicy	is-a	subsumes	SecurityPolicy
TAS3Policy	CredentialValidationPolicy	has	of	AuthoritativeSource
TAS3Policy	AuthoritativeSource	is-a	subsumes	DUO:Agent
TAS3Policy	AuthoritativeSource	issues	issued-by	AssignedAttribute
TAS3Policy	AssignedAttribute	is-a	subsumes	DUO:Attribute
TAS3Policy	AssignedAttribute	assigned-to	has	DUO:Agent
TAS3Policy	AssignedAttribute	maps-to	mapped-to	SecurityAttribute
TAS3Policy	SecurityAttribute	is-a	subsumes	DUO:Attribute
TAS3Policy	SecurityAttribute	assigned-to	has	DUO:Agent
TAS3Policy	CredentialValidationPolicy	has	of	SecurityAttribute
TAS3Policy	CredentialValidationPolicy	has	of	Validity
TAS3Policy	Validity	expressed-by	expresses	DUO:Time

A *credential* is issued by a trusted *authority*, which is responsible for determining who (i.e. *agent*) should be assigned which *assigned attributes* (Table 5.3). The authority can be trusted in different security domains, and the *signature* is present to allow the credential to be transferred between and validated in different domains. Note that a credential authority can be different from the authoritative source in a credential validation policy especially when credentials are delegated to other agents. Note also that an authority will associate a *validity* period and a policy, called an *issuer policy*, with the credential it issues, in order to restrict the use of the credential.

Table 5-3: Lexons representing credentials.

Context	Head term	role	co-role	Tail term
TAS3Policy	Credential	issued-by	issues	AuthoritativeSource
TAS3Policy	Credential	has	of	Signature
TAS3Policy	Credential	has	of	Validity
TAS3Policy	Credential	has	of	DUO:Agent
TAS3Policy	Credential	has	of	AssignedAttribute
TAS3Policy	Credential	has	of	IssuerPolicy

5.3 Access Control Policy

An *access control* policy determines who is authorized to *access* a *target* (Table 5-4). A *permission* grants the right to an agent to perform an *action* on a *target.* Once an agent has been granted access to a target, she can perform one or more *actions* on that target. In the eHealth domain, for example, a doctor may be able to consult (i.e. the action) the content of his patient's record (i.e. the target) once she has gained access to it. In general, a policy enforcement point (PEP) enforces the action of an access control policy after the evaluation of its condition by a policy decision point (PDP).

Table 5-4: Lexons representing access control policies.

Context	Head term	role	co-role	Tail term
TAS3Policy	AccessControlPolicy	is-a	subsumes	SecurityPolicy
TAS3Policy	AccessControlPolicy	enforced-by	enforces	PEP
TAS3Policy	AccessControlPolicy	evaluated-by	evaluates	PDP
TAS3Policy	AccessControlPolicy	controls	controlled-by	Access
TAS3Policy	Access	is-a	subsumes	Action
TAS3Policy	Access	on	under	Target
TAS3Policy	Access	enables	enabled-by	Action
TAS3Policy	AccessControlPolicy	has	of	DUO:Agent
TAS3Policy	DUO:Agent	has	of	Permission
TAS3Policy	Permission	grants	granted-by	Action
TAS3Policy	AccessControlPolicy	subsumes	is-a	DACPolicy
TAS3Policy	AccessControlPolicy	subsumes	is-a	MACPolicy
TAS3Policy	AccessControlPolicy	subsumes	is-a	ABACPolicy

Samarati and De Capitani di Vimercati [23] differentiate between three types of access control policies; namely *discretionary* (DAC), *mandatory* (MAC) and *attribute-based* (ABAC) access control. DAC policies enforce access control based on the identity of the service requester and define rules to control who can (or cannot) perform an action on a target, while MAC policies are based on mandated regulations determined by a central authority which assign clearances to users. Finally, ABAC policies control accesses based on the security attributes that users have been assigned within a system. RBAC policies are a subclass of ABAC policies. Note that in TAS³ access control is implemented through ABAC policies.

Table 5-5: Lexons representing ABAC policies.

Context	Head term	role	co-role	Tail term
TAS3Policy	ABACPolicy	is-a	subsumes	AccessControlPolicy
TAS3Policy	Target	has	of	DUO:Attribute
TAS3Policy	DUO:Agent	has	of	SecurityAttribute
TAS3Policy	SecurityAttribute	is-a	subsumes	DUO:Attribute
TAS3Policy	SecurityAttribute	grants	granted-by	Permission
TAS3Policy	Permission	controls	controlled-by	Action
TAS3Policy	SecurityAttribute	has	of	Validity

In the ABAC model, a *security attribute* grants *permissions* to an *agent* to perform one or more *actions* on one or more *targets* (Table 5-5). In a network environment, a security attribute may describe any property of the agent, such as its network address or its identity (e.g. passport number, login ID, or email address). These security attributes are usually assigned to an entity by an *authoritative source.*

In Role-Based Access Control (RBAC) systems [24], a *security role* is associated with a set of *permissions*, where a permission is the right to perform a particular *action* on a particular *target*. The TAS^3 model supports hierarchical RBAC in which roles may be organized in a partial hierarchy, with some being *superior* to others. A superior role inherits all the permissions allocated to its subordinate roles. For example, if the role Staff is subordinate to Doctor, then the Doctor role will inherit the permissions allocated to the Staff role. Note that the set of permissions associated with the Doctor role will not only include those associated with Staff, but will also include permissions specific to doctors (e.g. add content to patient records). Sandhu and Samarati [25] introduce the concept of Separation of Duty (SoD) as the principle to ensure that no user should is given sufficient permission to misuse the system on their own.

Table 5-6: Lexons representing RBAC policies

Context	Head term	role	co-role	Tail term
TAS3Policy	RBACPolicy	is-a	subsumes	ABACPolicy
TAS3Policy	DUO:Agent	has	assigned-to	SecurityRole
TAS3Policy	SecurityRole	has	of	Permission
TAS3Policy	Permission	grants	granted-by	Action
TAS3Policy	Action	on	under	Target
TAS3Policy	SecurityRole	is-a	subsumes	SecurityAttribute
TAS3Policy	SecurityRole	has	of	SuperiorRole
TAS3Policy	SuperiorRole	is-a	of	SecurityRole
TAS3Policy	SuperiorRole	inherits	inherited-by	Permission
TAS3Policy	RBACPolicy	has	of	SeparationOfDuty

5.4 Obligation Policy

An *obligation policy* determines the *obligations* that are to be performed by the host system when a given *event* occurs providing the associated *condition* is fulfilled. For example, an obligation policy could state that an e-mail notification has to be sent every time a request to access a particular resource (the event) has been denied (the condition). The obligation contains the *action* that is to be performed, and the *fallback* obligation that is to be performed if the original obligation fails.

Table 5-7: Lexons representing obligation policies.

Context	Head term	role	co-role	Tail term
TAS3Policy	ObligationPolicy	is-a	subsumes	SecurityPolicy
TAS3Policy	ObligationPolicy	triggered-by	triggers	DUO:Event
TAS3Policy	ObligationPolicy	has	of	Obligation
TAS3Policy	Obligation	performs	performed-by	Action
TAS3Policy	Obligation	has	of	TemporalType
TAS3Policy	Obligation	has	of	Fallback
TAS3Policy	Fallback	is-a	subsumes	Obligation

Table 5-7 represents the conceptualisation of obligation policies. In Deliverable D7.1 [22], an *obligation* is defined to have *fallback* obligations, which are enforced in cases where the obligation fails to be enacted. More specifically, a fallback is a list of obligations to be applied whenever it is found that the original obligation cannot be enforced. Obligations have a temporal type that specifies at which stage of the enforcement of a user's action the obligation's *action* should be performed. In the deliverable, the authors define three different values for temporal types; namely "*before*", "*with*", and "*after*".

5.5 Privacy Policy

A *privacy* policy describes how a party retains, processes, discloses, and purges personal data. Note that these privacy policies can be inserted within authorization policies depending upon the specific authorization policy language used, but not all authorization policy languages can support all privacy specific rules.

A privacy policy addresses specific goals such as *accountability*, *accuracy*, *proportionality*, *transparency, legitimacy* and *finality* [26]. Accountability is concerned with ensuring that information systems processing personal data are responsible for complying with data protection regulations. Accuracy and proportionality extend the integrity goal by determining that the personal data gathered by a system is (i) relevant, (ii) adequate, (iii) accurate, (iv) up to date, and (v) not kept for longer than necessary, while proportionality focuses on assessing whether the personal data gathered is not excessive. Finally, transparency is concerned with ensuring that the data subject (or supervisory authority) is (or will be) sufficiently informed of the fact that an information system is accessing and processing is personal data. For instance, the Directive 95/46/EC states that the data controller should notify the appropriate supervisory authority before carrying out any processing operation.

Table 5-8 shows the lexons associated with a privacy policy. It states that privacy policies control the actions performed on *personal data.* For example, a privacy policy could define the amount of time a piece of personal data may be retained by an organisation. As such, a privacy policy defines the *retention period* of the data. Once this period has expired, the data should be deleted or anonymised by the information system. Note that the *data subject* must in principle give his consent for the data to be used in the first place (legitimacy). Note also that privacy policy have a *purpose* stating the context in which personal data may be manipulated (finality).

Table 5-8: Lexons representing privacy policies.

Context	Head term	Role	co-role	Tail term
TAS3Policy	PrivacyPolicy	is-a	subsumes	SecurityPolicy
TAS3Policy	PrivacyPolicy	controls	controlled-by	Action
TAS3Policy	Action	on	under	PersonalData
TAS3Policy	PrivacyPolicy	enforces	enforced-by	Accountability
TAS3Policy	PrivacyPolicy	enforces	enforced-by	Accuracy
TAS3Policy	PrivacyPolicy	enforces	enforced-by	Proportionality
TAS3Policy	PrivacyPolicy	enforces	enforced-by	Transparency
TAS3Policy	PrivacyPolicy	defines	defined-by	RetentionPeriod
TAS3Policy	RetentionPeriod	is-a	subsumes	Validity
TAS3Policy	PrivacyPolicy	specifies	specified-by	Consent
TAS3Policy	Consent	of	gives	DataSubject
TAS3Policy	DataSubject	is-a	subsumes	Subject

TAS3Policy	PrivacyPolicy	has	of	Purpose
TAS3Policy	PrivacyPolicy	written-by	write	DataController
TAS3Policy	DUO:Location	influences	influenced-by	DPL

A privacy policy is written by a *data controller*[5], regardless of whether or not such data are collected, stored, processed or disseminated by that party or by an agent in its behalf (see Section 8.2 in D2.2 [12]). Note that the content of privacy policies is heavily influenced by local *data protection law* (DPL).

5.6 Delegation Policy

A delegation policy defines whether an agent, called *delegator*, is able to delegate her *permissions* or *security roles* (Table 5-6) to another agent, called the *delegatee*. For example, a delegation policy could state that the manager (i.e. the delegator) can transfer her permissions or roles to a trusted employee (i.e. the delegatee) whilst she is on holiday.

Table 5-9: Lexons representing delegation policies.

Context	Head term	role	co-role	Tail term
TAS3Policy	DelegationPolicy	is-a	subsumes	SecurityPolicy
TAS3Policy	RoleDelegationPolicy	is-a	subsumes	DelegationPolicy
TAS3Policy	RoleDelegationPolicy	delegates	delegated-by	SecurityRole
TAS3Policy	PermDelegationPolicy	is-a	subsumes	DelegationPolicy
TAS3Policy	PermDelegationPolicy	delegates	delegated-by	Permission
TAS3Policy	AdminPermission	subsumes	is-a	Permission
TAS3Policy	AdminPermission	delegates	delegated-by	PolicyUpdate
TAS3Policy	PolicyUpdate	is-a	subsumes	Action
TAS3Policy	PolicyUpdate	updates	updated-by	SecurityPolicy
TAS3Policy	DelegationPolicy	has	of	Delegator
TAS3Policy	Delegator	is-a	subsumes	DUO:Agent
TAS3Policy	DelegationPolicy	has	of	Delegatee
TAS3Policy	Delegatee	is-a	subsumes	DUO:Agent
TAS3Policy	DelegationPolicy	has	of	Validity

In delegation policies, we can identify two specific types of policies; namely *role delegation policies* and *permission delegation policies*. Role delegation delegates a security role from the delegator to the delegate, whereas permission delegation delegates a permission. A *permission* allows the delegatee to perform a certain action on a certain target, whereas an *administrative permission* (i.e. AdminPermission in Table 5-9) allows the delegatee to update (action) security policies (target).

[5] A data controller is an agent that determines the ‘purpose and means’ of processing personal data.

5.7 Audit Policy

An *audit* policy states what *actions* should be recorded when they are performed (Table 5-10). For example, it could state that if the security level is orange then all user access are logged in an audit log. An audit log comprises a series of *audit records*. Both the audit log and each audit record have their *time of creation*. The *protection method* of an audit policy defines the method (e.g. encryption) to be used to protect the information recorded in an *audit log*.

Table 5-10: Lexons representing of audit policies.

Context	Head term	role	co-role	Tail term
TAS3Policy	AuditPolicy	is-a	subsumes	SecurityPolicy
TAS3Policy	AuditPolicy	identifies	identified-by	Action
TAS3Policy	AuditPolicy	has	of	ProtectionMethod
TAS3Policy	ProtectionMethod	protects	protected-by	AuditLog
TAS3Policy	AuditLog	is-a	subsumes	Resource
TAS3Policy	AuditLog	has	of	CreationTime
TAS3Policy	CreationTime	is-a	subsumes	DUO:Time
TAS3Policy	AuditLog	contains	included-in	AuditRecord
TAS3Policy	AuditRecord	records	recorded-in	Action
TAS3Policy	AuditRecord	has	of	CreationTime

6 Use of Ontology for Secure Business Processes

In TAS³, business processes are modelled using the Business Process Modeling Notation (BPMN). A business process is modelled as a workflow, orchestrating several web services and data in a coordinated flow, with cooperation among users (subjects) in different roles. The security-specific tasks (including the definition of security policies) are modelled in a descriptive way, as text annotations. That implies that the business modeller should model its process in the usual way and then specify the security constraints in the next step, as security annotations. Security annotations will be transformed into business process fragments and executed.

A service ontology has been created to represent the security constraints that apply to business processes. The security constraints ontology is used to assist the business modeller (see the approach described in Deliverable D3.1 [27]) in annotating the following BPMN elements: Roles (R), Activities (A), Groups of Activities (GoA), Pools and Lanes (P&L), Data (D), Message Flows (MF) and Events.

The taxonomy of BPMN elements is illustrated in Figure 6.1 Taxonomy of BPMN elements. Roles are assumed to be assigned to Pools and Lanes, while Activities are considered to be either tasks or sub-processes. Data subsumes data objects, data stores, data inputs and data outputs.

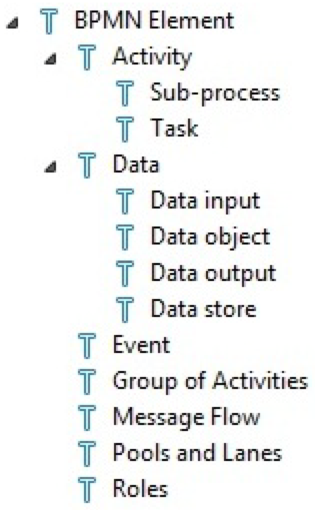


Figure 6.1 Taxonomy of BPMN elements.

The BPMN elements are annotated with security annotations, as illustrated in Figure 6.2 BPMN element and Security Annotation concepts. Each security annotation applies to at least one BPMN element.

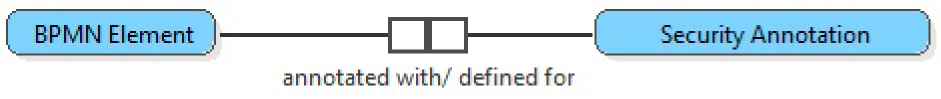


Figure 6.2 BPMN element and Security Annotation concepts.

The security annotations are classified into the following several security categories: Auditing, Authenticity, Authorization, Data and Messageflow Security, Delegation and User interaction (as shown in Figure 6.3).

Figure 6.4 illustrates the concept of security annotation. A security annotation defines a security policy. A security annotation is specified by an annotation term, followed by a list of parameters with their corresponding values: `<<Annotationterm:list(parameter="value")>>`. A security annotation can refer to more than one BPMN element.

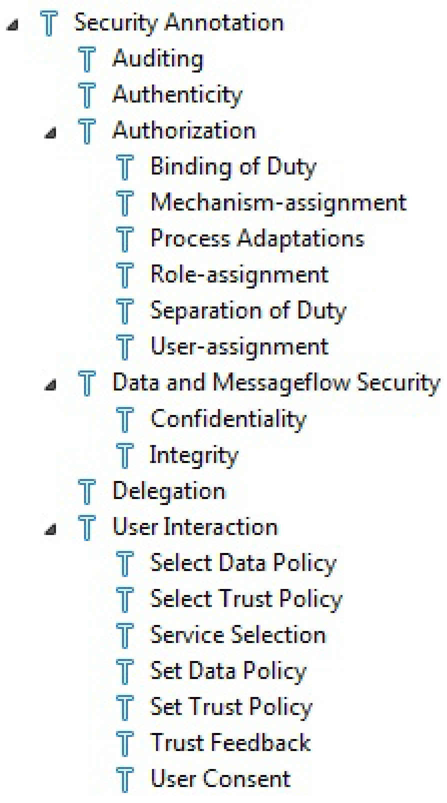


Figure 6.3 Taxonomy of Security Annotations.

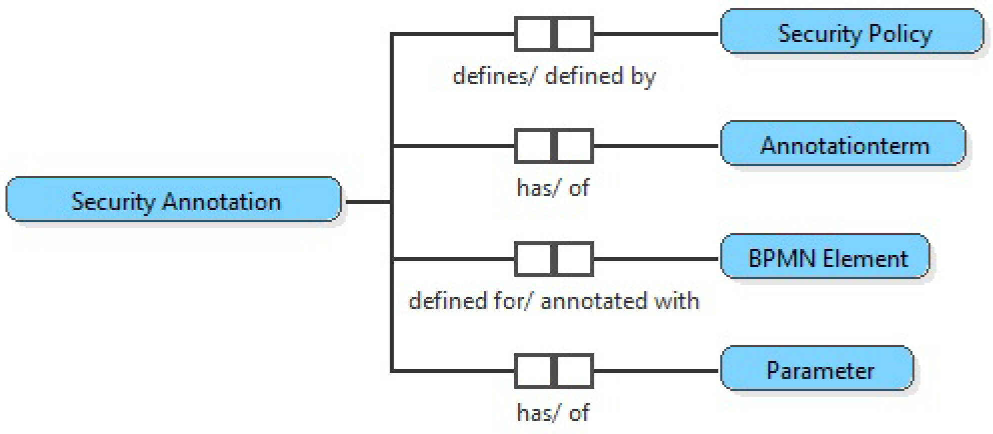


Figure 6.4 Representation of the Security Annotation Concept.

The parameters used for the specification of the security annotations are listed in Figure 6.5. The parameters can be specified as being optional – using the relation "has optional / optional of" – or compulsory (in which case the "has / of" relation is applied).

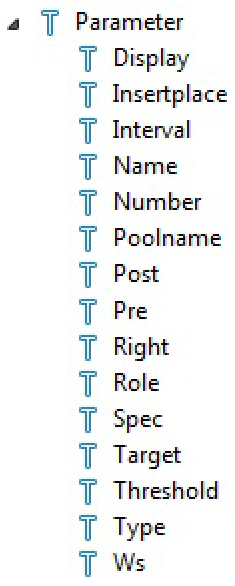


Figure 6.5 Taxonomy of Parameters used in the Specification of Security Annotations.

The Mechanism-assignment security annotation is exemplified with its corresponding specification elements (annotation term, parameters, and the BPMN elements it refers to), as shown in Figure 6.6. The Mechanism-assignment annotates the BPMN elements Group of Activities, Pools and Lanes, Roles and Activity. The compulsory parameters specifying this type of annotation are: Type and Name. Assignment represents the annotation term for the Mechanism-assignment security annotation.

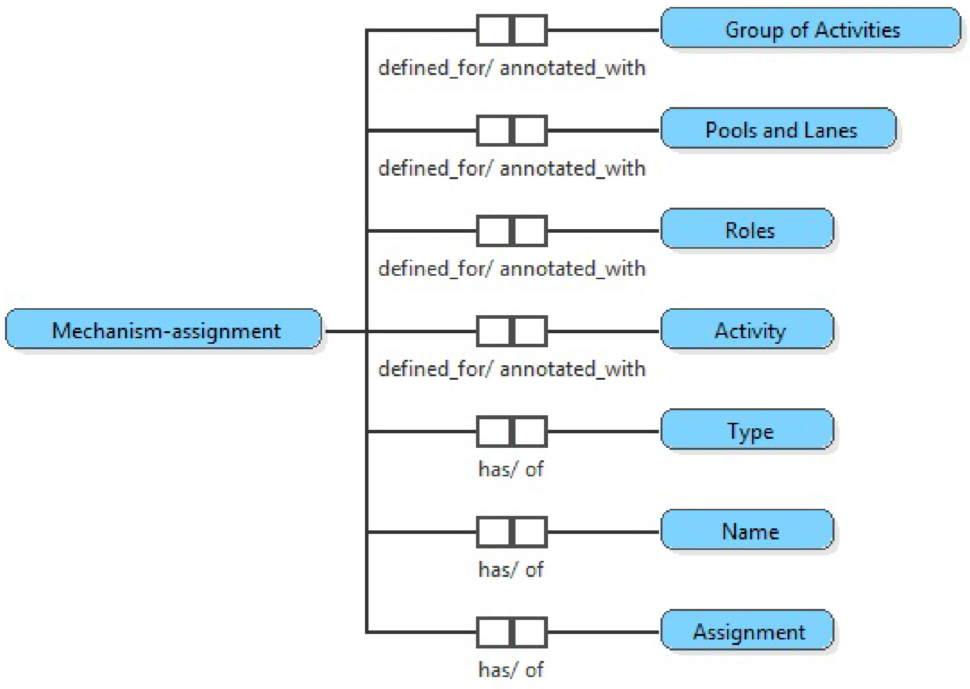

Mechanism-assignment
defined_for/ annotated_with
Group of Activities
defined_for/ annotated_with
Pools and Lanes
defined_for/ annotated_with
Roles
defined_for/ annotated_with
Activity
has/ of
Type
has/ of
Name
has/ of
Assignment

Figure 6.6 Representation of the Mechanism-assignment Security Annotation.

The security annotations with their corresponding annotation terms, parameters and BPMN elements are represented by the DOGMA ontology in Table 6-1.

Table 6-1: Ontology of Security Constraints

Head term	Role	Co-role	Tail-term
Security Anotation	subsumes	is-a	Authorization
Security Anotation	subsumes	is-a	Delegation
Security Anotation	subsumes	is-a	Data and Messageflow Security
Security Anotation	subsumes	is-a	Auditing
Security Anotation	subsumes	is-a	Authentication
Security Anotation	subsumes	is-a	User Involvement
BPMN Element	subsumes	is-a	Role
BPMN Element	subsumes	is-a	Activity
BPMN Element	subsumes	is-a	Group of Activities
BPMN Element	subsumes	is-a	Pools and Lanes
BPMN Element	subsumes	is-a	Data
BPMN Element	subsumes	is-a	Message Flow
BPMN Element	subsumes	is-a	Event
Role	assigned-to	assigned	Pools and Lanes
Activity	subsumes	is-a	Task
Activity	subsumes	is-a	Sub-process
Data	subsumes	is-a	Data Object
Data	subsumes	is-a	Data Store

Data	subsumes	is-a	Data Input
Data	subsumes	is-a	Data Output
Security Annotation	defined-for	annotated-with	BPMN Element
Security Annotation	has	of	Annotationterm
Security Annotation	has	of	Parameter
Parameter	has	of	Value
Authorization	subsumes	is-a	Role-assignment
Authorization	subsumes	is-a	Mechanism-assignment
Authorization	subsumes	is-a	User-Assignment
Authorization	subsumes	is-a	Separation of Duty
Authorization	subsumes	is-a	Binding of Duty
Authorization	subsumes	is-a	Process Adaptations
Data and Messageflow Security	subsumes	is-a	Confidentiality
Data and Messageflow Security	subsumes	is-a	Integrity
User Involvement	subsumes	is-a	Set Interaction Preferences
User Involvement	subsumes	is-a	Give Consent
User Involvement	subsumes	is-a	Service Selection
User Involvement	subsumes	is-a	Set Data Policy
User Involvement	subsumes	is-a	Select Data Access Policy
User Involvement	subsumes	is-a	Trust Feedback
User Involvement	subsumes	is-a	Set Trust Policy
User Involvement	subsumes	is-a	Select Trust Policy
Parameter	subsumes	is-a	Type
Parameter	subsumes	is-a	name
Parameter	subsumes	is-a	Role
Parameter	subsumes	is-a	number
Parameter	subsumes	is-a	threshold
Parameter	subsumes	is-a	spec
Parameter	subsumes	is-a	right
Parameter	subsumes	is-a	Pre
Parameter	subsumes	is-a	Post
Parameter	subsumes	is-a	target
Parameter	subsumes	is-a	interval
Parameter	subsumes	is-a	poolname
Parameter	subsumes	is-a	Ws
Parameter	subsumes	is-a	display
Parameter	subsumes	is-a	insertplace
Parameter	subsumes	is-a	object
Parameter	subsumes	is-a	owner
Parameter	subsumes	is-a	receiver
Parameter	subsumes	is-a	attribute
Parameter	subsumes	is-a	Idp
attribute	subsumes	is-a	name
attribute	subsumes	is-a	value

Parameter	subsumes	is-a	policy
Assignment	is-a	subsumes	Annotationterm
Role-assignment	has	of	Assignment
Role-assignment	has	of	Type
Role-assignment	has optional	optional of	name
Role-assignment	defined-for	annotated-with	Activity
Role-assignment	defined-for	annotated-with	Group of Activities
Role-assignment	defined-for	annotated-with	Pools and Lanes
Mechanism-assignment	has	of	Assignment
Mechanism-assignment	has	of	name
Mechanism-assignment	defined-for	annotated-with	Role
Mechanism-assignment	defined-for	annotated-with	Activity
Mechanism-assignment	defined-for	annotated-with	Group of Activities
Mechanism-assignment	defined-for	annotated-with	Pools and Lanes
User-assignment	has	of	Assignment
User-assignment	has	of	Type
User-assignment	has	of	name
User-assignment	defined-for	annotated-with	Activity
User-assignment	defined-for	annotated-with	Group of Activities
User-assignment	defined-for	annotated-with	Pools and Lanes
SoD	is-a	subsumes	Annotationterm
Separation of Duty	defined-for	annotated-with	Activity
Separation of Duty	defined-for	annotated-with	Group of Activities
Separation of Duty	defined-for	annotated-with	Pools and Lanes
Separation of Duty	has optional	optional of	Role
Separation of Duty	has	of	number
Separation of Duty	has optional	optional of	threshold
BoD	is-a	subsumes	Annotationterm
Binding of Duty	has	of	BoD
Binding of Duty	has optional	optional of	spec
Binding of Duty	defined-for	annotated-with	Activity
Binding of Duty	defined-for	annotated-with	Group of Activities
Binding of Duty	defined-for	annotated-with	Pools and Lanes
Adaptation	is-a	subsumes	Annotationterm
Process Adaptations	has	of	Adaptation
Process Adaptations	has	of	right
Process Adaptations	has	of	Role
Process Adaptations	has	of	Pre
Process Adaptations	has	of	Post
Process Adaptations	defined-for	annotated-with	Activity
Process Adaptations	defined-for	annotated-with	Group of Activities
Process Adaptations	defined-for	annotated-with	Pools and Lanes
Process Adaptations	defined-for	annotated-with	Data
Process Adaptations	defined-for	annotated-with	Message Flow

Process Adaptations	defined-for	annotated-with	Event
D-Delegation	is-a	subsumes	Annotationterm
T-Delegation	is-a	subsumes	Annotationterm
Delegation	has	of	D-Delegation
Delegation	has	of	T-Delegation
D-Delegation	has optional	optional of	right
D-Delegation	has	of	object
D-Delegation	has optional	optional of	spec
D-Delegation	has	of	interval
D-Delegation	has optional	optional of	owner
D-Delegation	has	of	receiver
D-Delegation	defined-for	annotated-with	Activity
D-Delegation	defined-for	annotated-with	Group of Activities
D-Delegation	defined-for	annotated-with	Pools and Lanes
D-Delegation	defined-for	annotated-with	Message Flow
T-Delegation	has optional	optional of	right
T-Delegation	has optional	optional of	object
T-Delegation	has optional	optional of	spec
T-Delegation	has optional	optional of	owner
T-Delegation	has	of	receiver
T-Delegation	defined-for	annotated-with	Activity
T-Delegation	defined-for	annotated-with	Group of Activities
T-Delegation	defined-for	annotated-with	Pools and Lanes
T-Delegation	defined-for	annotated-with	Message Flow
Conf	is-a	subsumes	Annotationterm
Confidentiality	has	of	Conf
Confidentiality	has	of	spec
Confidentiality	defined-for	annotated-with	Message Flow
Integr	is-a	subsumes	Annotationterm
Integrity	has	of	Integr
Integrity	defined-for	annotated-with	Data
Integrity	defined-for	annotated-with	Message Flow
Audit	is-a	subsumes	Annotationterm
Auditing	has	of	Audit
Auditing	has	of	policy
Auditing	defined-for	annotated-with	Activity
Auditing	defined-for	annotated-with	Group of Activities
Auditing	defined-for	annotated-with	Pools and Lanes
Auditing	defined-for	annotated-with	Data
Auditing	defined-for	annotated-with	Message Flow
Auditing	defined-for	annotated-with	Event
Authn	is-a	subsumes	Annotationterm
Authentication	has	of	Authn
Authentication	has	of	attribute

Authentication	has	of	Idp
Authentication	defined-for	annotated-with	Role
Authentication	defined-for	annotated-with	Pools and Lanes
UInvolve	is-a	subsumes	Annotationterm
Set Interaction Preferences	has	of	UInvolve
Set Interaction Preferences	has	of	Type
Set Interaction Preferences	has	of	display
Set Interaction Preferences	has optional	optional of	Role
Set Interaction Preferences	has optional	optional of	insertplace
Set Interaction Preferences	defined-for	annotated-with	Activity
Set Interaction Preferences	defined-for	annotated-with	Group of Activities
Set Interaction Preferences	defined-for	annotated-with	Event
Give Consent	has	of	UInvolve
Give Consent	has	of	Type
Give Consent	has optional	optional of	Role
Give Consent	has	of	target
Give Consent	has	of	display
Give Consent	has optional	optional of	insertplace
Give Consent	defined-for	annotated-with	Activity
Give Consent	defined-for	annotated-with	Group of Activities
Give Consent	defined-for	annotated-with	Event
Service Selection	has	of	UInvolve
Service Selection	has	of	Type
Service Selection	has optional	optional of	Role
Service Selection	has	of	display
Service Selection	has optional	optional of	insertplace
Service Selection	defined-for	annotated-with	Activity
Service Selection	defined-for	annotated-with	Group of Activities
Service Selection	defined-for	annotated-with	Event
Set Data Access Policy	has	of	UInvolve
Set Data Access Policy	has	of	Type
Set Data Access Policy	has optional	optional of	Role
Set Data Access Policy	has	of	name
Set Data Access Policy	has	of	target
Set Data Access Policy	has optional	optional of	insertplace
Set Data Access Policy	defined-for	annotated-with	Activity
Set Data Access Policy	defined-for	annotated-with	Group of Activities
Set Data Access Policy	defined-for	annotated-with	Event
Select Data Access Policy	has	of	UInvolve
Select Data Access Policy	has	of	Type
Select Data Access Policy	has optional	optional of	Role
Select Data Access Policy	has	of	target
Select Data Access Policy	has	of	display
Select Data Access Policy	has optional	optional of	insertplace

Select Data Access Policy	defined-for	annotated-with	Activity
Select Data Access Policy	defined-for	annotated-with	Group of Activities
Select Data Access Policy	defined-for	annotated-with	Event
Trust Feedback	has	of	UInvolve
Trust Feedback	has	of	Type
Trust Feedback	has optional	optional of	Role
Trust Feedback	has	of	display
Trust Feedback	has	of	insertplace
Trust Feedback	defined-for	annotated-with	Activity
Trust Feedback	defined-for	annotated-with	Group of Activities
Set Trust Policy	has	of	UInvolve
Set Trust Policy	has	of	Type
Set Trust Policy	has optional	optional of	Role
Set Trust Policy	has	of	name
Set Trust Policy	has optional	optional of	insertplace
Set Trust Policy	defined-for	annotated-with	Activity
Set Trust Policy	defined-for	annotated-with	Group of Activities
Set Trust Policy	defined-for	annotated-with	Event
Select Trust Policy	has	of	UInvolve
Select Trust Policy	has	of	Type
Select Trust Policy	has optional	optional of	Role
Select Trust Policy	has	of	display
Select Trust Policy	has optional	optional of	insertplace
Select Trust Policy	defined-for	annotated-with	Activity
Select Trust Policy	defined-for	annotated-with	Group of Activities
Select Trust Policy	defined-for	annotated-with	Event

The security annotations are specified using a tool for annotating secure business processes, which is introduced in D3.1. The annotator uses an ontology base specifying the structure of the security annotations and a knowledge base which captures the user (business modeller) knowledge. The knowledge base stores previously annotated security constraints, based on which it is able to assist the user with recommendations and statistics.

The annotator (called knowledge annotator, since it is using a knowledge base) is designed as a user-friendly system, intended to assist the process modeler during the specification of the security-specific constraints and aids learning from the process modeler by using a dedicated knowledge base. This is realized by capturing the process modelers' modeling intentions via a user-friendly interface (UI) and by presenting him/her with recommendations (as shown in Figure 6.7). The recommendations are determined by an ontology-based data matching operation between the user input, the security constraints ontology, user-defined dictionaries, Synsets from lexical databases (e.g. WordNet) and the collected security annotations retrieved from the knowledge base.

The annotator tool encapsulates several functions in a web service, which are supported by six architectural components: (1) the capturer – for capturing the user design intent; (2) the annotator – for annotating objects with concepts from the security ontology; (3) the indexer – for indexing elements for efficient retrieval; (4) the retriever – for retrieving information; (5) the comparator – for comparing user input with the knowledge base; and (6) the presenter – for presenting the user with recommendations and user query specification. For details on the design and implementation of the business process annotator, the reader can refer to Deliverable D3.1 and Deliverable D3.2.

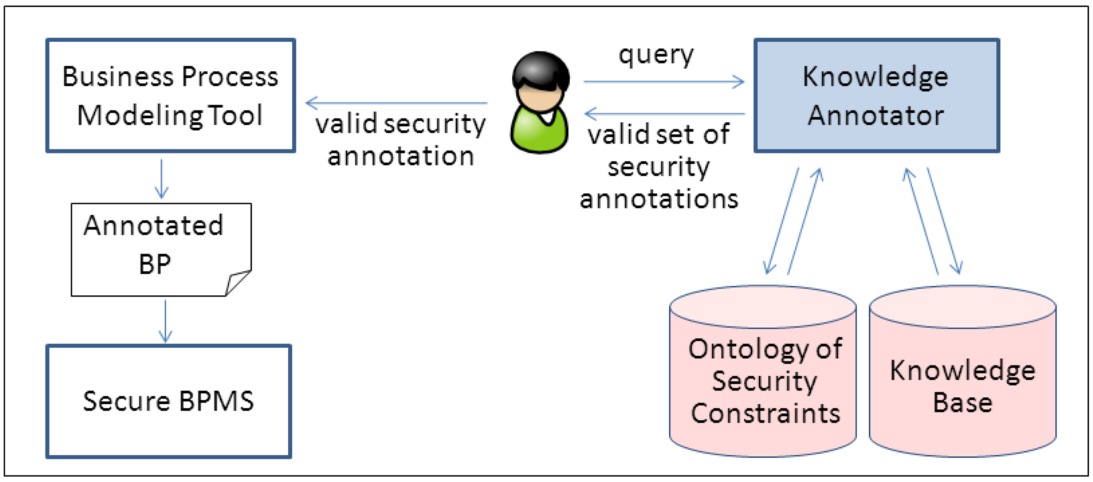

Business Process Modeling Tool
valid security annotation
query
valid set of security annotations
Knowledge Annotator
Annotated BP
Secure BPMS
Ontology of Security Constraints
Knowledge Base

Figure 6.7: User-system interactions for the annotation of security constraints.

The basic data object used by the annotator is represented by the Security Annotation Term – Element – Parameter -Value (STEPV) object. The STEPV object encapsulates the five entities needed to completely define a security annotation: the security constraint, the BPMN element being annotated, the parameters and their corresponding values. In case the value of the parameters is ignored, the object will be referred to as STEP object.

When the process modeller wants to define a security annotation, he/she fills in the fields corresponding to his/her design intent and knowledge (STEPV elements). This represents the input to the web service call. The system captures the input and analyses it against the ontology base and the knowledge base in order to make the best fitted recommendations to the process modeller. The STEPV elements returned are considered valid annotations according to the constraints defined in the commitment layer of the ontology.

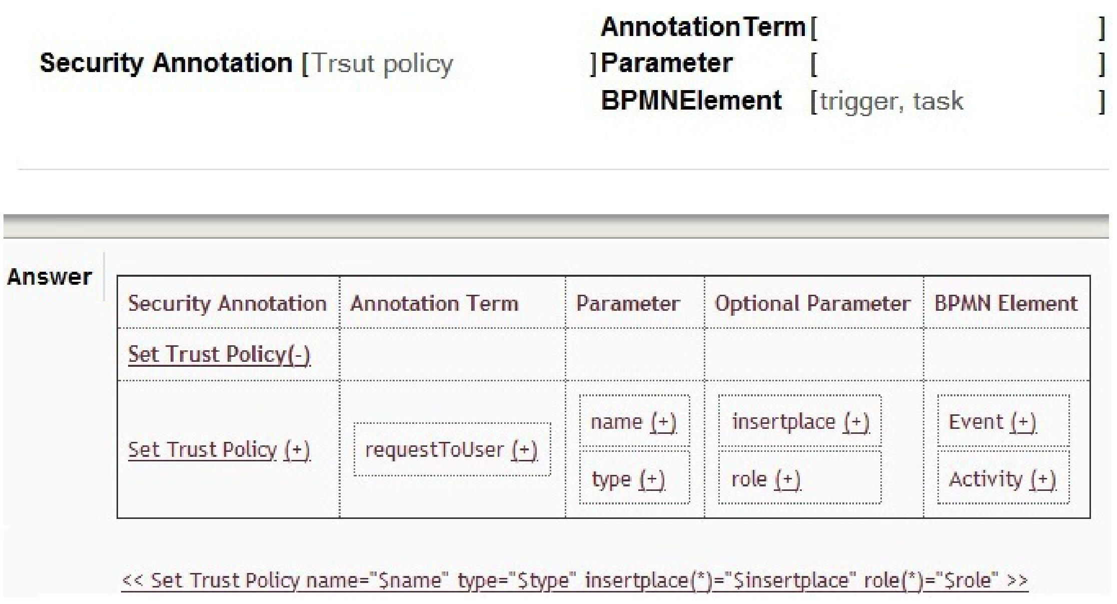
During the last period of the project, we have been focusing on the implementation of components (4) and (5) of the annotator:

(4) *The retriever* component retrieves similar fragments from the knowledge base (e.g., all existing security annotations which share at least one common element with the input object). The similarity measure can be defined according to the user needs. For example, the user could only be interested in STEPV objects with a particular value for the "name" parameter. The knowledge base contains semantic annotation instances (STEPVA objects) of the security constraints.

(5) *The comparator* component performs a matching operation in order to compare the process modeller's demand (input object) with the resulted elements retrieved from the knowledge base in the previous step.

We take the 'Set trust policy' annotation to illustrate the matching process. Via the UI, the process modeller can specify a desired STEPV object by indicating parts of it he/she knows. For this specific example, the user inputs his/her choices for the Security Annotation and the BPMNElement fields. The STEPV element is therefore specified only by two fields out of five. The value parameter is excluded from this example, for simplification. In this case, we are dealing with STEP objects. The system will interpret the user input by performing matching at different levels and will infer the correct most similar annotation. The user input and the system result are illustrated in Figure 6.8.

Let us now analyze how these results were inferred by the system. The process starts with the user specifying fields of the STEP object. He/she indicates 'Trsut policy' and 'trigger, task'. These values are resolved by performing matching at different levels. In case of 'Trsut policy', a matching operation is performed at string level and the correct string (concept in the ontology) is returned 'Set Trust Policy'. In case of 'trigger' and 'task', they are matched to the correct concepts 'event' and 'activity' respectively in the ontology, by performing matching at lexical level, using WordNet[6] and a user-defined dictionary to resolve the synonymy (see Table 6-2).



Security Annotation	[Trsut policy]	**AnnotationTerm**	[ ]
		Parameter	[ ]
		BPMNElement	[trigger, task]

Answer

Security Annotation	Annotation Term	Parameter	Optional Parameter	BPMN Element
Set Trust Policy(-)				
Set Trust Policy (+)	requestToUser (+)	name (+) type (+)	insertplace (+) role (+)	Event (+) Activity (+)

<< Set Trust Policy name="$name" type="$type" insertplace(*)="$insertplace" role(*)="$role" >>

Figure 6.8: User query for retrieving the Select Trust Policy security annotation and result.

Once the correct concepts in the ontology are found with their corresponding annotation sets (lexons), a matching step is performed by an ontology-based matching strategy at graph (ontology) level the most similar annotation is inferred:

[6] http://wordnet.princeton.edu/

```
<<     Set     Trust     Policy     name="$name"     type="$type"
insertplace(*)="$insertplace" role(*)="$role" >>
```

This annotation is presented to the user as a recommendation. The user can further refine his/her search by modifying fields of the retrieved STEP object.

Table 6-2: Lexical synonyms defined for concepts in the ontology

Synonyms	Concept in the ontology
'user' 'user interaction' 'interaction'	'requestToUser'
'select service' 'choice of service' 'choice'	'service selection'
'task' 'subprocess' 'flow element'	'activity'
'trigger' 'message'	'event'

7 Application of the Ontology in the TAS^3 Pilots

Up to present, the LCO has been used for security policy interoperability in the employability and in the health care scenarios of the TAS^3 project. This is detailed in the following sub sections.

7.1 Ontology for Employability

In the Nottingham demonstrator, the accurate and secure presentation and exchange of verified skills data and personal information is vital. For example, recruiters and prospective employers want to access verified data in a standardized format (e.g. HR-XML[7]) to facilitate comparisons and to match students with job profiles. Similarly, candidates want to retain control over how their personal information is accessed, processed and stored by third parties by setting their own security policies.

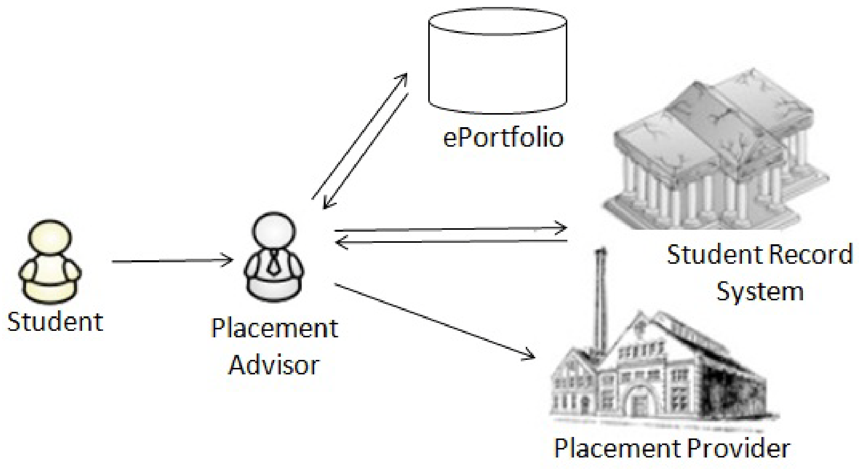


Figure 7.1 The employability scenario.

Figure 7.1 represents one of the employability scenarios in TAS^3. In this scenario, Alice is a second year student at a UK university and seeks a summer work placement. Alice contacts a placement service approved by the university to discuss the details of her application. Her placement advisor, called Bob, informs her that he first needs to verify that she is a registered student at the university. Once Bob has received the confirmation, he contacts Alice to get permission to access relevant information to match her to available placements. Alice is happy to share this information subject to this it not being shared to third parties without her approval. Based on this information, Bob identifies a number of placement providers that he believes have suitable placements for students like Alice. Alice wishes to be put forward for two placements and agrees that the placement advisor can act on her behalf to agree terms of a work placement and

[7] http://ns.hr-xml.org/schemas/org_hr-xml/3_0/

she consents to have relevant personal information to be disclosed to them. Bob forwards Alice's information to the placement providers for consideration.
In this scenario, several security and trust issues may be encountered, as follows:

- Does the student trust that the placement advisor is approved by the university?
- Can the student trust that only relevant personal information is used during the placement process?
- Can the student trust that the information provided to the placement provider is protected as per her privacy policy?

The placement advisor, placement providers, etc. use their own systems to store the information. How can all stakeholders be sure that personal information is secure between one placement and another?
When Alice issues a request for a placement or internship, the placement service first verifies with the Identity Provider (IdP) who the person is who she claims to be (see Figure 7.2).

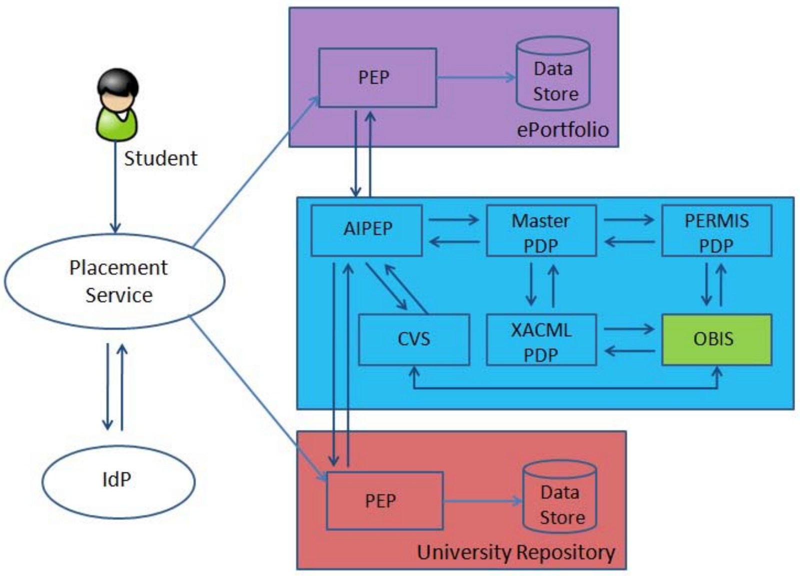


Figure 7.2: Architecture to enforce security and privacy policies.

After this, Bob needs to access relevant information about Alice from distributed data repositories (e.g. a CV from within the student's ePortfolio and university repositories). The information stored in these repositories and is protected by Alice's privacy policy as well as the keeper's own policy. If Alice's privacy policy states that only members of the university have permission to access her work, then Bob's request to access a subset of her work needs to be validated against this policy to see whether he can gain access or not. OBIS is needed to aid this access control decision.

The role of OBIS in the authorization architecture is to determine whether (1) a foreign subject dominates the subject in the authorization policy, (2) the requested action is dominated by the action in the access control policy and (3) the resource to be accessed by the subject is dominated by the resource in the policy. In addition OBIS may be called directly by the placement application to determine if information in Alice's documents fulfils the requirements of the various placement providers. Some examples from the employability scenario are illustrated in Table 7-1: OBIS Role in the Authorization Policy Interoperability.

Table 7-1: OBIS Role in the Authorization Policy Interoperability

OBIS Role	Security Concept Matching
To check whether Bob can access Alice's personal data by calculating the relation between the placement advisor (Bob) **role** in the request and the University Staff role in Alice's policy (to determine whether a foreign role dominates the local role in the credential validation policy).	**Role** matching for CVS. **Translator**: **SR**: Bob = delegate_right **SP**: University Staff = delegate_right **Path finder:** 1 "equivalence" ➔Attribute is valid
To determine whether the **resource** to be accessed by the service requester (e.g. Alice's previous employment experience) is more specific than the resource protected by the security policy (e.g. Alice's CV).	**Resource** matching for PDP. **Translator**: **SR**: Alice previous employment experience = work experience **SP**: Alice's CV = CV **Path finder:** 0 "less general" ➔Access granted
To determine whether Alice's programming competency (**resource** in the CV) is more specific than the requested competency ("the candidate should have good programming skills") for a vacant job.	**Resource** matching for application. **Translator**: **SR**: Java programming = competencies **SP**: the candidate should have good programming skills = competency profile **Path finder:** 0 "less general" ➔Satisfied

The security policy ontology used for this use case scenario is partially depicted in Figure 7.3, with the focus on the "ePortfolio". The lines represent the hierarchy of the considered resources, the most dominating concept being "Resource".

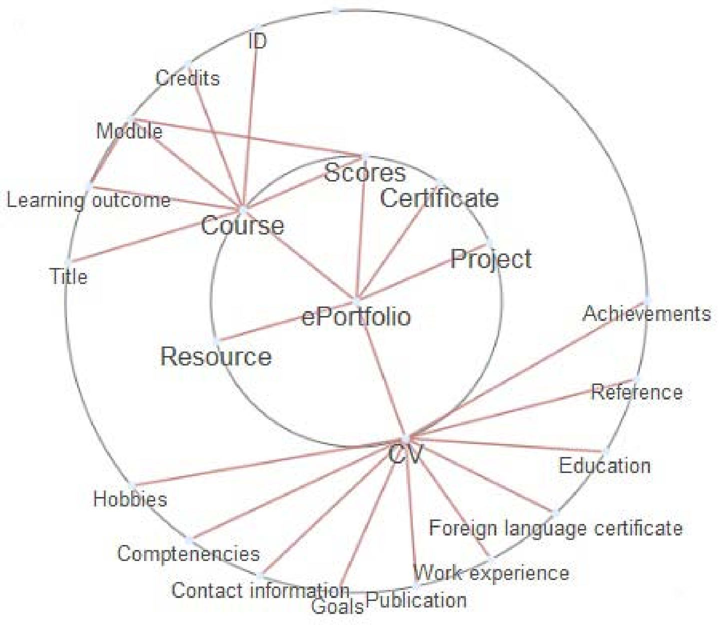


Figure 7.3: ePortfolio in the "Resource" Ontology.

7.2 Ontology for Health Care

TAS³ primarily puts people into control over their personal data in a service-oriented architecture. The e-Health pilot demonstrates how TAS³ accomplishes this objective in the highly regulated e-health environment, where user centric personal data management translates into:

(1) The possibility for patients to adjust the default e-health domain policies determined by legislation and ethical guidelines, so that their personal data is protected according to their personal preferences on data protection;
- For example: a patient should have the option not to disclose mental health related information from a replacement physician (out-of-office hours).
- However, in a highly regulated environment such as healthcare, personal freedom to hide or disclose health information has its boundaries (e.g. where hiding it could result in bad treatment or damage the treating healthcare professionals). These need to be taken into account.

(2) The possibility for data users to query patients for specific (extraordinary) access requests for data processing (e-consent).

For example: a patient could be invited to share existing data into a clinical study.

The demonstration environment (see Figure 7.4) was modeled according to the "distributed health repositories with central access" concept, which forms the basis of many e-health information sharing initiatives in the EU and the US. In particular, the use case was staged in a Belgian setting.

Central to the system is the Patient Information Location Service (PILS [36]) which is used by professionals (e.g. medical doctors, researchers) to find patient information in distributed repositories. Two types of repositories have been connected in this demonstrator: (1) hospital results servers and (2) summary record repositories, as illustrated in Figure 7.4.

Multiple Identity Providers (IdP) exist in the trial, all of which are authoritative with respect to unique user identities and to unique healthcare professional identifiers (similar to the actual situation in Belgium). Finally, there is a privacy management centre where patients can set access policies on their personal data.

In the privacy centre, patients can specify their personal privacy preferences which are then to be enforced over the health information sharing network for health professionals. The preferences set by the individual patients are translated into XACML policies which are loaded into a central Policy Decision Point (PDP). Service providers participating in the health network are required to forward access requests to that central PDP (if they involve resources covered by the central PDP policies).

Apart from the differences in authorization frameworks, different service providers use different security vocabularies (attribute-value pairs describing the subject, resource and action). In the demonstrator, the OBIS Domain Mapper service instances are responsible for translating local security vocabularies (name-value pairs) into the "domain" vocabulary, used in by the central PDP.

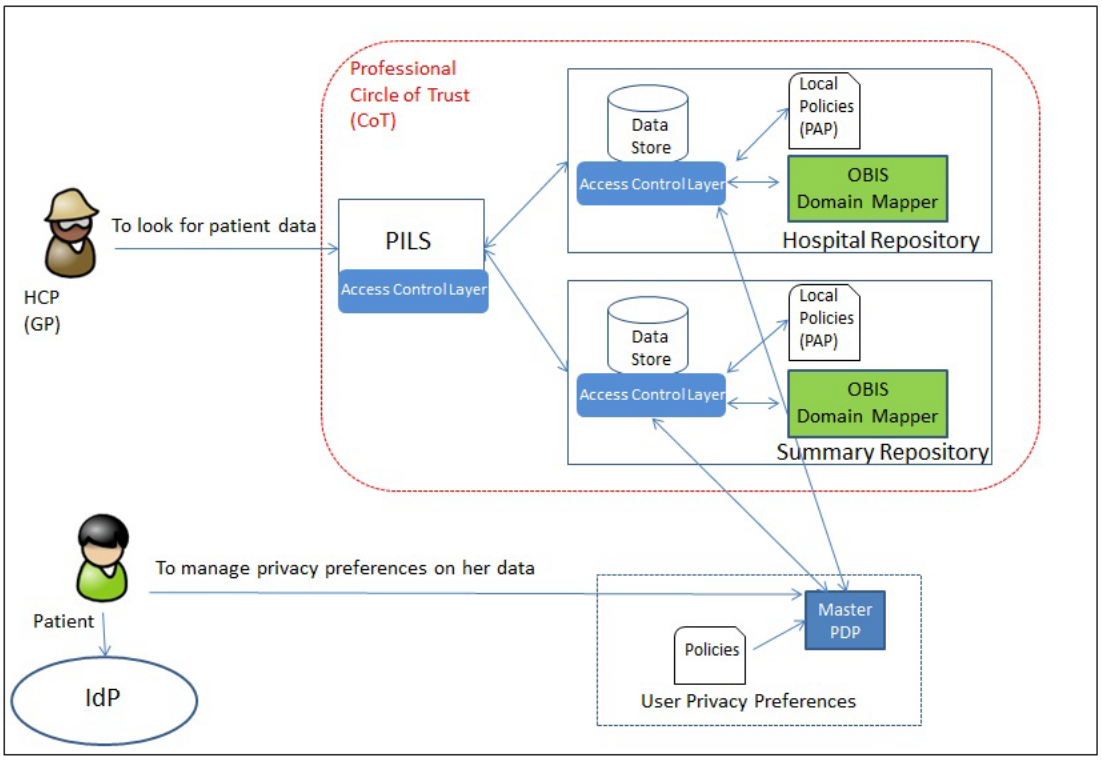


Figure 7.4: Health Care Scenario.

Essentially the OBIS DM service performs the following:

Given a representation of one resource in one repository which uses a set of property-value pairs, return a representation of the same resource based on the central repository vocabulary, using a set of property-value pairs that may be distinct. The service can handle multiple translations at once in order to minimize invocations.

Table 7-2 exemplifies request content of an access request to the central (Master) PDP in the first scenario.

Table 7-2: Example of content of an access request to the central PDP

Request			Decision
Subject	**Action**	**Resource**	
Subject_type = GP	Action_type = Read	Resource_type = labresult	Permit
Subject_type = GP	Action_type = Create	Resource_type = document	Permit
Any	Any	Any	Deny

Below we provide an example of a XACML policy rule that returns Permit for access requests that have value physician for attribute subject, value read for attribute action and value summary information for attribute resource (see example from Table 7-2).

```
<Target>
  <Resources>
    <Resource>
      <ResourceMatch
        MatchId="function:string-equal">
        <AttributeValue
          DataType="#string">summary_information
          </AttributeValue>
        <ResourceAttributeDesignator
          AttributeId="…information-class "
          DataType="#string" />
        </ResourceMatch>
      </Resource>
    </Resources>

  <Actions>
    <Action>
      <ActionMatch
        MatchId="function:string-equal">
        <AttributeValue
          DataType=="#string">read
        </AttributeValue>
      <ActionAttributeDesignator
        AttributeId="…action"
        DataType="#string" />
      </ActionMatch>
    </Action>
  </Actions>
</Target>

<Rule RuleId="SenderIsPhysician" Effect="Permit">
  <Description>Physicians can create new documents
  </Description>
  <Condition>
    <Apply>
      FunctionId="function:string-is-in"
      <AttributeValue>
        DataType="#string">physician
      </AttributeValue>
      <SubjectAttributeDesignator
        DataType="#string"
        AttributeId="hcp-type"/>
    </Apply>
  </Condition>
</Rule>
```

When a physician tries to view a `vaccination fiche` in one of the repositories through the health information sharing infrastructure, the following happens: Inside the contacted repository an access control request for a `read` action on a `vaccination fiche` (resource) is triggered (to eventually determine if the `vaccination fiche` can be shown to the physician). This request is to be evaluated by a local access control decision engine according to locally formulated policies (which do e.g. also deal with access rules for "locally originated" requests). However, for this type of access through the health information network, also the central PDP needs to be queried (access control decisions by different engines are eventually to be combined).

The security vocabulary used in the local repository (which is typically implementation specific) is not aligned with the more generic security vocabulary used in the wider health domain (used in the policies handled by the central PDP). The local access control request can thus not be evaluated as such by the central PDP.

A generic approach to translation of access requests from one domain vocabulary to another is provided by the OBIS Domain Mapper. This component translates attributes (e.g. XACML name value pairs) from one security domain to another (here from a "local" repository into the vocabulary used as reference in the distributed environment).

More specifically, in the described example, the OBIS Domain Mapper will look into the ontology hierarchy and constraints via the Ω -RIDL mappings and will infer that a `vaccination fiche` document (as known in the repository) classifies as `summary information` according to the central policy vocabulary.

Figure 7.5 illustrates the access policy ontology used in the e-Health scenario. The focus in the figure is on the "target" concept, showing its constituents hierarchically (see the circles). The semantic relations are of the type 'part-of' and 'is-a' (which grow or shrink a set), indicating the (security-specific) domination relation between the concepts. OBIS computes the domination relation between two concepts in the ontology using AND/OR graphs. The figure includes core concepts from the TAS³ security ontology (e.g. 'subject', 'action', 'resource') linked to application-specific concepts derived from the e-Health scenario (e.g. 'patient-id', 'hcp-type').

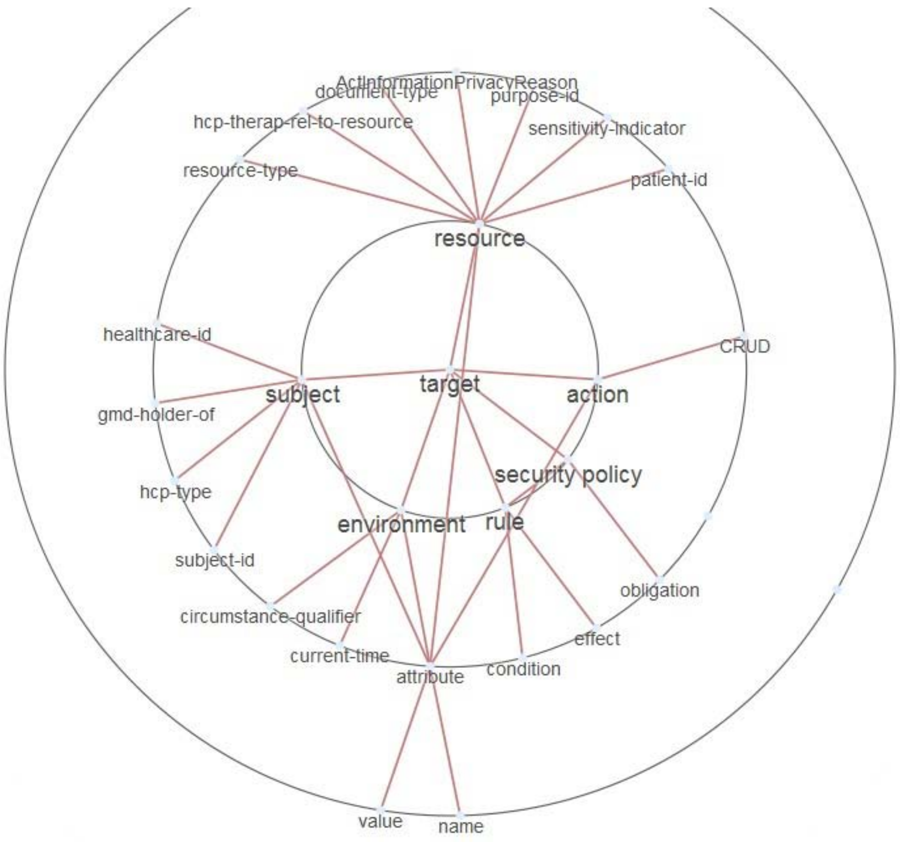


Figure 7.5: The "Target" concept in the access control policy ontology.

Every component in the above described scenario commits to this ontology. Every stakeholder organization (or department), must provide a mapping file between its own terminology and the core ontology. This task is the responsibility of the security officer of every participant organization. The mapping files will serve to translate the local concepts to central ones as a preliminary step before performing the ontology-based data matching (inference) with Ω-RIDL (see Section 3.4).

Table 7-3 shows mappings between the local policies of the Hospital data repository and the central ontology. The first mapping concerns a subject mapping from the local hospital repository `A`, where the subject attributes are expressed as `name = employee_type` and `value = nurse`, to the central vocabulary used by the Master PDP, where `name(employee_type)` maps to `hcp-type` and `value(nurse)` maps to `nurse`.

Table 7-3: Mappings between the local (organization-specific) terminology and the core ontology

Repository	Term	Concept in the ontology
Hosp. RepositoryA	employee_type = nurse	hcp-type = nurse
Hosp. RepositoryA	file_type = vaccination fiche	document-type = summary information

HIV Center	Doc-type = medication fiche
	Location = Brussels
	Patient-Name = Herve
		Sensitivity-indicator = HIV

The second row shows a resource attribute mapping from the local vocabulary of hospital repository `A`, where the resource `name = file_type` and the resource `value = vaccination type`, to the central vocabulary where `name` maps to `document-type` and `value` maps to `summary information`.

8 Conclusion

In this document, we presented a security policy ontology based on the DOGMA (Developing Ontology-Grounded Methods and Applications) framework. Given this ontology, a system evaluating security policies can interpret the attributes and values of a policy, thus enabling policies to be defined using different vocabularies.

Although different security policy specifications (e.g. Ponder, REI or KAoS) have been proposed to reduce policy conflicts and facilitate interoperability, these specifications tend to concentrate on defining access control, obligation and delegation policies. Furthermore, these specifications require the author of a policy to have knowledge of logic paradigms (e.g. OWL). In contrast, our security and privacy policy ontology is grounded in natural language. This makes the DOGMA TAS^3 approach more user centric. The use of natural language enables technical (e.g. custodians, data controllers) and non-technical users (e.g. data subjects) to easily specify policies to control the use of data (e.g. CV). This does not have an impact on the enforceability of these policies by agents as the DOGMA framework provides mechanisms to convert DOGMA-inspired ontologies into description logic paradigms. As a result, our ontology follows the five general requirements of any policy representation; namely expressiveness, simplicity, enforceability, scalability and analysability. In this document, we first described the core elements of security policies; namely condition, subject, action, and target. Based on these concepts, we provided a conceptualisation for different types of security policies related to the TAS^3 project. Note that the concepts defined in the security policy ontology draws upon the concepts defined in the Descriptive Upper Ontology as well as the TAS^3 Upper Common Ontology. As a result, the TAS^3 lower common ontology provides a framework that (i) can be implemented to respond to policy requirements of a system in different security domains, (ii) can be extended through agreements between technical and non-technical user, (iii) can be enforced by in different systems, and (iv) can be analyzed by both human and software agents. The focus on this final version of the report was on the use of ontology and ontology-based tools in the TAS^3 overall architecture and pilots. We finally described how the ontology of security constraints can be used to assist the business process modeller into designing the security policies. The lower common ontology has continuously evolved in the security policy interoperability both in the employability and the eHealth domains, with the development of adapted (security policy interoperability) tools and more sophisticated authorization policies, including concepts such as environmental parameters and separation of duty.

Future work includes transferring the ontology-based solutions proposed here to cloud computing. Open research questions emerge from this work, such as how to evolve the security ontology through social processes or how to best take into account the user context for annotation recommendations.

References

[1] J. Weise and C. R. Martin, *Developing a Security Policy*, Sun blueprints online, Sun Microsystems, Inc., 2001. URL: http://www.sun.com/blueprints/1201/secpolicy.pdf.

[2] *eXtensible Access Control Markup Language (XACML)*, Standard version 2.0, OASIS, 2005.

[3] T. Berners-Lee, J. Hendler, and O. Lasilla, *The semantic web*, Scientific American May (2001), 34–43.

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

[5] P. Spyns, Y. Tang, and R. Meersman, *An Ontology Engineering Methodology for DOGMA*, Journal of Applied Ontology 3 (2008), 13–39.

[6] Gianluca Tonti, Jeffrey M. Bradshaw, Renia Jeffers, Rebecca Montanari, Niranjan Suri, and Andrzej Uszok, *Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder*, The SemanticWeb - ISWC 2003, 2003, pp. 419–437.

[7] N. Damianou, Dulay, E. Lupu, and M. Sloman, *The Ponder Policy Specification Language* , Proceedings of Workshop on Policies for Distributed Systems and Networks (POLICY 2001), 2001.

[8] L. Kagal, T. Finin, and A. Joshi, *A policy based approach to security for the semantic web*, The SemanticWeb - ISWC 2003, 2003, pp. 402–418.

[9] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott, *KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction and Enfo rcement*, Proceedings of 4th International Workshop on Policy (POLICY 2003), 2003.

[10] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein, *OWL Web Ontology Language Reference*, W3C Recommendation, World Wide Web Consortium, 10 February 2004, URL: http://www.w3.org/TR/owl-ref.

[11] M. Jarrar, *Mapping ORM into SHOIN/OWL Description Logic - Towards a Methodological and Expressive and Graphical Notation for Ontology Engine*, Proceedings of the OTM workshop, 2007, pp. 729–741.

[12] Q. Reul (Eds.), *TAS³ Common Upper Ontology*, Deliverable D2.2, Trusted Architecture for Security Shared and Services (TAS3), 2009.

[13] J. Stewart, E. Tittel, M. Chapple, *CISSP: Certified Information Systems Security Professional study guide*, SYBEX Inc., 3rd Edition, 2005.

[14] M. Sloman and E. Lupu, *Security and Management Policy Specification*, IEEE Network Special Issue on Policy Based Networking, 16(2):10-19, March 2002.

[15] M. Hafner and R. Breu, *Security engineering for Service-Oriented Architectures*, Springer-Verlag, 2009.

[16] A. Kim, J. Luo, and M. Kang, *Security ontology for annotating resources*, Proceedings of the 4th International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE'05), 2005, pp. 1483–1499.

[17] D. Trivellato, F. Spiessens, N. Zannone, and S. Etalle, *POLIPO: Policies & OntoLogies for Interoperability, Portability, and autOnomy*, Proceedings of the International Symposium on Policies for Distributed Systems and Networks, 2009, pp. 110–113.

[18] P .Kumar, S. Maheshwari, A. Kumar, R. Singh, A. Pruthi, A. Goyal, R. Goyal, *IT Security and Audit Policy*, Indian Department of Information Technology, URL: http://www.nsit.ac.in/pdf/itsa_policy.pdf. Last accessed: December 22, 2009.

[19] J. P. de Albuquerque, H. Krumm, P. Licio de Geus, *Policy Modeling and Refinement for Network Security Systems*, Proceedings of the 6th International Workshop on Policies for Distributed Systems and Networks (POLICY'05), pp. 24-33, 2005.

[20] N. Zhang, L. Yao, A. Nenadic, J. Chin, C. Goble, A. Rector, and D. Chadwick, *Achieving Fine-grained Access Control in Virtual Organisations*, Concurrency and Computation: Practice and Experience, 19(9):1333-1352, 2007.

[21] W. Burr, D. Dodson, R. Perlner, T. Polk, S. Gupta, E. Nabbus. *Electronic Authentication Guideline*, NIST Special Publication NIST Special Publication 800-63-1, Feb 2008.

[22] D. Chadwick (Eds.), *Design of Identity Management, Authentication and Authorization Infrastructure*, Deliverable D7.1, Trusted Architecture for Security Shared and Services (TAS³), 2009.

[23] P. Samarati and S. De Capitani di Vimercati, *Access Control: Policies, Models, and Mechanisms*, Foundation of Security Analysis and Design, pp 137-196, 2001.

[24} D. Chadwick and A. Otenko, *The PERMIS X.509 Role Based Privilege Management Infrastructure*, Proceedings of 7th ACM Symposium On Access Control Models And Technologies (SACMAT 2002), 2002, pp. 135–140.

[25] R. Sandhu and P. Samarati, *Access Control: Principles and Practice*, IEEE Communication Magazine, pp 40-48, September 1994.

[26] J. Alhadeff and B. Van Alsenoy (Eds.), *Requirements: Privacy, governance and contractual options*, Deliverable D6.1, Trusted Architecture for Security Shared and Services (TAS³), 2009.

[27] J. Mulle, J. Muller, Q. Reul, I. Ciuciu, J. Hoppenbrouwers, A. Boisvert, A. Blandin, Design of a semantic underpinned, secure & adaptable process management platform, Dec. 2010.

[28] I. Ciuciu, G. Zhao, J. Mulle, S. von Stackelberg, C. Vasquez, T. Haberecht, R. Meersman and K. Bohm (2011) *Semantic Support for Security-Annotated Business Process Models*, In. Proc. of Int. Conf. on Business Process Modeling, Development and Support (BPMDS'2011) held at CAISE 2011.

[29] I. Ciuciu and Y. Tang, *Towards Retrieving and Recommending Security Annotations for BPM Using and Ontology-based Data Matching Strategy*, In. Proc. of 1st Symposium on Data-Driven Process Discovery and Analysis (SIMPDA'2011), pp. 71-81, 2011.

[30] C. Vasquez and I. Ciuciu, *Ontology-based Context-dependent Annotation Templates Reccomendations*, to appear in Proc. of Workshop on Semantic&Decision Support (SeDeS'2011), OTM Conferences and Workshops, 2011.

[31] I. Ciuciu, G. Zhao, D.W. Chadwick, Q. Reul, R. Meersman, C. Vasquez, M. Hibbert, S. Winfield and T. Kirkham, *Ontology Based Interoperation for Securely Shared Services*, In Proc. of Int. Conf. on New Technologies, Mobility and Security (NTMS'2011), 2011.

[32] I. Ciuciu, B. Claerhout, L. Schilders and R. Meersman (2011) Ontology-based Matching of Security Attributes for Personal Data Access in e-Health, to appear in Proc. of International Symposium on Secure Virtual Infrastructures (DOA-SVI'2011), OTM Conferences and Workshops 2011.

[33] D. Trog, Y. Tang and R. Meersman, *Towards Ontological Commitments with Ω-RIDL Markup Language*. In: Ontologies, Databases and Applications of Semantics, Villamoura, Portugal, 2007.

[34] Verheyden, P. De Bo and R. Meersman, Semantically Unlocking database Content Through Ontology Based Mediation. In SWDB, pp. 109-206, 2004.

[35] P. De Baer, Y. Tang and R. Meersman, *An Ontology-Based Data Matching Framework: Use Case Competency-Based HRM.* In: Proceedings of the 4th Int. OntoContent'09 Workshop, On the Move to Meaningful Internet Systems, LNCS, pp. 514--523, Portugal, 2009.

[36] B. Claerhout, D. Carlton, C. Kunst, L. Polman, D. Pruis, L. Schilders, S. Winfield, *Pilots Specifications and Use Case Scenarios, TAS^3*, Deliverable D9.1, Trusted Architecture for Securely Shared Services. Available at: http://tas3.eu/, 2010.

Amendment History

Ver	Date	Author	Description/Comments
1.1	2010-01-11	QR	First version
2.3	2010-12-21	IC	Intermediate version
3.0	2011-10-14	IC	Final version