# Dealing with the Formal Analysis of Information Security Policies through Ontologies: A Case Study

**G. M. H. da Silva[1], A. Rademaker[1], D. R. Vasconcelos[1,2] , F. N. Amaral[3], Carlos Bazílio[3] V. Gonçalves[1] and E. H. Haeusler[1]**

[1] TECMF – Departamento de Informática – PUC-Rio
Catholic University of Rio de Janeiro,
Email: {hamazaki,arademaker,davirv,vaston,hermann}@inf.puc-rio.br

[2] Departamento de Computação – UFC

[3] Departamento de Ciência e Tecnologia – PURO – UFF
Email: {fnaufel,bazilio}@inf.puc-rio.br

**Abstract**

We present in this work, the structure of an ontology for Information Security (IS) applied to extract knowledge from Natural Language texts (IS standards, security policies and security control descriptions). This ontology is composed of the vocabulary for the IS Domain, and a particular kind of ontology description, the logical forms, that shows how should be the structure of the DL formulas associated with the texts. It is also discussed the relation between the structure of the formulas with the efficience of the reasoner.

*Keywords:* Information Security Policies, Description Logic, Ontologies

## 1 Introduction

Information Security (IS) is a non-trivial problem that usually comes in different levels of abstraction. At the *Business* level of abstraction we can find the information related to *Processes* and *Persons* that are involved in any of the enterprises activities. At the *System's* level we find the *Software* and *Hardware* under safeguard. The *Enterprise*'s or *Organization*'s security management realizes (Caralli 2004) this problem as a worth question at its own level, not merely as a technological problem to be solved by means of a software installation (as *Firewalls*, for example). Social aspects should be also taken into account[1] when studying security problems at the *Process* or *Person's* level. On the other hand, there are many attacks reported at the *System's* level[2], enough to take the technological problem also into worth count. Unnecessary to say that a security protocol broken can avoid an *organization* to fulfill its social role.

In order to solve the IS problem, two main approaches have been taken, one is based on the defense against predicted (rather hopefully predicted) threats and the other is based on maintaining the organization on previously known security levels. The former, named *Threat-Based* IS approach, tries to mount a strong defense to likely attacks, while the latter maintain the behavior of each entity in the organization at ever-controlled states. Most of the *security standards*, as well as *security protocols*, seems to adhere to each approach at some level, not necessarily disjoint.

[0]Research partially supported by CNPq grant 550652/05-1, The Anubis Project.

[1]Social Engineering techniques can be envisaged in order to design an attack to access the Enterprises ordinary trash, by means of an intruder personified by an employee or acting with the help of others naive employees.

[2]The CERT/CC (CERT/CC 2007) is responsible has a register of most of them.

The IS community created sets of *rules*, in a quite artificial Natural Language, in the sense of security conditions to be verified. These rules have been continuously updated and an organization must satisfy them in order to be considered *secure*. Well-known examples of these sets of rules are the *standards* provided by some committees (ISO, COSO, ANSI, ABNT, etc). Each set of rules, designed to a specific level of abstraction, carries a *terminology*, that is basically formed by the *linguistic* terms that denote the concepts involved in the specific domain of IS. One can verify by reading a typical *standard*, for example ISO-27001 (ISO 2005*b*), that it is, in form, a set of phrases in a quite artificial Natural Language pattern. The reason for that is the (intended) lack of ambiguity that such texts must have. Besides the *standards*, each organization has its IS Policy (**ISP**), which the abstraction level is significantly lower than a *standard*. One can easily verify that a condition for a *standard* must be turned into an obligation at the **ISP** level. And of course, in order to be implemented, this obligation must be written in a way that becomes easier the task of verifying the status of its own implementation. Thus, in *IS* terms we have two sets of rules and we should ensure that each rule of the higher set is covered by the rules of the lower set in every viewpoint for the application of the first. We call this a compliance testing. Thus, besides the task of designing rules and proving them to be adequate to the "reality" (the organization and where it "lives"), there is the task of comparing, by means of a compliance testing, two or more sets of rules. This is what we call the **formal** statement of the IS problem.

For the people from the Ontology community, the above pictured scenario has turned into an invitation to work. Ontologies are formalizations of conceptual worlds and they also serve to prove certain properties about it. These proofs may show the adequacy of the Ontology to its object. Among the worth properties one may find out, the consistency testing, that it is able to ensure that the ontology speaks about something, and the subsumption of concepts, that it is able to verify the partial order among concepts, are quite useful to apply to our formal statement of the IS problem. In fact, there is worldwide many projects and initiatives that do that. The references would be huge if we decided to cite them. And because the wide range of applications, strategies and level of formalism used by them, as well as the level of detail reported by each of them, we consider out of the scope of the present article. Only as a matter of preference, we point out the repository of ontologies worldwide placed by the users of the Protégé Ontology Editor (Stanford University 2006).

In (do Amaral et al. 2006) we describe the **Anubis** project, i.e. an architecture and a set of formal tools, as well as a methodology, to help a Security Company in de-

signing, validating and maintaining a knowledge base on **ISPs**. As a case study and with the purpose of preserving industrial property (do Amaral et al. 2006), we omit many details and in certain moments the actual names and denominations inside the architecture were presented in a more general way. The relevant information is briefly presented in Section 2. The structure used representing Natural Language construction in order to facilitate the task of formalizing IS concepts and rules from NL Norms is presented In Section . The aim of the present article is to report worth conclusions that have been drawn during the development of the above mentioned project.

We focus on two main conclusions, discussed at the end of this article, that were drawn. The particular style of an Ontology Description, as a set of DL formulae, has became critical in terms of efficiency. It is interesting to note how the style of equivalent, by means of a syntactic homomorphism, DL sets of formulas can produce highly different performance regarding validation. Although linguistic glues could drive into a more **nested** *logical form* representation, the use of **flat** representations showed up quite better computational performances. This is detailed in Section 4. Secondly, and not surprising, many *actions* conceived from a set of controls by an specialist did not validated against their respective set of controls, without adding some particular axiom stating some either obvious or automatic effect, transparent to the specialist. Section 4 lists some examples where this situation happened.

## 2 A brief presentation of the case study

The formalization of text-based information is an important issue in the deployment of semantics-aware technologies in the enterprise. It is very common to encounter situations where knowledge stored in natural-language documents must be made available to agents (human or software-based) for processing and decision-making. This case study can be seem as an attempt to provide an ontology-based approach to the formalization of normative texts in the domain of Information Security (IS), such as security policies defined by organizations and *standards* defined by Security Committees. In (do Amaral et al. 2006) we discuss the principles involved in development this approach.

Because the IS-related terminology tends to vary according to the source, we adopt the following definitions: a *standard* is a public document consisting of a set of *control objectives*, which are goals to be attained by the organization if a great level of security is desired. Roughly speaking, control objectives state *what* should be achieved; being expressed at a rather high level of abstraction, they do not lend themselves to direct application to the organization's processes and practices. It is by means of *security controls* that the organization actually specifies *how* to achieve the security requirements laid out by the control objectives. Security controls (or simply *controls*) are low-level technical measures that can be deployed in order to protect the organization's devices and processes against potential threats. To bridge the gap between high-level control objectives and low-level controls, the organization defines its *security policy*, consisting of *actions* to be taken in order to comply with the adopted *standards* and possibly with other security requirements identified by a process of risk analysis. In this scenario, one control objective may give rise to several different actions in the security policy, and each of those actions may be implemented by a set of different controls.

Many tasks are involved in the formalization of a IS domain as described above: for example, *standards* must be selected, actions must be formulated, controls must be defined, deployed and managed. Furthermore, all levels must support maintenance: updates in the *standards* must be followed, policies must be revised, and controls must be replaced or incremented because they become ineffective, inapplicable or simply insufficient. It should be clear that security experts can greatly benefit from the use of semi-automatic, knowledge-based and formal tools to assist them in these activities. For the computer science community we would say that we focus on the use of CAV[3] tools. In fact, in our case study we have already the set of *controls*, responsible by the effectiveness of its security analysis system in the market. We call, by an abuse of language, this set of *controls* the IS Knowledge Base (**ISKB**). Thus, the approach to the followed is to group *controls* into *actions*, checking their respective consistency and verifying the compliance of the group of *controls* with regard to the respective *actions*. Subsequently, *actions* and *control objectives* should be checked for compliance too.

Before explaining in more detail our approach. It would be interesting to mention the almost natural boundaries of our industrial scenario: 1- *Controls* (*Security Controls*) modification should be avoided and their level of abstraction is at the lowest possible; 2- *Control objectives* cannot be modified even in its form, since they are rigid documents (*the standards*); 3- *Actions* are designed by the human being for a better clustering and understanding of the Base of *controls* ISKB, they can and must be modified as a way to easily reach an understandable compliance between the *KB* and the chosen *standard*.

Our approach consists of the following elements:

- *Actions* are represented at the *logical form* level, a concept from the area of natural language understanding (Allen 1995). Basically, logical forms are constructs in some suitable formalism used to represent the context-independent semantics of natural language utterances. Currently the edition of *actions*, *controls*, and *control objectives* in the future, is accomplished by means of a Protégé Plug-in developed as a Ontology-Driven editor guided by the *logical forms* ontology.

- The inference capabilities of the proposed framework are based on a description logic (DL) (Baader et al. 2003). Logical forms representing actions are actually stored as DL concepts, and the facts that must hold about these concepts are stored as DL axioms. The user may pose queries to a DL reasoner, which will provide answers based on these axioms. Some background on DL is assumed in this article. The reasoner used in the experiments here reported is the Pellet implementation of the Tableaux proof method for DL. Note that the IS domain and the need of additional axioms as the reported in Section 4 could ask a more powerful DL, but this did not happen, indeed.

- Ontologies (Gruber 1993) serve as the unifying structure for the above two elements. Unnecessary to say that an ontology consists of concepts, properties and logical expressions denoting constraints that hold between these concepts and properties. In our approach, actions in logical forms and axioms about them are expressed in terms of these concepts, properties and constraints. One language for representing ontologies is OWL DL (Dean & Schreiber 2004), which can be translated in a straightforward way to the language used by DL reasoners, providing for an easy interface between the ontology and the inference services of our framework.

The ideal and goal architecture to help on formalizing IS from NL texts guided by a domain specialist is depicted in Figure 1. The life-cycle of our IS ontology development is shown in Figure 2.

[3]Computer Aided Validation.

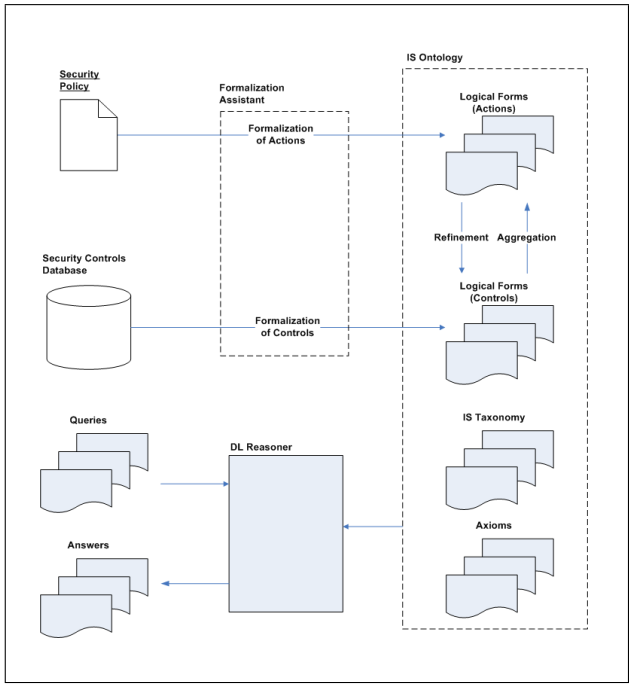


Figure 1: Elements of our ontology-based approach

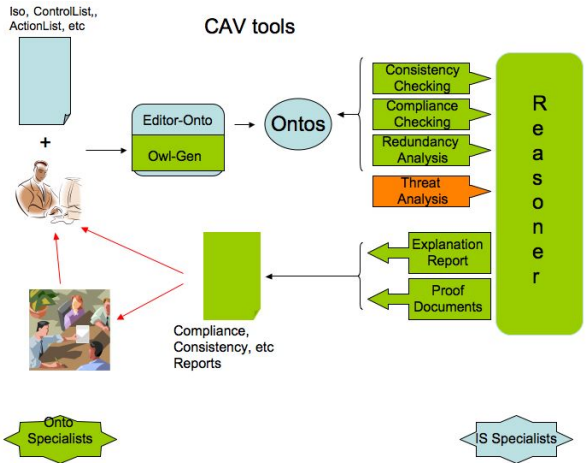


Figure 2: Life-cycle of IS ontology development.

## 3 The rationale on the *IS* Ontology defined in the project

In (do Amaral et al. 2006) it is discussed the decision for the use of an structured way for representing NL constructs in order to facilitate the task of formalizing IS concepts and rules from NL Norms. In fact, almost every rule is ruled by a verb (only one verb)[4] and other elements are linked by the respective attribute that can be, but does not need to be, determined by the syntactic role it has in the phrase. For example, **hasTheme** is normally associated to the direct object, **hasAgent** to the subject, **hasPurpose** to the subordinated phrase and so on. These structures are described using DL formulas as part of the *Logical forms*, that were envisaged just in order to solve this situation in a NL processing scenario. We use it as the core of the Ontology. Thus, our IS ontology has a linguistic core that is responsible to describe concepts and relationships worth in structuring Normative texts, as an indexed set of Phrases (the *actions*, *controls* and *control objectives*). Unnecessary to say that this core is semantically neutral. In order to reason on IS concepts, there must be an Ontology on the IS vocabulary. For example, *Server*, *Firewall*, *Operating System*, *CEO*, *Meeting Room* and so on are concepts belonging to this IS vocabulary.

One of the points that were strongly stressed during the project was the ontology definition by the domain specialist himself. The IS specialist is used to work on Natural Language texts, namely, *standards*, Laws, Norms of the Organization, Security Protocols (not only the software, but Human-based Security Protocols) and so on. It was noted that specialist prefers keep working this way, that is, his focus is on the written material. Discussing the, general, extraction of knowledge from Natural Language texts is outside the scope of this project. However, the use of some methodology and a tool supporting it urged. In order to solve this, a Protégé Plug-in was implemented. This plug-in is an editor guided by the core ontology (an OWL-DL document), and, by interacting with a specialist on IS, produces a set of DL-formulas representing either a IS vocabulary, or an *action*, or a *control*, or a *control objective*. The process of edition works by marking and typing with DL meaning the linguistics elements of a NL normative text. In fact, the plug-in was not used for the edition of the almost 2000 controls and actions, already taking part of the last version of the Ontology. Besides that, part of the vocabulary related to the IS on the *Linux* Operating System was defined by an IS specialist (an employee of the Industrial Partner).

The main advantage of use *logical forms* is how it is well-suited to the guidelines for Norms constructions (see (ISO 2005*a*)). For example, in most cases our ontology does not, and should not, distinguish between different parts of speech, so as to render the constructed logical forms as general as possible; thus, a noun whose root is shared with a verb is represented by the concept associated to the verb; for example, “connect” and “connection” are both associated to the single concept Connect, and “configure” and “configuration” are both associated to the single concept Configure. In (ISO 2005*a*), sec 6.4, states that all concepts should be preferentially stored in noun form; we have chosen verb form instead of noun form because the actions in a security policy have verbs as their most important words.

Before we dig into the next section that describes some examples that allowed us to draw the conclusions mentioned in sections 1 and 5, a couple of words on way our IS ontology is, from the logical point of view is needed. The main feature of our IS ontology is that it is a T-Box ontology. There is no individual description or mention to them either. This is justified by observing that from the point of view of IS, individuals seem to have no importance. For example, a *Person* invades a private area of a corporation equals to any other invader in the same conditions. It is not important to name the invaders, but to describe them. The same can be said about the facilities of the corporation (organization), about the CEO, the *Operating System*, and, finally, a password instance is unimportant whenever compared both concepts *valid password* and *invalid password*. Besides that, because we deal with negation, disjunction and conjunctions, and, the core ontology uses restriction, we use $\mathcal{ALC}$(Baader et al. 2003) as our logic language. This has the side effect of not going so high in the complexity hierarchy of the DLs. Basically we are inside PSPACE-complete worst-case complexity. This is not so far[5] from the usual reasoning complexity for knowledge representation systems.

The next section is the core of this article showing many interesting examples taking part in this IS ontology under construction. Both conclusions mentioned in the introduction were drawn from almost the same kind of formalization. Thus, both subjects are analyzed in the sequel.

[4] Modal forms must be distilled, since they are in general redundant.

[5] We believe $NP \neq PSPACE$.

## 4 Formalization of Controls and Actions of IS

In this section we illustrate the formalization of actions and controls of IS from normative texts. In the sequel, we consider a few examples of actions and controls formalized by nested and flat representations. These examples provide samples of the different types of axioms that are needed in the ISKB in order to accomplish compliance tests. It is worth mentioning that the automatic process is simpler in flat form in which the axioms are more modular than a nested one. And, in addition, the validation is also more efficient using flat forms than nested forms. At the end of this section, we show the performance using both approaches.

Before going into the examples, a couple of words is need. The *nested* style of specifying *Logical Forms* (*LF*) was the first choice, it was induced by the nested style of the modifiers (modalities, adverbs, subordinated sentences, and etc) usually found in Natural Languages sentences. The use of attributes (roles), in our *LF* linguistic ontology, as a way of specifying the role each phrasal element has in the phrase is of course, a natural one. We recall that if $R_1$ and $R_2$ are DL roles, and, $C$ and $D$ are concepts, there is no logical equivalence between the concepts $\exists R_1.(C \sqcap D)$ and $\exists R_1.C \sqcap \exists R_1.D$, respectively, as well as, for general $R_1$ and $R_2$, $\exists R_1 \exists R_2.C$ are not logically equivalent concepts either. Note that in this last case we can say that $\exists R_2.C$ is in the context of $R_1$ and not the other way around. We have an explicit dependence of $R_2$ from $R_1$. Thus, by taking the *nested* way of specifying the *LF* ontology, we force the dependence everywhere. This is not bad, if this is done on the whole set of *actions* and *controls* consistently. This has an advantage of a better explanation on how a *control* is subsumed by a particular *action*. Other advantage of the nested style is the fact that it provides a more automatic way of rendering phrases into *LF*, for the sequence of modifiers is already at the sentence itself. However, as anyone can observe, the use of passive voices, the indirect style of speech and similar features that Natural Languages possesses, might force us to consider the *nested* style advantages as apparent advantages. In fact, the first experiments taken inside our knowledge base (KB) proceeded by the use of the *nested* style. This style provided almost clearer ways by the simple act of putting side-by-side the subsumed pair (*control*, *action*). They were, almost the cases, of the same kind when considering the replacements induced by the *IS*ontology. For example, from $Linux \sqsubseteq System$ and $Execute \sqsubseteq Manage$ we have:

$$\exists \textit{hasVerb}.(Execute \sqcap \exists \textit{hasTheme}.Linux \sqcap \exists \textit{hasPurpose}.(\exists \textit{hasVerb}.Update \sqcap \exists \textit{hasTheme}.Linux)) \sqsubseteq \exists \textit{hasVerb}.(Manage \sqcap \exists \textit{hasTheme}.System \sqcap \exists \textit{hasPurpose}.(\exists \textit{hasVerb}.Update \sqcap \exists \textit{hasTheme}.System))$$

However, when the ontology became bigger (more than 100 *controls* and some action, *circa* 5000 concepts and 27 roles), the classification (by means of subsumption) task got extremely heavy, taking in extreme cases more than a couple of hours. Because of this we made the ontology into a flat style *LF*ontology. Who knows the proof method **Pellet** uses (tableaux), agrees that the time to classification of concepts must die down. We turn to this discussion after the following examples on the need of additional axioms.

**Example 4.1.** *Suppose the organization has deployed an action with the following description:*

> *Configure the parameter"type" parameter with "Nt5DS" value parameter in [SO Windows].*

*The nested and flat logical form expressions that represent such action are the concepts defined by:*

*Action1Nested* $\equiv$
1 $\exists \textit{hasVerb}.(Configure \sqcap$
2 $\quad \exists hasTheme.(ParameterType \sqcap$
3 $\quad\quad \exists hasValue.(Nt5DS)) \sqcap$
4 $\quad \exists hasLocation.Windows)$

*Action1Flat* $\equiv$
1 $\exists \textit{hasVerb}.Configure \sqcap$
2 $\exists hasTheme.(ParameterType \sqcap$
3 $\quad \exists hasValue.(Nt5DS)) \sqcap$
4 $\exists hasLocation.Windows$

*Now consider the organization has deployed a security control with the following description:*

> *The parameter Registry "type" must be configured with the value of "Nt5DS".*

*The nested and flat logical form expressions that represent such control are the concepts defined by:*

*Control1Nested* $\equiv$
1 $\exists \textit{hasVerb}.(Configure \sqcap$
2 $\quad \exists hasTheme.(ParameterType \sqcap$
3 $\quad\quad \exists hasValue.(Nt5DS \sqcap$
4 $\quad\quad\quad \exists hasPossessor.WindowsRegister))))$

*Control1Flat* $\equiv$
1 $\exists \textit{hasVerb}.Configure \sqcap$
2 $\exists hasTheme.(ParameterType \sqcap$
3 $\quad \exists hasValue.(Nt5DS)) \sqcap$
4 $\exists hasPossessor.WindowsRegister$

*To prove the compliance of our structures, we need to add axioms that correlate the location of Windows parameter with Windows Register. As we already said, the nested approach is more complicated than flat approach even in the used axioms.*

*The following axioms are necessary for the nested formalization.*

$$\textit{WindowsParamType} \equiv \textit{RegWindowsParamType}$$

*WindowsParamType* $\equiv$
1 $\exists hasTheme.(ParameterType \sqcap$
2 $\quad \exists hasValue.(Nt5DS)) \sqcap$
3 $\exists hasLocation.Windows$

*RegWindowsParamType* $\equiv$
1 $\exists hasTheme.(ParameterType \sqcap$
2 $\quad \exists hasValue.(Nt5DS \sqcap$
3 $\quad\quad \exists hasPossessor.WindowsRegister))$

*The following axioms are necessary for the flat formalization.*

$$\textit{WindowsLocation} \equiv \textit{WindowsRegPossessor}$$

$$\textit{WindowsLocation} \equiv \exists hasLocation.Windows$$

$$\textit{WindowsRegPossessor} \equiv \exists hasPossessor.WindowsRegister$$

In the following example, we illustrate that some representations do not need additional axioms.

**Example 4.2.** *Suppose the following action is part of the organization's security police:*

> *Set the Date and Time of the [system xyz].*

*The nested and flat logical form expressions that represent such action are the concepts defined by:*

*Action2-Nested* $\equiv$
1 $\exists \textit{hasVerb}.(Set \sqcap$
2 $\quad \exists hasTheme.(DateTime \sqcap$
3 $\quad\quad \exists hasLocation.(System)))$

*Action2-Flat* $\equiv$
1 $\exists \textit{hasVerb}.Set \sqcap$
2 $\exists hasTheme.DateTime \sqcap$
3 $\exists hasLocation.System$

*Suppose the organization has deployed a security control with the following description:*

*The date and time of Linux Red Hat must be configured.*

*The nested and flat logical form expressions that represent such control are the concepts defined by:*

*Control2-Nested* $\equiv$
1 $\exists$*hasVerb.*$(Configure \sqcap$
2 $\exists hasTheme.(DateTime \sqcap$
3 $\exists hasLocation.(LinuxRedHat)))$

*Control2-Flat* $\equiv$
1 $\exists$*hasVerb.*$Configure \sqcap$
2 $\exists hasTheme.DateTime \sqcap$
3 $\exists hasLocation.LinuxRedHat$

*In this example, there was no need of additional axioms to prove compliance of this control and action since we have* $Set \equiv Configure$ *and* $LinuxRedHad \sqsubseteq OperatingSystem \sqsubseteq System$ *in the ISKB.*

The next example illustrate the need of a more sophisticated set of axioms in order to show that an actions subsumes a control, or set of controls.

**Example 4.3.** *Suppose the following action is part of the organization's security police:*

*Define TCP gate with a different value of the pattern for the execution of [xyz service].*

*The nested and flat logical form expressions that represent such action are the concepts defined by:*

*Action3-Nested* $\equiv$
1 $\exists$*hasVerb.*$(Define \sqcap$
2 $\exists hasTheme.TCPGate \sqcap$
3 $\exists hasValue.(TCPPatternDifferentValue) \sqcap$
4 $\exists hasPurpose.(Execute \sqcap$
5 $\exists hasTheme.SoftwareService))$

*Action3-Flat* $\equiv$
1 $\exists$*hasVerb.*$Define \sqcap$
2 $\exists hasTheme.TCPGate \sqcap$
3 $\exists hasValue.(TCPGatePatternDifferentValue) \sqcap$
4 $\exists hasPurpose.(Execute \sqcap$
5 $\exists hasTheme.SoftwareService)$

*Now suppose the organization has deployed a security control with the following description:*

*The Apache Tomcat must be performed in a gate different from the standard gates (8080 e 8443) and from the gates reserved services (1 a 1024).*

*The nested and flat logical form expressions that represent such security control are the concepts defined by:*

*Control3-Nested* $\equiv$
1 $\exists$*hasVerb.*$(Execute \sqcap$
2 $\exists hasTheme.(ApacheTomcat \sqcap$
3 $\exists hasLocation.(TCPGate \sqcap$
4 $\exists hasValue.(TCPGateDifferentReservedService) \sqcap$
5 $\exists hasValue.(TCPGatePatternDifferentValue))))$

*Control3-Flat* $\equiv$
1 $\exists$*hasVerb.*$(Execute \sqcap$
2 $\exists hasTheme.(ApacheTomcat \sqcap$
3 $\exists hasLocation.(TCPGate \sqcap$
4 $\exists hasValue.(TCPGateDifferentReservedService) \sqcap$
5 $\exists hasValue.(TCPGatePatternDifferentValue))))$

*In any case, the structure of the* control *is quite different from the structure of the* action*. In fact, the level of abstraction of* action *seems to be higher. However, taking the meaning of both sentences into account, we can see that the subsumption would hold, if, we established that "To define Something* $X$ *in a way* $Z$ *in order to Execute Something* $Y$*" is subsumed by "To Execute Something* $Y$ *in a way* $Z$ *(different from the default way) running in Location* $X$*". In our specific case, this is expressed by the following axioms:*

*RightHand-on-Defining* $\equiv$
1 $\exists$*hasVerb.*$Define \sqcap$
2 $\exists hasTheme.TCPGate \sqcap$
3 $\exists hasValue.TCPGatePatternDifferentValue \sqcap$
4 $\exists hasPurpose.(Execute \sqcap$
5 $\exists hasTheme.SoftwareService)$

*LeftHand-on-Defining* $\equiv$
1 $\exists$*hasVerb.*$Execute \sqcap$
2 $\exists hasTheme.SoftwareService \sqcap$
3 $\exists hasLocation.(TCPGate \sqcap$
4 $\exists hasValue.(TCPGateDifferentReservedService \sqcap$
5 $\exists hasValue.TCPGatePatternDifferentValue))$

*LeftHand-on-Defining* $\sqsubseteq$ *RightHand-on-Defining*

*The above axiom together with the following axiom belonging to the ISKB finally prove the required compliance between the* control *(flat) and the* action *(flat). For the nested version the axioms (above) are analogous.*

*ApacheTomCat* $\sqsubseteq$ *SoftwareService*

The last thing to note in this section is the difference of performance between to two styles of formalization, the nested and the flat one. In order to compare performances a small part of the ISKB was extract. The ontology classified has 46 *controls*, 8 *actions* and 10 (additional) axioms. The time processed by **Pellet** was 22.844 seconds, the nested case, and, 4.734 seconds, the flat case. These experiments were executed in the same machine, a Pentium IV with 1 GB RAM, under similar conditions (no other application was running together with **Pellet**, but the **Protege** itself).

## 5 Conclusion

In section 4 two stylistically different DL representations for *logical forms* (*LF*) showed up quite different performances, although homomorphic[6] as formal specification in DL , when proceeding into automatic verification of compliance. The initial set of *LF* attributes had 27 different roles, and, it was designed with the explicit purpose of covering almost linguistic aspect a phrasal utterance in the IS domain. But, at the beginning of the studies, it was realized that a smaller set would be enough. A subset of the initial set, containing only 8 *LF* attributes, was defined. The *actions* that are shown in this article are already using only 8 *LF* attributes. The current stage of the IS ontology has been tuned according to the main observations on under-specification (lack of essential formalization, as for example the axioms shown in section 4), and, on performance.

During the development of this case study, besides the already mentioned conclusions, a methodological conclusion concerning the interaction with the specialist is worth of mentioning: the tools must be designed to attend whose asked the tool. In this sense, to the architecture and set of tools, it remains to add a explanation generator (from proofs) that allows the specialist to analyze the validation at almost the same level of abstraction than the specification (ontology).

Last, but not least, it is important to note that the overall approach shown here can be also, in principle, applied to any other domain with similar features, namely, to have written knowledge in an adequate NL form.

[6] This means that there is a function mapping on style into another preserving the subsumption relationship

## References


Allen, J. (1995), *Natural Language Understanding*, 2nd edn, Benjamin Cummings.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. & Patel-Schneider, P., eds (2003), *The Description Logic Handbook*, Cambridge University Press.

Caralli, R. A. (2004), 'Managing for enterprise security', Technical Note CMU/SEI-2004-TN-046. Available at `http://www.sei.cmu.edu/publications/documents/04.reports/04tn046.html`.

CERT/CC (2007), 'Incident notes'. `http://www.cert.org/incident_notes/`.

Dean, M. & Schreiber, G. (2004), 'OWL Web Ontology Language Reference', `http://www.w3.org/TR/owl-ref/`.

do Amaral, F. N., Bazílio, C., da Silva, G. M. H., Rademaker, A. & Haeusler, E. H. (2006), An ontology-based approach to the formalization of information security policies, *in* 'VORTE - Workshop on Vocabularies, Ontologies, and Rules for the Enterprise', Hong Kong.

Gruber, T. R. (1993), Towards principles for the design of ontologies used for knowledge sharing, *in* N. Guarino & R. Poli, eds, 'Formal Ontology in Conceptual Analysis and Knowledge Representation', Kluwer Academic Publishers.

ISO (2005*a*), *ANSI/NISO Z39.19-2005, Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies*, NISO.

ISO (2005*b*), *BS ISO/IEC 27001 Stand Alone*, ISO/IEC.

Stanford University (2006), 'The Protégé Ontology Editor and Knowledge Acquisition System', `http://protege.stanford.edu`.