EMPLOYING ONTOLOGIES FOR THE DEVELOPMENT OF SECURITY CRITICAL APPLICATIONS

The Secure e-Poll Paradigm

DRITSAS S.[1], GYMNOPOULOS L.[2], KARYDA M.[2], BALOPOULOS T.[2], KOKOLAKIS S.[2], LAMBRINOUDAKIS C.[2], GRITZALIS S.[2]

[1]*Department of Informatics, Athens University of Economics and Business, GR-10434, Greece*

[2]*Laboratory of Information and Communication Systems Security (Info-Sec-Lab, Department of Information and Communication Systems Engineering, University of the Aegean, Samos, GR-83200, Greece*

Abstract: Incorporating security in the application development process is a fundamental requirement for building secure applications, especially with regard to security sensitive domains, such as e-government. In this paper we follow a novel approach to demonstrate how the process of developing an e-poll application can be substantially facilitated by employing a specialized security ontology. To accomplish this, we describe the security ontology we have developed, and provide a set of indicative questions that developers might face, together with the solutions that ontology deployment provides.

Key words: Security Ontology, Application Development, Internet Voting, e-Poll

1. INTRODUCTION

Elections constitute a fundamental function of democracy, not only by providing a means for the orderly transfer of power, but also by promoting citizens' confidence in the government through their participation. Within the last years, there has been a strong interest in voting over the Internet as a means to provide a convenient way of voting, and thus increase participation

in elections. Election systems, however, need to meet demands concerning security, and especially confidentiality, among others. Security features are most important in the case of *remote Internet voting*, which poses increased security requirements with respect to other Internet voting types, such as *poll site Internet voting*, or *kiosk voting*.

In this paper we address the issue of developing a secure e-poll application, employing a specialized security ontology. Since developing electronic government and especially electronic voting applications requires meeting a wide range of security requirements, we have developed a security ontology that facilitates the development of secure applications by assisting the design and development process.

In the next section we give an overview of the connection between ontologies and software development, emphasizing on the role ontologies can play for building secure applications. In section three we describe the security ontology we have developed and its specific context. Section four demonstrates how this ontology can substantially facilitate the process of developing a secure e-poll application. Finally, section five discusses the advantages and limitations of the proposed use of ontologies in secure application development, and the last section presents our overall conclusions and directions for further research.

2. ONTOLOGIES AND SOFTWARE ENGINEERING

An ontology is a formal, explicit way for modelling and describing a segment of the world for which we agree to recognize the existence of a set of objects and their interrelations. They constitute an "[*e*]*xplicit specification of a conceptualization*" [25]. Thus, an ontology is the attempt to express an exhaustive conceptual scheme within a given domain, typically a hierarchical data structure containing all the relevant entities, their relations and the rules within that domain.

In computer science, ontologies are mainly used as a means for modelling information and for providing inference and reasoning techniques. For example, ontologies enable computers to go beyond the mere layout of documents by capturing their semantics and enabling computers to process them in a meaningful way.

2.1 Ontology based software engineering

Ontologies could play an important role in software engineering, as they do in other contexts, where they: (a) provide a source of precisely defined

terms that can be exchanged between people, organizations and applications, (b) provide a shared understanding concerning the domain of study, and (c) represent all hidden assumptions concerning the objects related to a certain domain. Although there are many research efforts to develop ontologies, software engineering still does not have a detailed ontology, which describes the concepts, that domain experts agree upon, as well as their terms, definitions and meanings.

Nowadays it is well understood that ontologies provide a useful theoretical and methodological tool for facilitating the software engineering process. In addition, it has been argued that security issues should be taken into consideration in all the stages of the software development process. Unfortunately, security features are typically built into an application in an ad hoc manner or are only integrated later during the system lifetime. In this context, it is obvious that we need methodologies, which can support the integration of security throughout the development life cycle, since it is generally accepted that security should be "*built in*" rather than "*added on*" applications.

Currently, a number of software engineering methodologies have been proposed for handling security issues at the design level: NFR [9, 10], Tropos [11], i* [12], RBAC [13], M-N framework [14], GBRAM [15,16]. Most of these methodologies consider the specification and validation of security requirements from the business goals, but do not refer to how these requirements can be translated into system components, nor do they offer any specific suggestion for related and applicable implementation techniques. Additionally, most of the above approaches are rather close to the technical aspects of security and particularly of specific application domains. As such, they do not provide a generic model of security and thus cannot be used for specifying security patterns. Such patterns could be used in order to incorporate security requirements and techniques into the software development process.

Therefore, we believe that the development of a security ontology that describes the basic security-related concepts would be very useful.

2.2 Security ontologies

Today, software developers lack a common approach that would bridge the desirable security requirements with the techniques that can be adopted in order to design and implement secure applications. A first step towards this could be the development of an ontology that will support the modelling of the basic security concepts and their integration into a model-driven software development process. The advantages of deploying such an

ontology are: (a) express the most important security concepts, (b) realize the relations among the above concepts, (c) provide a common understanding and vocabulary of security issues among application developers, and (d) facilitate the development of secure applications.

However, to the best of our knowledge, today there is not a shared body of practice for the development of a security ontology that will be used as a common base for the development of secure applications. Loosely related work focuses only on access control issues [24]. Standards discussed include XML Signatures [3] and integration with Security Assertions Markup Language (SAML) [4, 5]. Furthermore, work on KAON [6] focuses mostly on the managing infrastructure of generic ontologies and metadata, whereas in [7] authors present a policy-ontology.

Raskin et al. presented an ontology-driven approach to information security [8]. They argue that a security ontology could organize and systematize security concepts (e.g. attacks). Furthermore, the inherent ontology modularity could support the reaction to attacks by relating certain controls with specific attack characteristics, as well as attack prediction.

The KAoS Policy and Domain Services is another approach based on ontologies for the representation of security related concepts [17]. Specifically, while the approach was primarily oriented to the dynamic requirements of software agent applications, it has been used in general-purpose environments as well [18]. The KAoS framework proposes a detailed Ontology for Security Policies along with other notions (such as Actors, Entities, etc.) [19].

3. RESEARCH METHODOLOGY

It is widely accepted that no general and robust methodology exists for developing ontologies. However, several guidelines exist for dealing with the development of an ontology. According to [1], the two first steps towards the development of an ontology are: (a) determining its domain and scope, and (b) considering the use of existing ontologies.

3.1 Building a Secure Application Ontology

First of all, we decided that the e-poll domain of our developed ontology should have the following characteristics:

- Voter authentication is a mandatory requirement. Voters are issued credentials to authenticate themselves.

- There is a specific list of authorized voters (not everyone is allowed to vote).
- Voters are not allowed to vote more than once.
- Voters vote from any computer connected to the Internet.
- The ballot is constructed by the election officials or organizers and voters are presented with predefined multiple choices or/and with alternative ways of expressing opinion (to accommodate asset and range voting).

Our approach during the development of the ontology was heavily influenced by the related work presented in section 2, and was focused on the context of electronic voting. In order to produce an instantiated ontology, we used in a fair extent databases like the CRAMM database of countermeasures [2].

Besides choosing the basic concepts and instantiating our ontology, we also came up with subclasses. It should be noted however that our efforts were focused in the high level design of the ontology, and not on trying to include every possible subclass or instance we could think of. Details about the actual ontology are given in section 4.2.

3.2 Methods and Tools

Finding and reusing existing material was only one step towards the development of our ontology. We generally followed the steps provided in [1]; giving emphasis in the iterative procedure they propose. Every cycle in this procedure had roughly four phases: determining competency questions, enumerating important terms, defining classes and class hierarchy, and instantiating.

Competency questions are loosely structured questions that a knowledge base built on the ontology should be able to answer. Setting and elaborating on competency questions is an efficient way to identify and then focus on the desired area. Next, we give an example of a competency question and the respective answer given during the ontology construction:

```
Q: Are voters stakeholders of the system?
A: Yes.
```

Enumerating important terms within the scope set by competency questions and the respective answers is a prerequisite for defining ontology classes. We gathered approximately one hundred related terms. Some of them formed ontology classes; other formed properties of classes and some were not used at all.

In the next phase, classes and the class hierarchy were developed. In Figure 1, we depict the full ontology hierarchy, together with class slots and

their facets. After that, the relations between classes and also the domain and range of each slot were elaborated.

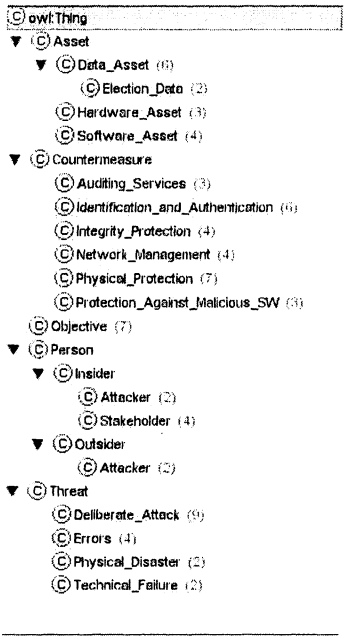

Figure 1. The ontology hierarchy.

The last phase in each cycle is that of instantiation; each class was given specific instances. The number of instances is depicted in parentheses in Figure 1.

The four phases described above were repeated several times. The ontology was queried after each iteration (please refer to section 4.3), and iterations ended only when the results obtained were considered satisfactory.

The tools used for developing and validating the ontology were Protégé and Racer. Protégé [21] is a software tool used by system developers and domain experts to construct domain ontologies. Protégé itself provides only core functionality; to develop our ontology, we used the Protégé plug-in, which facilitates the development of OWL [26] and RDF [27] ontologies.

Racer [22] is an inference engine that can be used for query answering over RDF documents. We used Racer in order to check our developed ontology (see section 4.2) for inconsistencies, and for submitting queries to the ontology in order to verify its validity (see section 4.3). The queries were expressed in the new Racer Query Language (nRQL).

nRQL is a description logic query language for retrieving individuals from an A-box (a set of assertions about individuals) according to specific

conditions. It allows the use of variables within queries which are bound against those A-box individuals that satisfy these conditions. The language is substantially more expressive than traditional concept-based retrieval languages offered by previous description logic reasoning systems. A description of nRQL's syntax is beyond the scope of this paper, but the interested reader is referred to [23].

The integration between Protégé and Racer was achieved through the RQL Tab plug-in, which allows the OWL plug-in of Protégé to send queries to Racer and receive back the results.

4. DEVELOPING A SECURE APPLICATION FOR E-POLLS

4.1 The e-Poll Application

Remote Internet voting is an attractive solution, especially for the disabled, since it allows voting from home or work; at the same time however, such systems face significantly higher risks with regards to the confidentiality and integrity of the voting process. Developing applications that support Internet voting is therefore a security critical task, since application designers and developers make critical decisions about issues concerning the confidentiality and integrity of the data, as well as about the availability of the voting system.

To validate the usability of the ontology we developed, we employed it to the design and development of a secure e-poll application. The application we worked on supports Internet voting for organizations as well as for other bodies (e.g. local authorities) wanting to organize an e-poll. It is a distributed application, requiring that people participating in the e-poll have Internet access and can visit the web site that is hosting the e-poll. Organizers, on the other hand, use the back-office application, that can help them manage all necessary voting processes, such as voter registration, vote tallying, ballot design and so forth.

Generally, the voter registration process is considered one of the weakest links in the electoral process. Secure and reliable Internet-based voter registration relies on appropriate authentication infrastructure. Since an adequately secure authentication infrastructure is not yet available, initial registration for the e-poll system is conducted offline. After being registered, voters are provided with a password and/or a digital signature from the

election organizers. They can use this authentication means for voting from their home, work, or any other place having Internet access.

For the Internet poll application to be used in a trustworthy manner, a list of requirements must be satisfied: voter authentication, ballot confidentiality, ballot integrity, reliable vote communication, storage and tallying, prevention of multiple voting, protection against attacks on the server as well as the application side. To address these security critical issues during the design of the e-poll application, we used the security ontology we developed, as described in the following paragraphs.

4.2 Secure e-Poll Ontology

Based on the above scenario, we used the methods and tools presented in section 3.2.1 to develop a security ontology that corresponds to the specific context. We will now present the basic concepts of the ontology along with their relationships in triplets of directly connected concepts.

The basic concepts of the proposed ontology are: objective, countermeasure, stakeholder, threat, asset, attacker, and deliberate attack. *Objectives* are the desired properties of the system (e.g. vote anonymity). A *countermeasure* is an action taken to protect an asset against threats (e.g. investigation of incidents). *Stakeholders* are the people that place value on the system (e.g. voter). A *threat* is a potential for a damage of an asset (e.g. fire). *Assets* are pieces of information or resources upon which stakeholders place value (e.g. e-poll application server). An *attacker* is a person, which deliberately damages an asset (e.g. hacker). *Deliberate Attack* is a deliberate human action that damages an asset (e.g. vote corruption).

In Figure 2, we depict the direct relations of the "Objective" concept. Objectives are *defined* by Stakeholders, and they are *threatened* by threats.

Figure 2. Relations of the class "Objective".

Countermeasure's and Asset's direct relations are depicted in Figure 3. We place those two concepts together because they are related with the same two concepts: Threat and Stakeholder. Of course the relations are different: Stakeholders *implement* Countermeasures while they *use* Assets, and Countermeasures *address* Threats while Threats *damage* Assets.

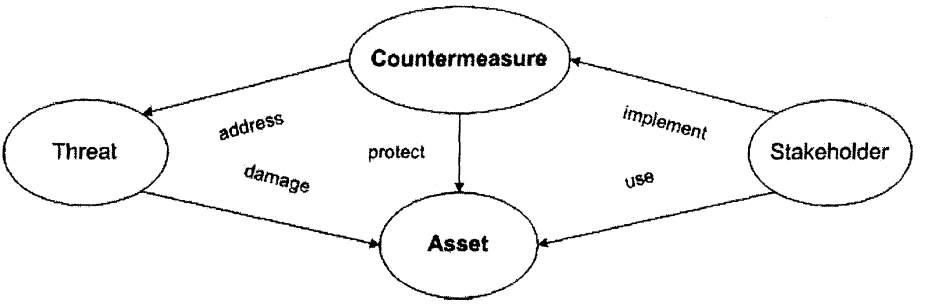

Figure 3. Relations of the classes "Countermeasure" and "Asset".

In Figure 4, we depict the direct relations of the "Threat" notion. Countermeasures *address* Threats and thus *protect* Assets while Threats *threaten* Objectives and *damage* Assets.

This dual "*behaviour*" of the Threat concept can be explained by the rest of Figure 4. As one can see in Figure 4 a Deliberate Attack is a *subclass of* the Threat class and is *realized* by an Attacker.

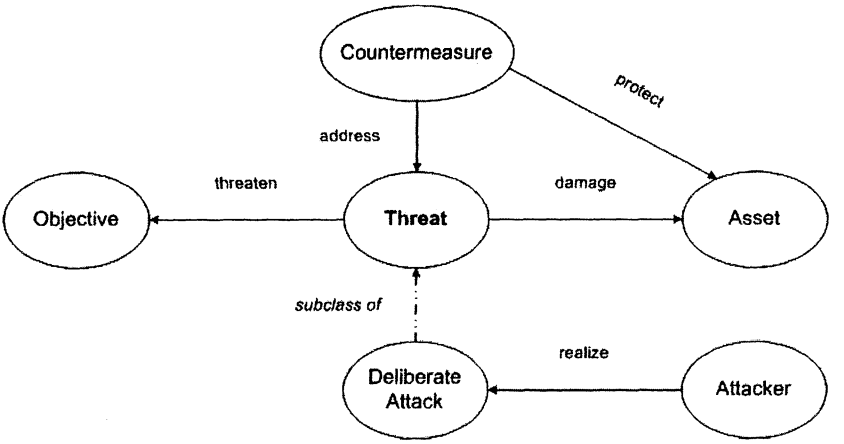

Figure 4. Relations of the class "Threat".

Stakeholder's direct relations are depicted in Figure 5. A Stakeholder *defines* Objectives, *implements* Countermeasures and *uses* Assets (e.g. a voter uses the e-poll system to vote etc.). As stated before, Countermeasures protect Assets.

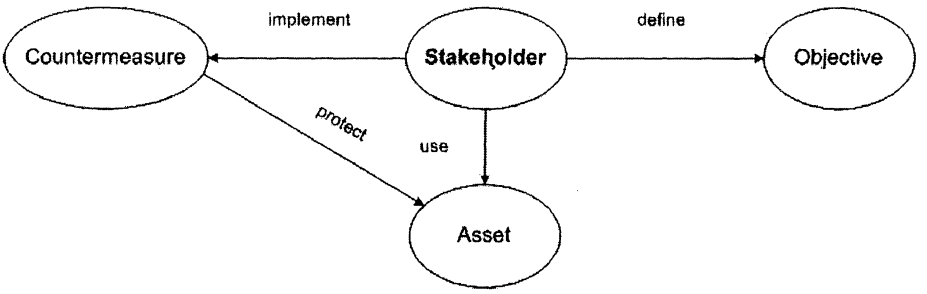

Figure 5. Relations of the class "Stakeholder".

4.3 nRQL Queries and Results

An ontology gains practical value when it is able to give consistent answers to real-world questions. This section lists a number of questions a software developer faced with an e-poll software project is likely to come up with. These questions should not be regarded as exhaustive, but as indicative of what the ontology can deal with and reason about. Each of the questions is firstly expressed formally as an nRQL query, then the result of executing this query is presented, and finally, where appropriate, the rationale behind the result is explained.

Q1. Which are the typical objectives of an e-poll system?

nRQL Query:
```
(retrieve (?obj) (?obj |Objective|))
```
nRQL Result:
```
(((?OBJ |Vote_Anonymity|))
 ((?OBJ |Confidentiality|))
 ((?OBJ |Availability|))
 ((?OBJ |Integrity|))
 ((?OBJ |Voter_Eligibility|))
 ((?OBJ |Accountability|))
 ((?OBJ |Accuracy|)))
```

Q2. Which threats might compromise the vote anonymity objective?

nRQL Query:
```
(retrieve (?threat)
(|Vote_Anonymity| ?threat is_threatened_by|))
```
nRQL Result:
```
(((?THREAT |Impersonation|))
 ((?THREAT |Malicious_Code|))
 ((?THREAT |User_Error|))
 ((?THREAT |OS_Bugs|))
 ((?THREAT |Application_Bugs|))
 ((?THREAT |Attack_On_Voter_Terminal|)))
```

Q3. Which countermeasures protect a voter's personal data?

nRQL Query:
```
(retrieve (?cm)
 (|Voter_Data| ?cm |protected_by|))
```
nRQL Result:
```
(((?CM |Encryption|))
 ((?CM |Access_Control|))
 ((?CM |Certificates|))
 ((?CM |Intrusion_Detection_SW|))
 ((?CM |Malicious_SW_Detection|)))
```

Q4. Which countermeasures can prevent vote replay?

nRQL Query:
```
(retrieve (?cm)
 (?cm |Vote_Replay| |address|))
```
nRQL Result:
```
(((?CM |Identification|))
 ((?CM |Authentication|))
 ((?CM |Auditing|)))
```
Using identification and authentication we can audit the persons that have voted. To prevent them from voting again, we need to check the audit before accepting any vote.

Q5. Should the e-poll organizer be regarded as a threat to vote anonymity?

nRQL Query:
```
(retrieve () (|ePoll_Organizer|
|Compromise_Anonymity| |realizes|))
```
nRQL Result:
```
T(rue)
```
The e-poll organizer should not be trusted more than is necessary, especially since she is in a privileged position. Therefore, he should be regarded as a threat to vote anonymity.

Q6. Which assets are confidential?

nRQL Query:
```
(retrieve (?asset) (and (|Confidentiality|
?threat |is_threatened_by|) (?asset ?threat
|damaged_by|)))
```
nRQL Result:
```
(((?ASSET |Ballot|))
 ((?ASSET |Voter_List|))
 ((?ASSET |Voter_Data|))
 ((?ASSET |Voter_Credentials|))
 ((?ASSET |Vote|))
 ((?ASSET |Cryptographic_Keys|)))
```
To answer this question, we first have to find the possible threats to the confidentiality objective and then list the assets that may be damaged by these threats. For example, confidentiality is threatened by user errors; and a user error may disclose the user's vote.

Q7. Which countermeasures can prevent a hacker but not a vandal?

nRQL Query:
```
(retrieve (?cm) (and (and (|Hacker| ?threat
|realizes|) (not (|Vandal| ?threat |realizes|)))
(?cm ?threat |address|)))
```
nRQL Result:
```
(((?CM |OS_Permissions|)))
```
Operating system permissions are likely to be more effective against a hacker's objectives than against a vandal's, since the vandal's main intention is to irrevocably destroy the system than to just alter its functionality.

Q8. Which threats are not present in an homomorphic encryption voting scheme, but are present in other voting schemes?

nRQL Query:
```
(retrieve (?threat) (and (?schemes
|Voting_Schemes|) (and (?schemes ?threat
|damaged_by|) (not (|Homomorphic_Encryption|
?threat |damaged_by|))))))
```
nRQL Result:
```
(((?THREAT |Vote_Selling|))
 ((?THREAT |DoS_Attack|))
 ((?THREAT |Compromise_Anonymity|)))
```
Most e-voting schemes are based on homomorphic encryption, mixnets or secret sharing among several mutually distrustful election authorities [20]. The stated question aims to illustrate the threat environment in which homomorphic encryption has advantages compared to the other approaches. The justification of the answer is the following:

- In mixnet schemes, when the domain of the possible votes is sufficiently large, a voter may effectively uniquify his/her vote (e.g. by altering the vote's low-significance bits) and sell it to a buyer who had pre-chosen it. This is much harder to do in homomorphic encryption, as only an aggregate (sum) of the votes is disclosed and not the votes themselves.
- As mixnet schemes operate, they necessarily perform a massive amount of communication between the different parties. This makes them much more vulnerable to a denial-of-service attach than other schemes.
- Election schemes based on secret sharing among several mutually distrustful election authorities suffer from the vulnerability that, if a sufficient number of these authorities cooperate, they can link votes to voters. The security of the other schemes is not based on trust among authorities, and hence this vulnerability does not apply to them.

We believe we have demonstrated that the developed ontology is able to give useful answers to the questions a software developer faced with an e-poll software project is likely to come up with.

5. DISCUSSION

Questions similar to the ones presented in the previous section are very likely to come up in any design and development process concerning electronic voting, or electronic government applications. In these cases, designers and developers need to make critical decisions for security related issues. We believe that by employing a specialized ontology, such as the one presented in section three of this paper, all involved parties, including the people that will use the application to organize an electronic voting, as well as the developers, can have a common frame of reference and thus build a common understanding on security related issues.

A security ontology, in particular, such as the one presented in this paper, can be an aid to security critical issues, such as identifying the possible threats to the application that is being developed and deciding on the designated countermeasures to be incorporated at the early stages of designing and developing the application.

It should not go without mention, however, that to address the issue of secure applications effectively and thoroughly, developers need fully developed, specialized ontologies. In this paper we have presented an ontology that can be further developed and enhanced, so as to support the development of other security critical applications.

6. CONCLUSIONS AND FURTHER RESEARCH

In this paper we established that the process of developing a security critical application can be substantially facilitated by employing a specialized security ontology. To do this, we developed such an ontology for the domain of e-poll and demonstrated how software developers working in related software projects can use the ontology to get useful answers for a wide range of security questions. Furthermore, we argued that developing and using similar ontologies brings additional benefits, such as the formation of a common understanding among designers and developers.

We intend to further investigate the possibilities offered by employing security ontologies in this and other security critical contexts.

7. REFERENCES

1. Noy, N.F. and Mc Guinness, D.L. "Ontology Development 101: A Guide to Creating Your First Ontology", Stanford Knowledge Systems Laboratory Technical Report KSL-01-05. (2001)

2. CCTA, CRAMM User Manuals, ver. 5.0, United Kingdom, (2002)
3. IETF and W3C XML Signature Working Group. http://www.w3.org/Signature/
4. OASIS Security Service TC. Security Assertion Markup Language (SAML)
5. http://www.oasis-open.org/committees/security/ (accessed September 2004)
6. Bozsak, E., Ehrig, M., Handschub, S., Hotho: KAON - Towards a Large Scale Semantic Web. In: Bauknecht, K.; Min Tjoa, A.; Quirch¬ma¬yr, G. (Eds.): Proc. of the 3rd International Conference on E-Commerce and Web Technologies, (2002), pp. 304-313
7. Kagal, L., Finin, T., Joshi, A.: "A policy language for a pervasive computing en¬vironment". In IEEE 4th International Workshop on Policies for Dis¬tributed Systems and Networks, (2003)
8. Raskin, V., Hempelmann, C., Triezenberg, K., and Nirenburg, S.: Ontology in Information Security: A Useful Theoretical Foundation and Methodological Tool. In Viktor Raskin and Christian F. Hempelmann, editors, Proceedings of the New Security Paradigms Workshop, New York. ACM, (2001)
9. Chung, L.: Dealing with Security Requirements during the development of Information Systems. CaiSE '93. The 5th Int. Conf of Advanced Info. Systems Engineering. Paris, France, (1993)
10. Mylopoulos, J., Chung L., Nixon, B.: Representing and Using Non-Functional Requirements A Process Oriented Approach. IEEE Trans. Soft Eng., vol. 18. pp. 483-497 (1992)
11. Mouratidis, H., Giorgini, P., Manson, G.: An Ontology for Modelling Security: The Tropos Project. Proceedings of the KES 2003 Invited Session Ontology and Multi-Agent Systems Design (OMASD'03), United Kingdom, University of Oxford, (2003)
12. Liu, L., Yu, E., Mylopoulos, J.: Analyzing Security Requirements as Relationships among Strategic Actors. (SREIS'02), Raleigh, North Carolina, (2002)
13. He, Q., Antón, I., A.: A Framework for modeling Privacy Requirements in Role Engineering. Int'l Workshop on Requirements Engineering for Software Quality (REFSQ) Austria Klagenfurt / Velden (2003)
14. Moffett, D., J., Nuseibeh, A., B.: A Framework for Security Requirements Engineering. Report YCS 368, Department of Computer Science, University of York, (2003)
15. Antón, I., A.: Goal-Based Requirements Analysis. ICRE '96 IEEE Colorado Springs Colorado USA pp.136-144 , (1996)
16. Antón, I., A., Earp, B., J.: Strategies for Developing Policies and Requirements for Secure Electronic Commerce Systems. 1st ACM Workshop on Security and Privacy in E-Commerce (2000)
17. Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S. and Lott, J. KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction and Enforcement. In Proceedings of the IEEE Workshop on Policy (2003)
18. Uszok, A., Bradshaw, J., Jeffers, R. (2004). KAoS: A Policy and Domain Services Framework for Grid Computing and Semantic Web Services. In Proceedings of the Second International Conference on Trust Management (iTrust 2004), Springer-Verlag
19. http://ontology.ihmc.us/ontology.html
20. Smith, W.: Cryptography Meets Voting,
http://www.math.temple.edu/~wds/homepage/cryptovot.pdf
21. Protégé, http://protege.stanford.edu/
22. Racer Inference Engine, http://www.sts.tu-harburg.de/~r.f.moeller/racer/
23. The New Racer Query Language,
http://www.cs.concordia.ca/~haarslev/racer/racer-queries.pdf

24. Denker, G., Access Control and Data Integrity for DAML+OIL and DAML-S, SRI International, USA, (2002)
25. Gruber T. Toward principles for the design of ontologies used for knowledge sharing, in Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer (1993).
26. Web Ontology Language (OWL), http://www.w3.org/2001/sw/WebOnt/
27. O. Lassila, Ralph Swick (eds).: Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999, http://www.w3.org/TR/REC-rdf-syntax/