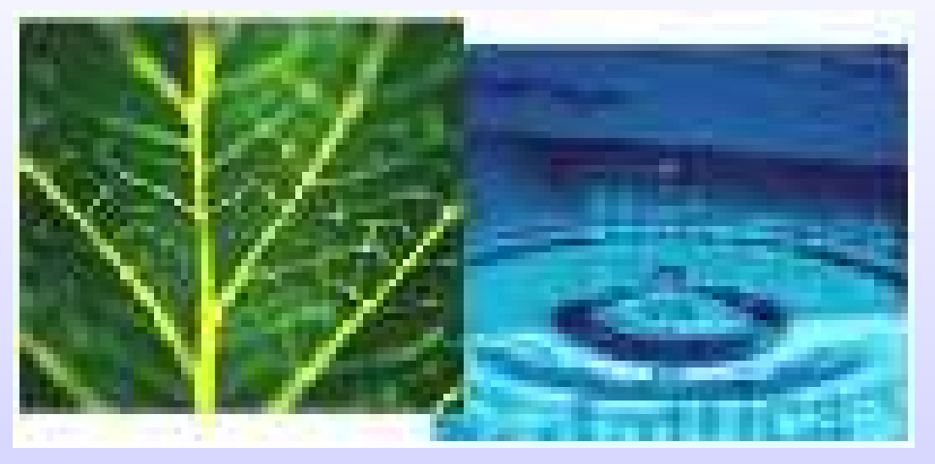
A Methodology for Security Requirements Engineering

Nicola Zannone

Dep. of Information and Communication Technology
University of Trento, Italy
zannone@dit.unitn.it



Abstract

In the last years software engineers have recognized the need to integrate security into the software development processes. However, most of the current methodologies treat security in system-oriented terms. Essentially, they model systems through the policies and access control mechanisms systems support. In contrast, we argue that modeling the organization and the relationships between all involved actors is essential in order to understand the problem of security engineering.
In our previous work we have introduced Secure Tropos, a formal framework for modeling and analyzing security requirements. Secure Tropos is based on the notions of ownership, trust and delegation. We now refine the notions of delegation and trust by distinguishing between permission and execution. We also propose monitoring as a security design pattern in order to guarantee that services are not misused when a trust relation between delegater and delegatee is missing. Finally, we present a CASE tool developed for supporting the entire methodology.

Tropos and Secure Tropos

Tropos [1] is a methodology for developing agent-oriented software. It is founded on models and uses the concepts of actor, goal, task, resource and social dependency for defining the obligations among actors. A goal represents the strategic interests of an actor. A task represents a way of doing something and it is executed in order to satisfy a goal. A resource represents a physical or an informational entity. Finally, a dependency between two actors indicates that one actor depends on another to accomplish a goal, execute a task, or deliver a resource. Tropos is well suited for modeling both an organization and IT systems operating within it, but lacks the ability to capture important aspects of security.
Secure Tropos [2] has been proposed as a formal extension of Tropos, intended for modeling and analysing both functional and security requirements. To simplify terminology, the notion of service is used to refer to a goal, task, or resource. Secure Tropos introduces three new relationships:

- **Ownership** states that an actor is the legitimate owner of a service;
- **Trust** marks a social relationship that indicates the belief of one actor that another actor will not misuse the service he has been granted.
- **Delegation** marks a formal passage of permission.

Refining Delegation and Trust

After a large case study [3], we have sees that the concepts offered by Secure Tropos are the right ones but are too coarse-grained to capture important security facets [4]. Consider the following scenario. Alice is interested in gathering data on student performance, for which she depends on Sam. Bob owns his personal data, such as his academic record. Bob delegates permission to provide information about his academic record to Sam, on condition that his privacy is protected.

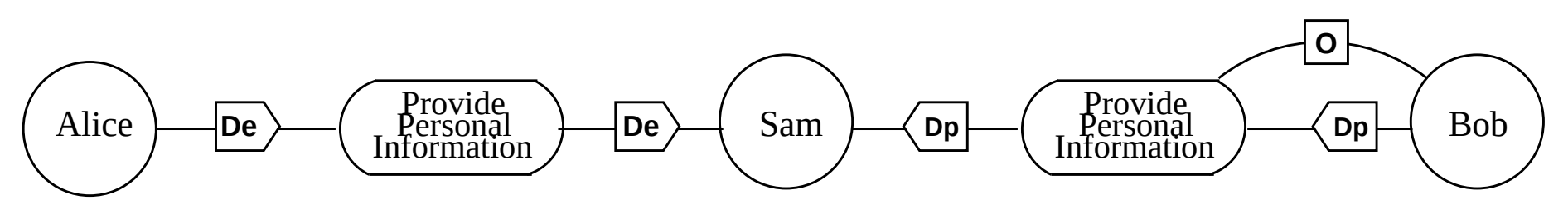


Figure 1: Refining Delegation

There is a difference in the relationships between Alice–Sam and Bob–Sam. This is due to a difference in the type of delegation. Bob *delegates permission* to Sam to provide only the relevant information and nothing else. On the other hand, Alice needs student data to perform her duties, and so she *delegates* the *execution* of providing information to Sam. According to Alice, Sam should fulfill the goal she wants. She is not interested in whether Bob trusts Sam, she simply wants information. On the other hand, Bob worries about who can access his personal information.
To ensure that functional and security requirements are consistent to each other and that security requirements are met, it is essential to distinguish between two types of delegation.

- **Delegation of Permission**: the delegater wants the delegatee to at most fulfill a service. Essentially, the delegatee thinks "I have the permission to fulfill the service (but I do not need to)".
- **Delegation of Execution**: the delegater wants the delegatee to perform the service. Essentially, the delegatee thinks "Now, I have to get this service fulfilled (...let's get started)".

We want to separate the concepts of trust and delegation since we might need to model systems where some actors must delegate a service to other actors they don't trust. Also in this case it is convenient to have a distinction for trust in managing permission and trust in managing execution.

- **Trust of Permission**: an actor trusts that another actor will misuse a service.
- **Trust of Execution**: an actor trusts that another actor will fulfill a service.

The new Secure Tropos concepts allow us for a refinement of the dependency concept. In particular, we can now show how the dependency can be expressed in terms of trust and delegation [5].

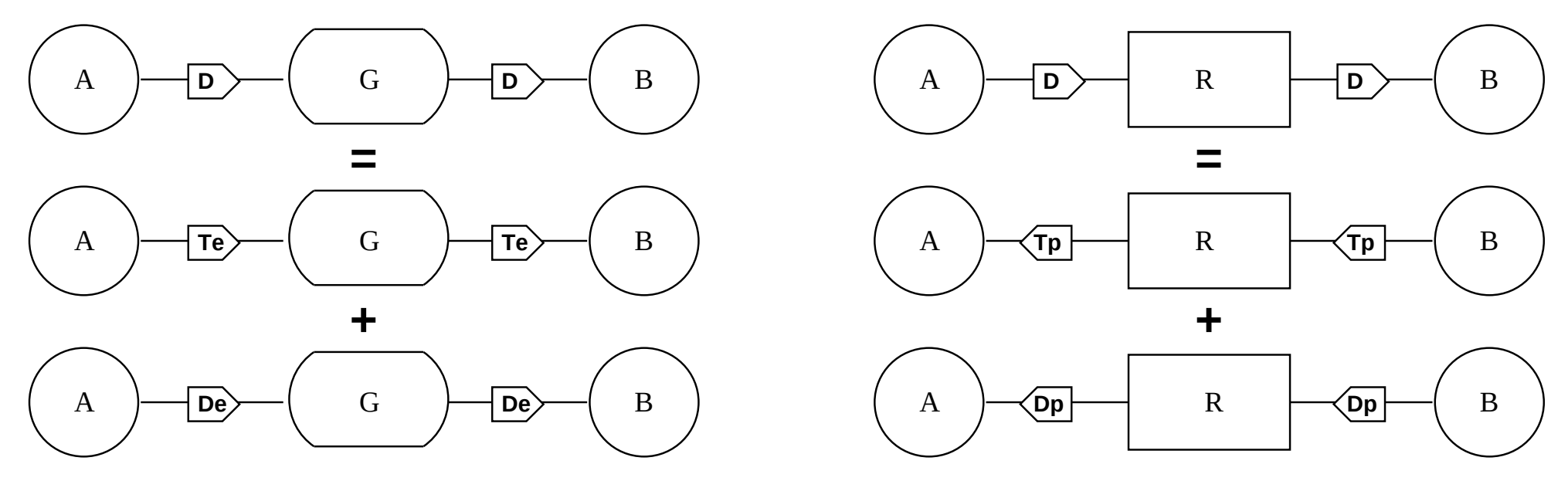


Figure 2: Tropos dependency in terms of Secure Tropos

In particular, when the dependum is a goal or a task we have delegation and trust of execution, whereas when the dependum is a resource we have delegation and trust of permission. Further, dependencies are mapped into differently oriented relations depending on they refer to a goal or task, or to a resource.

Monitoring

Specific situations may impose that some service has to be delegated even in the absence of trust. The existence of distrust can be tolerated with an additional overhead of monitoring the untrusted delegatee. Thus, the objective of the *monitor* is to check for the violation of trust. As for delegation and trust, monitors can be distinguished into **monitor of permission** and **monitor of execution**.

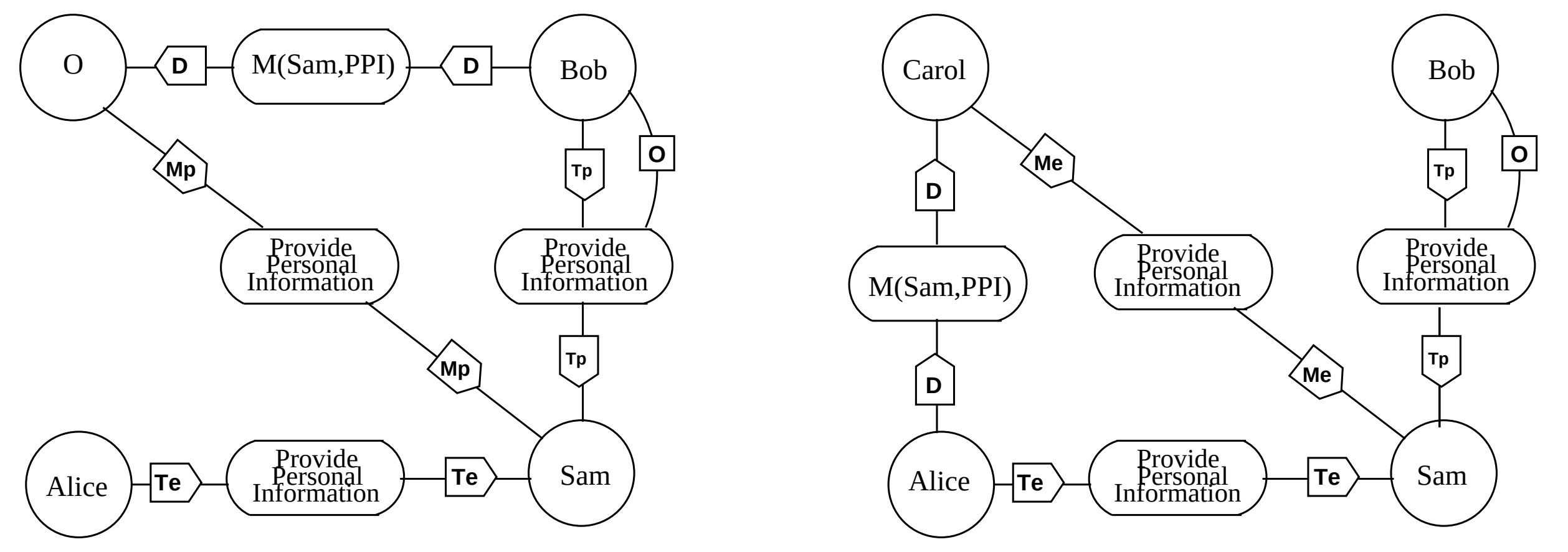

O
D
M(Sam,PPI)
D
Bob
Mp
Tp
O
Provide Personal Information
Provide Personal Information
Mp
Tp
Alice
Te
Provide Personal Information
Te
Sam
Carol
Me
Bob
D
Tp
O
M(Sam,PPI)
Provide Personal Information
Provide Personal Information
D
Me
Tp
Alice
Te
Provide Personal Information
Te
Sam

Figure 3: Monitor of Permission and Monitor of Execution

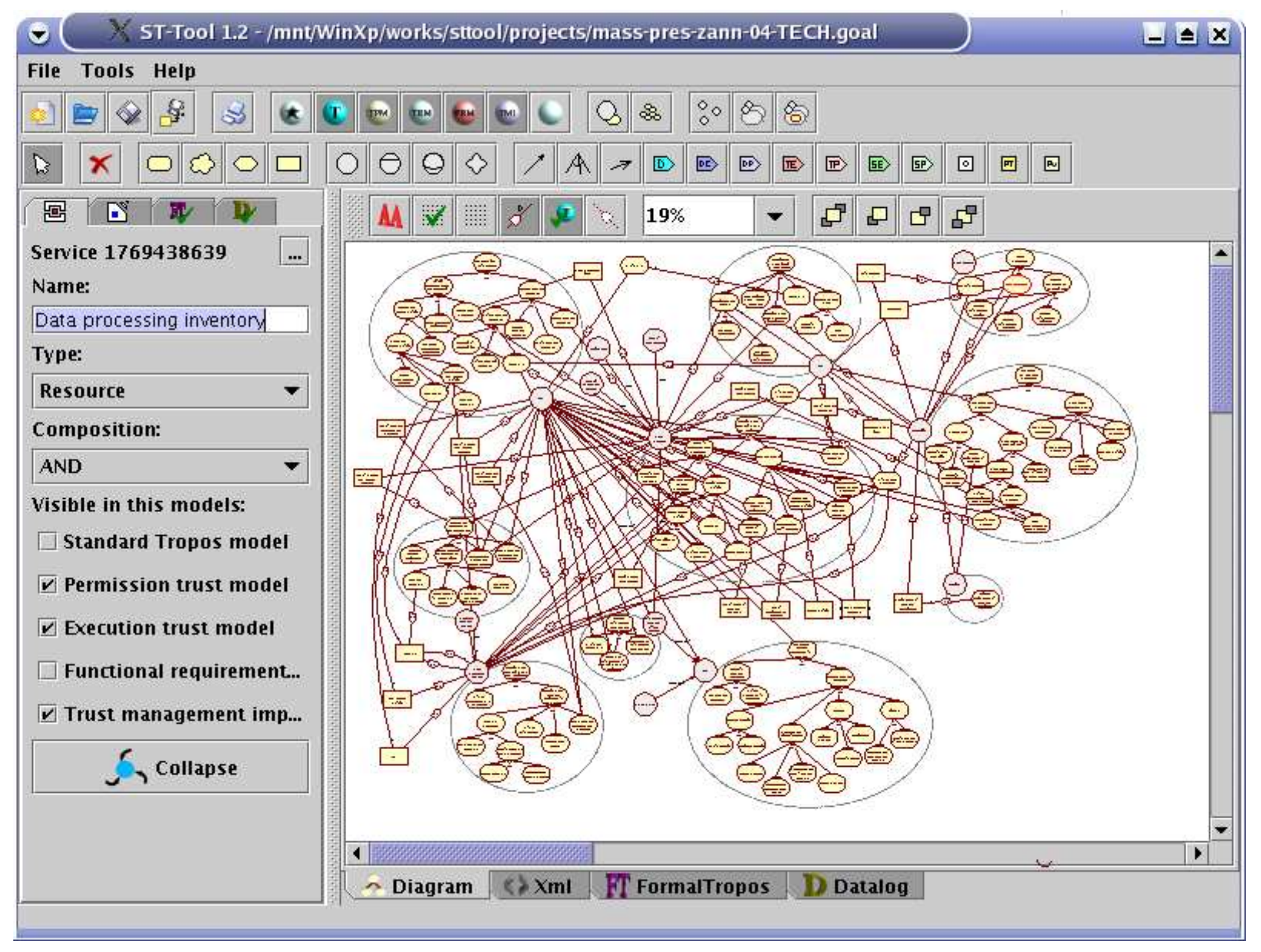
Suppose that Bob believes that Sam will misuse his personal data, but Bob must delegate permission nonetheless. In this case, he depends on ombudsman O for monitoring Sam's behavior. Similarly, if Alice believes that Sam will not provide updated information, she may delegate to her secretary Carol the task of checking up on him to make sure updated data about students is entered into the system.

Formalization

After drawing so many nice diagrams, designers may want to check whether the model satisfies some general desirable properties. As done in [2], we use Datalog to formalize the new concepts in order to automatically verify the correctness and consistency of functional and security requirements.
However, the intuitive descriptions of systems are often incomplete, and so we use *axioms* in order to complete the model for a correct analysis [2]. Then, the consistency of the model is verified through *properties*. Properties are different from axioms. They are constraints that must be satisfied. It is up to designers to decide which properties his own design should respect. Then, if the set of chosen constraints is not consistent, the system is inconsistent, and hence it is not secure.
Due to lack of space, we refer to [4, 5] for the introduction of the formal framework based on Datalog.

ST-Tool

Secure Tropos is supported by a CASE tool called ST-Tool [6]. This tool is implemented in JAVA and provides a user friendly interface within the DLV system for the verification of the correctness and consistency of trust and security requirements in the organization.


ST-Tool 1.2 - /mnt/WinXp/works/sttool/projects/mass-pres-zann-04-TECH.goal
File
Tools
Help
Collapse
Diagram
Xml
FormalTropos
Datalog

Service 1769438639
Name: Data processing inventory
Type: Resource
Composition: AND
Visible in this models:
- [ ] Standard Tropos model
- [✓] Permission trust model
- [✓] Execution trust model
- [ ] Functional requirement...
- [✓] Trust management imp...

19%

Figure 4: ST-Tool screenshot

Specifically, this tool provides advanced modeling and analysis functionalities based on Tropos and Secure Tropos methodologies. Main goals of the tool are:

- Graphical environment: provide a visual framework to draw models;
- Formalization: provide support to translate models into formal specifications;
- Analysis capability: provide a front-end to external tools for formal analysis.

We have already used the tool to model a comprehensive case study on the compliance to the Italian legislation on Privacy and Data Protection by the University of Trento, leading to the denition and analysis of an ISO-17799-like security management scheme [3].

References

[1] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An Agent-Oriented Software Development Methodology. *JAAMAS*, 8(3):203–236, 2004.

[2] P. Giorgini, Fabio Massacci, J. Mylopoulos, and N. Zannone. Requirements Engineering meets Trust Management: Model, Methodology, and Reasoning. In *Proc. of iTrust'04, LNCS 2995*, pages 176–190. Springer-Verlag, 2004.

[3] F. Massacci, M. Prest, and N. Zannone. Using a Security Requirements Engineering Methodology in Practice: The compliance with the Italian Data Protection Legislation. *Comp. Standards & Interfaces*, 27(5):445–455, 2005.

[4] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling Security Requirements Through Ownership, Permission and Delegation. In *Proc. of RE'05*, 2005.

[5] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modelling Social and Individual Trust in Requirements Engineering Methodologies. In *Proc. of iTrust'05, LNCS 3477*, pages 161–176. Springer-Verlag, 2005.

[6] ST-Tool. http://sesa.dit.unitn.it/sttool/.