ST-Tool: A CASE Tool for Modeling and Analyzing Trust Requirements*

P. Giorgini[1], F. Massacci[1], J. Mylopoulos[1,2], A. Siena[1], and N. Zannone[1]

[1] Department of Information and Communication Technology
University of Trento - Italy
{giorgini,massacci,asiena,zannone}@dit.unitn.it
[2] Department of Computer Science
University of Toronto - Canada
jm@cs.toronto.edu

Abstract. ST-Tool is a graphical tool integrating an agent-oriented requirements engineering methodology with tools for the formal analysis of models. Essentially, the tool allows designers to draw visual models representing functional, security and trust requirements of systems and, then, to verify formally and automatically their correctness and consistency through different model-checkers.

1 Introduction

Requirement Engineering is the phase of the software development process that aims at understanding the organization of a system, the goals of system actors and social relationships among them. This phase is critical since a misunderstanding may raise expensive errors during later development stages.

Visual modeling has been recognized as one of the relevant aspects in Software Engineering for aiming the parties involved in the development process at understanding requirements. However, though a graphical notation is useful for human communication, graphical models cannot be used for an accurate system verification. To do this, we need transformation mechanisms to support the translation from graphical model to formal specification languages.

This paper presents ST-Tool, a CASE tool for design and verification of functional, security and trust requirements. It has been designed to support the Secure Tropos methodology [3]. Specifically, this tool provides advanced modeling and analysis functionalities based on the Secure Tropos methodology. 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.

The remainder of the paper is structured as follows. Next (§2) we provide a brief description of Secure Tropos concepts and diagrams. Then, we describe ST-Tool and its components (§3). Finally, we conclude with some directions for future work (§4).

* This work has been partially funded by the IST programme of the EU Commission, FET under the IST-2001-37004 WASP project, by the FIRB programme of MIUR under the RBNE0195K5 ASTRO Project and by PAT MOSTRO project.

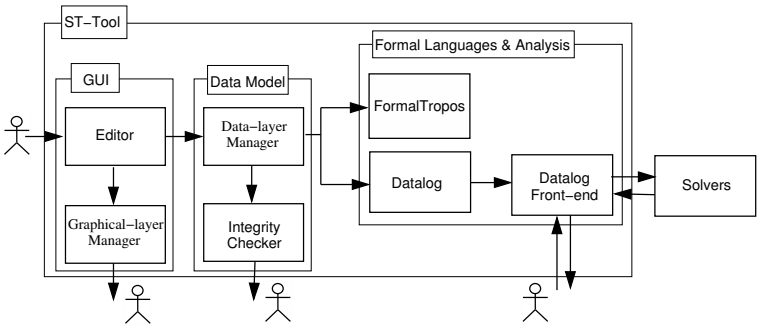

Fig. 1. The Architecture Overview

2 Background

Secure Tropos [3] is an agent-oriented software development methodology, tailored to describe both the organization and the system with respect to functional, security and trust requirements. Secure Tropos extends the Tropos methodology [1] and has the concepts of actor, goal, task, resource and social relationships for defining the obligations of actors to others. A full description of these concepts is provided in [3].

Various activities contribute to the acquisition of a first requirement model, to its refinement into subsequent models: Actor modeling, Permission Trust modeling, Execution Trust modeling, Execution Delegation modeling, Permission Delegation modeling, and Goal refinement (see [3] for full details). A graphical representation of the model obtained following the first five modeling activities is given through four different kinds of *actor diagrams*: *permission trust model*, *execution trust model*, *functional requirements model*, and *trust management implementation*. In these diagrams, actors are represented as circles; goals, tasks and resources are respectively represented as ovals, hexagons and rectangles. In the remainder of the paper, we refer to services when we don't need to distinguish goals, tasks and resources. Goal refinement aims to analyze any goals of each actor, and is conducted by using AND/OR decomposition. A graphical representation of goal refinement is given through *goal diagrams*.

Due to lack of space, we have focused on the key modeling aspects of the framework and refer to [3] for the introduction of the formal framework based on Datalog.

3 Overview of ST-Tool

ST-Tool is a CASE tool that provides a user interface for designing Secure Tropos models, support for translating automatically graphical models into formal specifications and a front-end with external tools for model checking. To manage visual editing features and data management consistency at the same time, we have adopted a two-layer solution: a graphical layer and a data layer. In graphical layer, models are shown as graphs where actors and services are nodes, and relations are arcs. Each visual object refers to a data object. The collection of data objects is the data layer.

ST-Tool is mainly composed of two parts: the ST-Tool kernel and external solvers. ST-Tool kernel has an architecture comprised of three major parts, each of which is

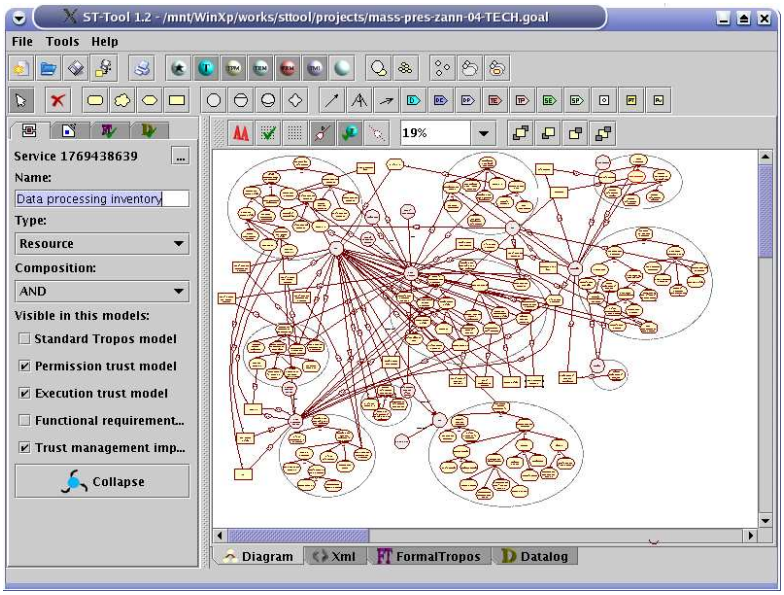

Fig. 2. ST-Tool screenshot

comprised of modules. Next, we will discuss these modules and their interconnections. In Fig. 1, the modules of ST-Tool are shown, their interrelations are also indicated.

The tool provides a graphical user interface (GUI), through which users can manage all the components and functionalities of the tool. A screenshot of the interface is shown in Fig. 2. The GUI's key component is the *Editor Module*. This module allows the user to visually insert, edit or remove graphical objects in the graphical layer and object properties in the data layer. A second GUI component is the *Graphical-layer Manager (GM) Module* that manages graphical objects and their visualization. GM supports goal refinement. A goal diagram is associated to each actor. When systems are very large, it could be difficult to read their models. To this end, GM aids users by supporting two types of collapsing nodes: service and actor collapsing. GM also allows users to display one or more Secure Tropos diagrams listed in Section 2 at the same time.

The *Data-layer Manager (DM) Module* is responsible for building and maintaining data corresponding to graphical objects. For example, DM manages misalignments between social relations and their graphical representation. Actually, GM uses arcs to connect two nodes to each other, while many Secure Tropos relations are ternary. DM rebuilds these relations by linking two appropriate graphical objects (the two arcs) to the same data object (the relation). ST-Tool allows users to save models through the DM module that stores a neutral description of the entire model in `.xml` format files. A support for detecting errors and warnings during the design phase is provided by the *Integrity Checker Module*. Integrity Checker analyzes models stored in the DM module and reports errors such as "orphan relations" (i.e. relations where an arc is missing) and "isolated nodes" (i.e. services not involved in any relations). Warnings are different from errors: they are failure of integrity constraints, like errors, but the designer may

be perfectly happy with a design that does not satisfy them. Integrity Checker reports warnings, for example, when more than one service have the same name. More than one service with the same name are needed to represent delegation and trust chains.

After drawing so many nice diagrams, system designers may want to check whether the models derived so far satisfy some general desirable properties. To support formal analysis, ST-Tool allows an automatic transformation from `.xml` files stored by DM into formal languages. Currently, two languages are supported: Formal Tropos [2] and Datalog. These transformations are performed, respectively, by two different modules: *Formal Tropos Module* and *Datalog Module*. *Datalog Front-end (DF) Module* provides direct support for model checking by using external Datalog solvers, namely ASSAT[3], Cmodels[4], DLV[5] and Smodels[6]. DF guarantees flexibility since it allows users to select which security properties they want to verify [3] and to complete models with additional "ad-hoc" Datalog statements related to the specific domain users are analyzing. Once a user is confident with the model, DF passes the specifications given by Datalog Module, the axioms and properties defined in the Secure Tropos formal framework, and the additional Datalog statements to the external solver. Once the solver ends its job, the output is parsed and presented in a more user-readable format by the DF module.

4 Conclusion

We have presented a tool for modeling and verifying functional, security and trust requirements. We have already used the tool to model a comprehensive case study of the application of the Secure Tropos methodology for the compliance to the Italian legislation on Privacy and Data Protection by the University of Trento, leading to the definition and analysis of an ISO-17799-like security management scheme [4].

Future work will involve a front-end with T-Tool [2] for automatically verifying Formal Tropos specification. Further, Secure Tropos is still under work, so is ST-Tool, too. We are also considering to integrate our tools into the ECLIPSE platform.

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. A. Fuxman, L. Liu, M. Pistore, M. Roveri, and J. Mylopoulos. Specifying and analyzing early requirements: Some experimental results. In *Proc. of RE'03*, page 105. IEEE Press, 2003.
3. P. Giorgini, F. Massacci, J. Mylopoulous, 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.
4. 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*, 2005. To Appear. An extended version is available as Technical report DIT-04-103 at `eprints.biblio.unitn.it`.

[3] http://assat.cs.ust.hk/

[4] http://www.cs.utexas.edu/users/tag/cmodels.html

[5] http://www.dbai.tuwien.ac.at/proj/dlv/

[6] http://www.tcs.hut.fi/Software/smodels/