STS-Tool: Specifying and Reasoning over Socio-Technical Security Requirements

Elda Paja[1], Fabiano Dalpiaz[2], Mauro Poggianella[1], Pierluigi Roberti[1], and Paolo Giorgini[1]

[1] University of Trento, Italy – {elda.paja, mauro.poggianella, pierluigi.roberti, paolo.giorgini}@unitn.it
[2] University of Toronto, Canada – dalpiaz@cs.toronto.edu

Abstract. STS-Tool is the modelling and analysis support tool for STS-ml, our proposed actor- and goal-oriented security requirements modelling language for Socio-Technical Systems (STSs). STS-Tool allows designers to model an STS through high-level primitives, to express security constraints over the interactions between the actors in the STS, as well as to derive security requirements once the modelling is completed. The tool features a set of automated reasoning techniques for (i) checking if a given STS-ml model is well-formed, and (ii) determining if the specification of security requirements is consistent, that is, there are no conflicts among security requirements. We have implemented these techniques using disjuntive datalog programs.

1 The Socio-Technical Security modelling language

The *Socio-Technical Security modelling language* (STS-ml) [1] is an *i** based security requirements modelling language. STS-ml includes high-level organisational primivites such as actor, goal, delegation, etc. A distinguishing feature of STS-ml is the ability to relate security requirements to *interactions*: actors' security needs constrain the interactions they enter into with other actors. Security requirements are mapped to *social commitments* [3]—contracts among actors—that actors in the STS shall comply with at runtime.

STS-ml modelling uses three complementary views, in which the analyst examines different types of interactions among actors.

The formal semantics of STS-ml [2] defines the behavior of STS-ml concepts and relationships, allowing to perform: (i) well-formedness analysis to determine if the model complies with well-formedness rules that are set to preserve the semantics of the STS-ml primitives (e.g., decompositions are not cyclic), and (ii) security analysis, i.e., if there are potential conflicts of security requirements.

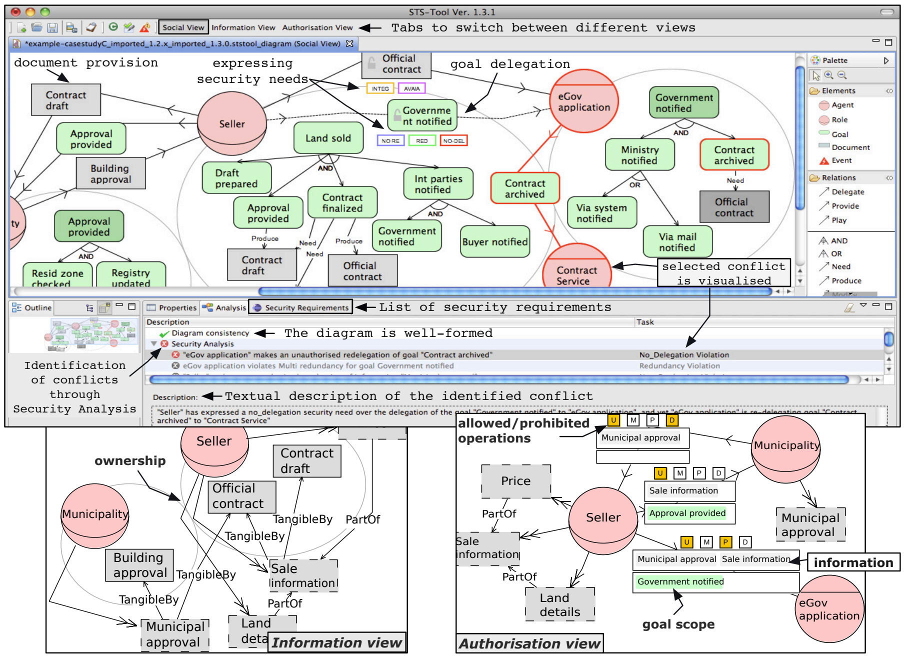
2 STS-Tool

STS-Tool is the modelling and analysis support tool for STS-ml. It is an Eclipse Rich Client Platform application written in Java, it is distributed as a compressed archive for multiple platforms (Win 32/64, Mac OS X, Linux), and it is freely available for download from `http://www.sts-tool.eu`. The website includes extensive documentation including manuals, video tutorials, and lectures. STS-Tool has the following features:

- *Diagrammatic*: the tool enables the creation (drawing) of diagrams. Apart from typical create/modify/save/load operations, the tool also supports:
  - Providing *different views* on a diagram, specifically: *social view*, *information view*, *authorisation view*. Each view shows specific elements and hides others, while keeping always visible elements that serve as connection points between the views (e.g., roles and agents). Inter-view consistency is ensured by for instance propagating insertion or deletion of certain elements to all views.
  - Ensuring diagram validity (online): the models are checked for syntactic/ well-formedness validity while being drawn.
  - Exporting diagrams to different file formats (png, jpg, pdf, svg, etc.).
- *Automatic derivation of security requirements*: security requirements are generated from a model as relationships between a *requester* and a *responsible* actor for the satisfaction of a *security need*. Security requirements can be sorted or filtered according to their different attributes.
- *Automated reasoning*
  - *Offline well-formedness analysis*: some well-formedness rules of STS-ml are computationally too expensive for online verification, or their continuous analysis would limit the flexibility of the modelling activities. Thus, some analyses about well-formedness are performed upon explicit user request. In Fig. 1, offline well-formedness analysis has found no errors.
  - *Security analysis*: verify (i) if the security requirements specification is consistent—no requirements are potentially conflicting; (ii) if the diagram allows the satisfaction of the specified security requirements. This analysis is implemented in disjunctive Datalog and consists of comparing the possible actor behaviors that the model describes against the security requirements. The results are enumerated in a tabular form below the diagram, and rendered visible on the diagram itself when selected (see Fig. 1). A textual description provides details on the identified conflicts.
- *Generating requirements documents*: the modelling process terminates with the generation of a *security requirements document*, which supports the communication between the analyst and stakeholders. This document is customisable: the analyst can choose among a number of model features to include in the report (e.g., including only a subset of the actors, concepts or relations he or she wants more information about). The diagrams are explained in detail providing textual and tabular descriptions of the models. An example report is provided in [3].

The current version of STS-Tool (v1.3.1) is ready for public use. This version of the tool is the result of an iterative development process, having been tested on multiple case studies and evaluated with practitioners [4] in the scope of the

[3] `http://www.sts-tool.eu/Documentation.php`


STS-Tool Ver. 1.3.1
Social View
Information View
Authorisation View
Tabs to switch between different views
document provision
expressing security needs
goal delegation
Seller
eGov application
Municipality
Contract Service
Information view
Authorisation view
ownership
allowed/prohibited operations
information
goal scope
selected conflict is visualised
Identification of conflicts through Security Analysis
List of security requirements
The diagram is well-formed
Textual description of the identified conflict

Description	Task
Diagram consistency	
Security Analysis	
"eGov application" makes an unauthorised redelegation of goal "Contract archived"	No_Delegation Violation
eGov application violates Multi redundancy for goal Government notified	Redundancy Violation

Description: "Seller" has expressed a no_delegation security need over the delegation of the goal "Government notified" to "eGov application", and yet "eGov application" is re-delegating goal "Contract archived" to "Contract Service"

Fig. 1: STS-Tool screenshot: multi-view modelling, automatic derivation of security requirements, and visualisation of security analysis results

FP7 European Project Aniketos [4]. It has proven suitable to model and reason over models of a large size from different domains [2], such as eGovernment, Telecommunications, and Air Traffic Management Control.

Acknowledgments. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no 257930 (Aniketos) and 256980 (NESSoS).

References

1. F. Dalpiaz, E. Paja, and P. Giorgini. Security requirements engineering via commitments. In *Proc. of STAST'11*, pages 1–8, 2011.
2. E. Paja, F. Dalpiaz, and P. Giorgini. Identifying conflicts in security requirements with STS-ml. TR DISI-12-041, University of Trento, `http://disi.unitn.it/~paja/tr-identifying-sec-conflicts.pdf`, 2012.
3. M. P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7(1):97–113, 1999.
4. S. Trösterer, E. Beck, F. Dalpiaz, E. Paja, P. Giorgini, and M. Tscheligi. Formative user-centered evaluation of security modeling: Results from a case study. *IJSSE*, 3(1):1–19, 2012.

[4] `http://www.aniketos.eu`