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

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. We present the latest version of STS-Tool, the modelling and analysis support tool for STS-ml, an actor- and goal-oriented security requirements modelling language for socio-technical systems. STS-Tool allows designers to model a socio-technical system in terms of high-level primitives such as actor, goal, and delegation; to express security constraints over the interactions between the actors; and to derive security requirements once the modelling is done. 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. These techniques have been implemented using disjuntive datalog programs. We have evaluated our tool through various industrial case studies.

1 Introduction

Today's systems are socio-technical, for they are an interplay of social actors (human and organisations) and technical components (software and hardware) that interact with one another for reaching their objectives and requirements [1]. Examples of these systems include healthcare systems, smart cities, critical infrastructure protection, next-generation of military protection and control, air traffic management control, etc.

The participants in a socio-technical system are autonomous, heterogeneous and weakly controllable. This raises up a number of security issues when they interact, especially when interaction involves the exchange of sensitive information: each participant would like to constrain, e.g., the way its information is to be manipulated by others, but has limited ways (due to uncontrollability) to do so.

When dealing with the security problem in socio-technical systems, it is not enough to consider technical mechanisms alone, because social aspects are a main concern. Considering the nature of the security problem in socio-technical systems, we have previously proposed STS-ml [2] (Socio-Technical Security modelling language), an actor- and goal-oriented security requirements modelling language for socio-technical systems, which relies on the idea of relating security requirements to interaction.

STS-ml allows stakeholders (reified as actors) to express *security needs* over interactions to constrain the way interaction is to take place, and uses the concept of *social commitment* [5] among actors to specify security requirements. For example, if a buyer

W. Ng, V.C. Storey, and J. Trujillo (Eds.): ER 2013, LNCS 8217, pp. 504–507, 2013.

sends its personal data to a seller, the buyer may require the data not to be disclosed to third parties. In STS-ml, commitments are used to guarantee the satisfaction of *security needs*: one actor (*responsible*) commits to another (*requestor*) that it will comply with the required *security need*. In the previous example, the seller would commit not to disclose personal data to other parties.

We have previously shown [4] the use of *social commitments* in specifying security requirements. We have explained how STS-Tool [1], the graphical modelling and analysis support tool for STS-ml, supports the automated derivation of commitments.

Practical experiences with STS-Tool have shown [6] that, in large-scale scenarios, the security requirements posed by the various stakeholders are often inconsistent. Coping with such conflicts at requirements time avoids developing a non-compliant and hard-to-change system. In this work, our purpose is to illustrate the automated reasoning techniques (theoretically presented in [3]) that STS-Tool implements to detect inconsistencies among security requirements.

2 Demonstration Content

We illustrate STS-Tool by using an already modelled scenario from a case study on e-Government, developed as part of the EU FP7 project Aniketos. For a more interactive demo, we are going to show the tool in action by refining existing models (*Example 1*).

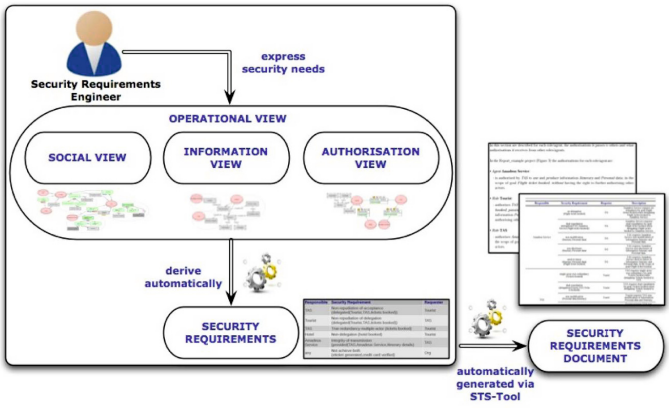

Fig. 1. From the operational view to security requirements

Example 1. Land selling involves finding a trustworthy buyer, and also exchanging several documents with various governmental bodies. The seller needs the municipality to certify that the land is in a residential zone. The land selling process is supported by an eGov application, through which the official contract (including the municipality's certification) is sent to the ministry (who has the right to object) and is archived.

Our demonstration covers the activities described in the following sub-sections.

[1] STS-Tool is freely available for download at `http://www.sts-tool.eu/`

2.1 Modelling with STS-Tool

We show how STS-Tool supports drawing STS-ml models of the system-to-be. STS-ml modelling is carried out by incrementally building three complementary views. As shown in Fig. 1, these views are: *social*, *information*, and *authorisation view*. The *security needs* constrain the interactions among actors. STS-Tool supports multi-view modelling by ensuring inter-view consistency by, for instance, propagating insertion or deletion of certain elements to all views, as well as ensuring diagram validity on the fly (well-formedness validity is checked while the models are being drawn). The tool also supports exporting the diagram (or the different views) to different image formats. We show how the tool supports the modelling process for the illustrating scenario.

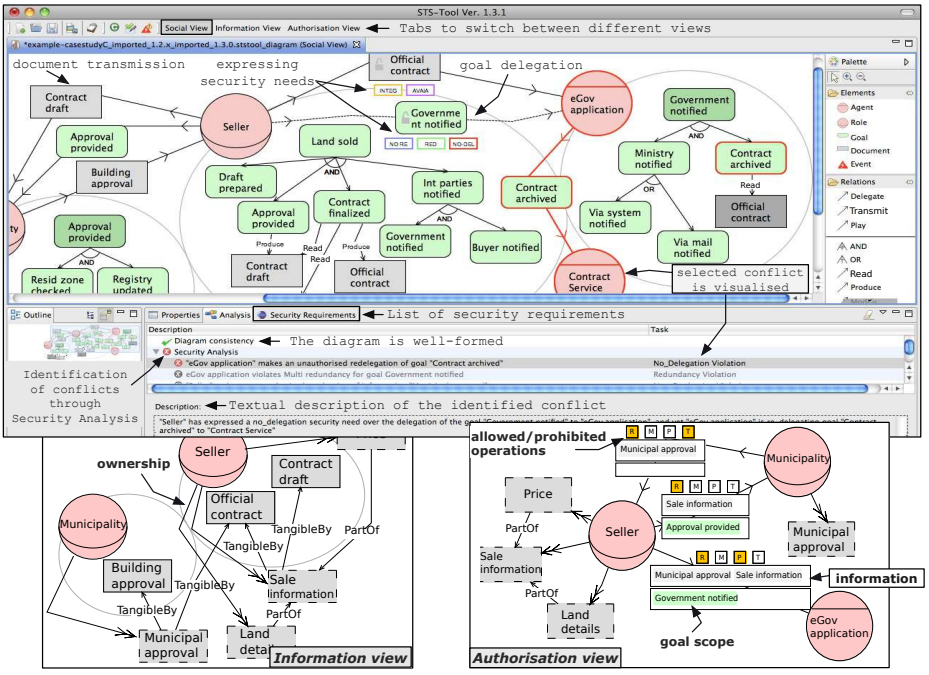

Fig. 2. Modelling, requirements derivation, and security analysis in STS-Tool

2.2 Specifying Security Requirements

We illustrate how security requirements are specified in terms of *social commitments*. Security requirements are automatically generated from a model as relationships between a *requester* and a *responsible* actor for the satisfaction of a *security need*. They can be sorted or filtered according to their different attributes. For instance, filtering the security requirements with respect to the *responsible* actor, highlights the actors that are responsible for satisfying the commitments (security requirements).

2.3 Reasoning about Security Requirements

We show the automated reasoning capabilities implemented in STS-Tool. The formal semantics of STS-ml [3] is defined in terms of possible actions and constraints on actions. STS-Tool supports the following checks: (i) well-formedness analysis to determine if the model complies with syntax restrictions (e.g., no cyclic decompositions), and (ii) security analysis, i.e., if there are potential conflicts of security requirements.

Well-formedness analysis is executed on demand, for its real-time execution would decrease the tool responsiveness. In Fig. 2, no well-formedness error was detected.

Security analysis is implemented in disjunctive Datalog and compares the possible actor behaviors that the model describes, against the security requirements that constrain possible behaviors. The results are enumerated in a table below the diagram (see Fig. 2). A textual description provides details on the identified conflicts.

2.4 Generating the Security Requirements Document

The modelling process terminates with *the generation of the security requirements document* (Fig. 1). 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). The diagrams are explained in detail providing textual and tabular descriptions of the models. The document is organised in sections, which the designer can decide to include or not in the document (see the website for an example).

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

References

1. Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Adaptive Socio-Technical Systems: a Requirements-driven Approach. Requirements Engineering 18(1), 1–24 (2013)
2. Dalpiaz, F., Paja, E., Giorgini, P.: Security requirements engineering via commitments. In: Proceedings of STAST 2011, pp. 1–8 (2011)
3. Paja, E., Dalpiaz, F., Giorgini, P.: Managing security requirements conflicts in socio-technical systems. In: Ng, W., Storey, V.C., Trujillo, J. (eds.) ER 2013. LNCS, vol. 8217, pp. 270–283. Springer, Heidelberg (2013)
4. Paja, E., Dalpiaz, F., Poggianella, M., Roberti, P., Giorgini, P.: STS-tool: Using commitments to specify socio-technical security requirements. In: Castano, S., Vassiliadis, P., Lakshmanan, L.V.S., Lee, M.L. (eds.) ER 2012 Workshops 2012. LNCS, vol. 7518, pp. 396–399. Springer, Heidelberg (2012)
5. Singh, M.P.: An ontology for commitments in multiagent systems: Toward a unification of normative concepts. Artificial Intelligence and Law 7(1), 97–113 (1999)
6. Trösterer, S., Beck, E., Dalpiaz, F., Paja, E., Giorgini, P., Tscheligi, M.: Formative user-centered evaluation of security modeling: Results from a case study. International Journal of Secure Software Engineering 3(1), 1–19 (2012)