STS-Tool: Socio-Technical Security Requirements through Social Commitments

Elda Paja, Fabiano Dalpiaz, Mauro Poggianella, Pierluigi Roberti and Paolo Giorgini
Department of Information Engineering and Computer Science - University of Trento
Via Sommarive, 14 - Trento, Italy - 38123
{*paja,dalpiaz,poggianella,roberti,giorgini*}*@disi.unitn.it*

***Abstract*—Security Requirements Engineering (SRE) deals with the elicitation and analysis of security needs to specify security requirements for the system-to-be. In previous work, we have presented STS-ml, a security requirements modelling language for Socio-Technical Systems (STSs) that elicits security needs, using a goal-oriented approach, and derives the security requirements specification based on these needs. Particularly, STS-ml relates security to the interaction among actors in the STS. In this paper, we present STS-Tool, the modelling and analysis support tool for STS-ml. STS-Tool allows designers to model a STS at a high-level of abstraction, while expressing security needs over the interactions between the actors in the STS, and derive security requirements in terms of social commitments—promises with contractual validity—once the modelling is done.**

***Keywords*-Socio-Technical Systems; Security Requirements; Social Commitments**

I. Introduction

In [1], we argued that security issues in Socio-Technical Systems (STSs) are mainly social, arising from the interaction between actors (humans, organisations, and software/hardware), especially when this involves information exchange. The importance of considering security from a social and organisational perspective has already been acknowledged in the literature [2], [3]. Yet, existing approaches offer high-level concepts that are hard to map to requirements (e.g. see [2]) or build upon technical mechanisms (e.g. see [3]). In our view, SRE should start from high-level abstractions, refining them to derive security requirements.

Based on this, we have proposed STS-ml [1] (Socio-Technical Security modelling language), an actor- and goal-oriented security requirements modelling language for STSs. Notably, STS-ml allows actors to express *security needs* over interactions, to constrain the way interaction is to take place. For instance, a patient might require the hospital not to disclose his medical records to other parties.

STS-ml uses the concept of *social commitments* [4] between actors to represent security requirements. In STS-ml, commitments refer to promises with contractual validity actors exchange with one another to guarantee security properties. For instance, the hospital commits to the patient that his medical records will not be disclosed. The important aspect of social commitments is that they have *contractual validity*. Therefore, in STS-ml they are used as a guarantee for the satisfaction of *security needs*.

In this paper, we illustrate STS-Tool [1], a graphical tool that supports modelling and analysing STSs in terms of the participating actors and their interactions, as well as the automatic derivation of security requirements in terms of *social commitments* once the modelling is performed.

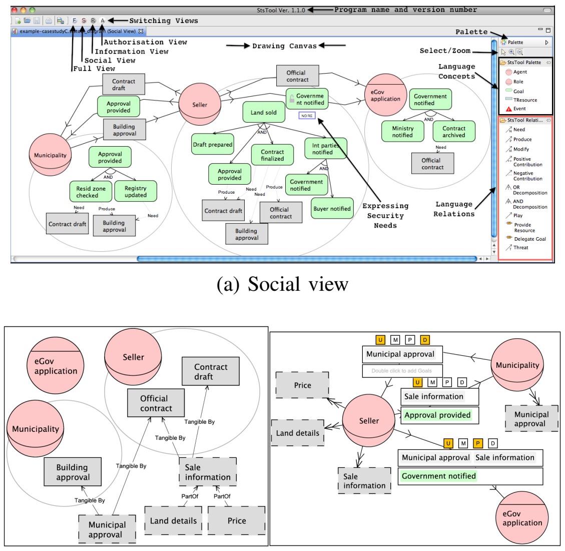
II. STS-ml

STS-ml belongs to the family of goal-oriented requirements engineering frameworks such as Tropos [5] and SI* [3]. It revises the high-level organisational concepts from Tropos, maintaining a minimal set of concepts, including actor, goal, delegation, etc., and uses the concept of social commitments to capture security requirements.

In STS-ml, a commitment refers to a relationship modelling a (contractual) promise made by an actor (*responsible*) to another actor (*requestor*) for the satisfaction of a *security need*. The list of *social commitments* is derived for each *security need* expressed by the stakeholders, and represents the security requirements specification for the system-at-hand. This specification prescribes the security properties stakeholders have to comply with for their interactions (and the STS) to be secure. A distinguishing feature of STS-ml is that it offers multi-view modelling, that is, interactions among actors are represented by focusing on different perspectives (views) at a time. STS-ml consists of three different views: *social*, *information*, and *authorisation* (Figure 1). The social view represents actors intentionality (have goals to achieve) and sociality (interact with others to achieve goals and exchange information). The information view gives a structured representation of actors' information and documents. Finally, the authorisation view shows the authorisations actors grant to others over information they own or have authority to do so.

The *security needs* are expressed through all the three views. The overall model is then automatically mapped to the specification of *security requirements*, which supports the specified *security needs*. Currently, STS-ml supports a number of security needs derived by good security practices with a special focus on interaction, such as non_repudiation, no_delegation, need_to_know, non_modification, non_disclosure, etc.

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



(a) Social view

(b) Information view

(c) Authorisation view

Figure 1: Multi-view modelling with STS-ml

III. STS-Tool

STS-Tool is the CASE tool for STS-ml. It supports modelling activities and the derivation of security requirements as proposed in STS-ml, and it has the following features:

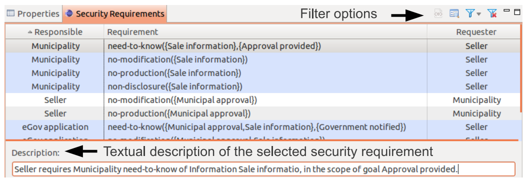
– *Provide different views*: STS-Tool provides different views on a diagram, specifically: *social view*, *information view*, and *authorisation view*. Each view shows specific elements and hides others, while keeping always visible elements that serve as connection points between the views (e.g. roles and agents). Inter-view consistency is ensured by for instance propagating insertion/deletion of certain elements to all views.

– *Export diagram to different file formats*: once the modelling is done, the tool offers designers the possibility to export the diagram (or the different views) to different file formats such as png, pdf, etc.

– *Consistency checking*: the tool helps to create diagrams that follow the semantics of the modelling language, thus improving consistency and validity.

– *Derivation of security requirements*: STS-Tool allows the automatic derivation of security requirements, which are displayed in a tabular form (Figure 2). The security requirements are listed, and they make clear the difference between actors that *request* a given security need from those that are *responsible* for satisfying it.

– *Generating requirements documents*: the tool allows designers to export models and generate automatically a *security requirements document*. The document provides a description of STS-Tool and communicates security requirements by providing details of each STS-ml view, together with their elements. The diagrams are explained in detail providing textual and tabular description of the models.



Responsible	Requirement	Requester
Municipality	need-to-know({Sale information},{Approval provided})	Seller
Municipality	no-modification({Sale information})	Seller
Municipality	no-production({Sale information})	Seller
Municipality	non-disclosure({Sale information})	Seller
Seller	no-modification({Municipal approval})	Municipality
Seller	no-production({Municipal approval})	Municipality
eGov application	need-to-know({Municipal approval,Sale information},{Government notified})	Seller

Seller requires Municipality need-to-know of Information Sale informatio, in the scope of goal Approval provided.

Figure 2: Security requirements via commitments

IV. Conclusion and Future Work

Our work on the STS-ml and tool is ongoing as part of the European research project Aniketos[2]. The current version of the tool is a result of an iterative development process, where the release of internal versions of the tool has been followed by evaluation activities [6].

Future work about STS-Tool includes (i) embedding automated reasoning capabilities to identify inconsistencies in requirements; and (ii) implementing a plugin management system to add functionalities to STS-Tool.

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 *Proceedings of the First Workshop on Socio-Technical Aspects in Security and Trust (STAST'11)*, 2011, pp. 1–8.

[2] L. Liu, E. Yu, and J. Mylopoulos, "Security and Privacy Requirements Analysis within a Social Setting," in *Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE 2003)*. IEEE Computer Society, 2003, pp. 151–161.

[3] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone, "Modeling Security Requirements through Ownership, Permission and Delegation," in *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE 2005)*. IEEE Computer Society, 2005, pp. 167–176.

[4] M. P. Singh, "An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts," *Artificial Intelligence and Law*, vol. 7, pp. 97–113, 1999.

[5] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An Agent-Oriented Software Development Methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004.

[6] 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," *International Journal of Secure Software Engineering*, vol. 3, no. 1, pp. 1–19, 2012.

[2] http://www.aniketos.eu/