Managing Security Requirements Conflicts in Socio-Technical Systems

Elda Paja[1], Fabiano Dalpiaz[2], and Paolo Giorgini[1]

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

Abstract. Requirements are inherently prone to conflicts, for they originate from stakeholders with different, often opposite, needs. Security requirements are no exception. Importantly, their violation leads to severe effects, including privacy infringement, legal sanctions, and exposure to security attacks. Today's systems are Socio-Technical Systems (STSs): they consist of autonomous participants (humans, organisations, software) that *interact* to get things done. In STSs, security is not just a technical challenge, but it needs to consider the social components of STSs too. We have previously proposed STS-ml, a security requirements modelling language for STSs that expresses security requirements as contractual constraints over the interactions among STS participants. In this paper, we build on top of STS-ml and propose a framework that, via automated reasoning techniques, supports the identification and management of conflicts in security requirements models. We apply our framework to a case study about e-Government, and report on promising scalability results of our implementation.

Keywords: security requirements, automated reasoning, requirements models.

1 Introduction

Socio-Technical Systems (STSs) are complex systems composed of autonomous subsystems (*participants*), which are either technical (software) of social (humans and organisations). These subsystems interact to achieve objectives they cannot achieve on their own and to exchange information. STSs are loosely controllable, for their participants are autonomous. Autonomy makes the design of secure STSs a challenging task. For example, if a participant transfers confidential information to another, how does she know that data is kept confidential, without having control?

Goal-oriented approaches to security requirements engineering [7,12,13] offer a suitable abstraction level for the design of secure STSs. They model STSs as a set of actors that are *intentional*—they have objectives—and *social*—they interact with others to achieve their objectives. Unfortunately, their underlying ontology is too abstract to effectively represent real-world information security requirements, which include fine-grained and contradictory authorisations over information entities [1,19].

To overcome this limitation, we have previously proposed the Socio-Technical Security modelling language for STSs (STS-ml) [2]. The language relies on a more expressive ontology and its security requirements are relationships between couples of STS actors, where a *requester* actor requires a *requestee* actor to comply with a security need. Each participant expresses her own requirements. STS-ml models are actor- and goal-oriented, and they represent the business policies of the participants, their security requirements over information and goals, and organisational constraints.

Being specified independently by different actors, policies and security requirements are likely to clash, thus leading to inconsistent specifications that cannot be satisfied by an implemented STS (at least one requirement would be violated). The detection and handling of conflicts between requirements is a hard task [5] (goal-models tend to become huge and complex), and it often requires the usage of automated reasoning techniques. This is true in STS-ml too, for the language supports complex security requirements (due to its expressiveness) and real-world models are typically large [17].

In this paper, we propose a framework for managing conflicts in STS-ml requirements. Our framework suggests to iteratively (i) create STS-ml models for the domain at-hand, (ii) identify conflicts through automated reasoning techniques, and (iii) resolve conflicts. Specifically, we focus here on the first two steps, and leave conflicts resolution for future work. We address two types of conflicts: among security requirements, and between actors' business policies and security requirements. We consider the interplay between different requirements sources: the business policies of individual actors, their security expectations on other actors, and the organisational constraints in the STS.

The contributions of the paper are as follows:

- A revised version of the STS-ml modelling language that includes a richer set of security requirements along with their formal definition.
- A framework for detecting conflicts between (i) security requirements, as well as (ii) among participants' business policies and security requirements.
- An implementation of our framework—using disjunctive Datalog logic programs—as essential part of a CASE tool called STS-Tool.
- Results from a case study, which show the effectiveness of our framework in identifying non-trivial conflicts as well as promising scalability results.

Section 2 reviews related work. Section 3 presents a case study about e-Government. Section 4 presents the current version of STS-ml and its supported security requirements. Section 5 describes our framework for identifying conflicts, while Section 6 evaluates it on the case study and reports on scalability results. Finally, Section 7 presents conclusions and future directions.

2 Related Work

We review related work concerning identifying conflicting requirements, reasoning about security requirements, and methodologies for security requirements engineering.

Conflicting Requirements. The importance of handling conflicts in requirements is well-known in practice and has been acknowledged by the research community [6,18]. We review here the main frameworks in goal-oriented requirements engineering.

Giorgini et al. [8] analyse goal satisfaction/denial by mapping goal models to the satisfiability problem. Their analysis determines evidence of goal satisfaction/denial by using label propagation algorithms. Conflicts occur in case of both positive and negative evidence. Their approach inspired further research. Horkoff and Yu [10] deal with conflicts by demanding resolution to the analyst in an interactive fashion. Fuxman et al. [6] use first-order linear-time temporal logic to identify scenarios with conflicts. KAOS [18] includes analysis techniques to identify and resolve inconsistencies that arise from the elicitation of requirements from stakeholders with different viewpoints.

All these approaches detect conflicting goals. Our approach, instead, treats security requirements as relationships between actors. These approaches could be used to detect inconsistencies within an individual business policy, i.e., within the scope of one actor.

Reasoning about Security Requirements. SI* [7] is a security requirements engineering framework that builds on *i** [20] by adding security concepts, including delegation and trust of execution or permission. SI* uses automated reasoning to check security properties of a model, reasoning on the interplay between execution and permission of trust and delegation relationships. STS-ml supports a more expressive ontology (featuring fine-grained authorisations) to represent information security requirements, and decouples business policies (an actor's goals) from security requirements.

De Landtsheer and van Lamsweerde [3] model confidentiality claims as specification patterns, representing properties that unauthorised agents should not know. They identify violations of confidentiality claims in terms of counterexample scenarios present in requirements models. While their approach represents confidentiality claims in terms of high-level goals, ours represents authorisation requirements as social relationships, and we identify violations by looking at the business policies of the actors.

Security Requirements Methodologies. These are methods to identify possible conflicts, as opposed to using automated reasoning. Secure Tropos [13] models security concerns throughout the whole development process. It expresses security requirements as *security constraints*, considers potential threats and attacks, and provides methodological steps to validate these requirements and overcome vulnerabilities.

Liu et al. [12] extend *i** to deal with security and privacy requirements. Their methodology defines security and privacy-specific analysis mechanisms to identify potential attackers, derive threats and vulnerabilities, thereby suggesting countermeasures.

Haley et al. [9] construct the context of the system, define security requirements as constraints over functional requirements, and develop a structure of satisfaction arguments to verify the correctness of security requirements. This approach focuses mainly on system requirements, while ours is centred on the interaction among actors.

3 Motivating Case Study: Tax Collection in Trentino

Trentino as a Lab (TasLab)[1] is an online collaborative platform to foster ICT innovation among research institutions, universities, enterprises, and public administration in the Trentino province [16]. We focus on a TasLab collaborative project concerning tax collection. The Province of Trento (*PAT*) and the Trentino *Tax Agency* require a system that verifies if correct revenues are collected from *Citizen*s and *Organisation*s, provides a complete profile of taxpayers, generates reports, and enables online tax payments.

[1] `http://www.taslab.eu`

This is an STS in which actors interact via a technical system: citizens and organisations pay taxes online; municipalities (*Municipality*) furnish information about citizens' tax payments; Informatica Trentina (*InfoTN*) is the system contractor; other IT companies develop specific functionalities (e.g., data polishing); the Tax Agency is the system end user; and PAT withholds the land register (information about buildings and lots).

These actors exchange documents that contain confidential information. Each actor has a business policy for achieving her goals, and expects others to comply with her security requirements (e.g., integrity and confidentiality). Organisational constraints (rules and regulations) apply to all actors. Different types of conflict may arise:

- Business policies can clash with security requirements. The Tax Agency's security requirements may include authorising InfoTN to read some data, but they prohibit further transmission of such data. If InfoTN's business policy includes relying upon an external provider to polish data, a conflict would occur.
- Security requirements can be conflicting. For instance, citizens' security requirements may include prohibiting IT companies to access their personal data, while the municipality's (possessing citizens' records) security requirements towards IT companies may specify granting them such authority.
- Organisational constraints may make some requirements unsatisfiable. For instance, a constraint may prohibit private subjects from matching citizens' personal information with tax records. This could create a conflict with the business policy of the data polishing company, which specifies as necessary matching such information.

4 The STS-ml Framework for Security Requirements Modelling

STS-ml is a security requirements modelling framework for designing secure STSs [2]. We present a revised version of STS-ml, which introduces (i) a specialised set of security requirements over information including non-reading, non-modification, non-production, non-disclosure, and non-reauthorisation; (ii) two specialisations of the non-repudiation security requirement; (iii) four specialisations of redundancy; (iv) requirements over the exchange of documents, such as integrity of transmission; and (v) security requirements from rules and regulations, such as separation and binding of duties.

In this section, we formalise the modelling primitives of STS-ml (Section 4.1) and the security requirements that STS-ml supports (Section 4.2).

4.1 Modelling with STS-ml

STS-ml models are constructed by iteratively building three views (*social*, *information*, and *authorisation*), each focussing on different aspects of the STS. An STS-ml model consists of all the elements and relationships from the three views. The multi-view modelling feature of STS-ml promotes modularity and separation of concerns. Figure 1 shows part of the model for our case study (the complete model is available in [14]).

Social View. This represents actors as intentional (having goals that they want to attain) and social (interacting via goal delegations and document exchange) entities. STS-ml supports two types of actors: *agents* are concrete participants, while *roles* are abstract actors, used when the actual participant is unknown. We model InfoTN as an agent,

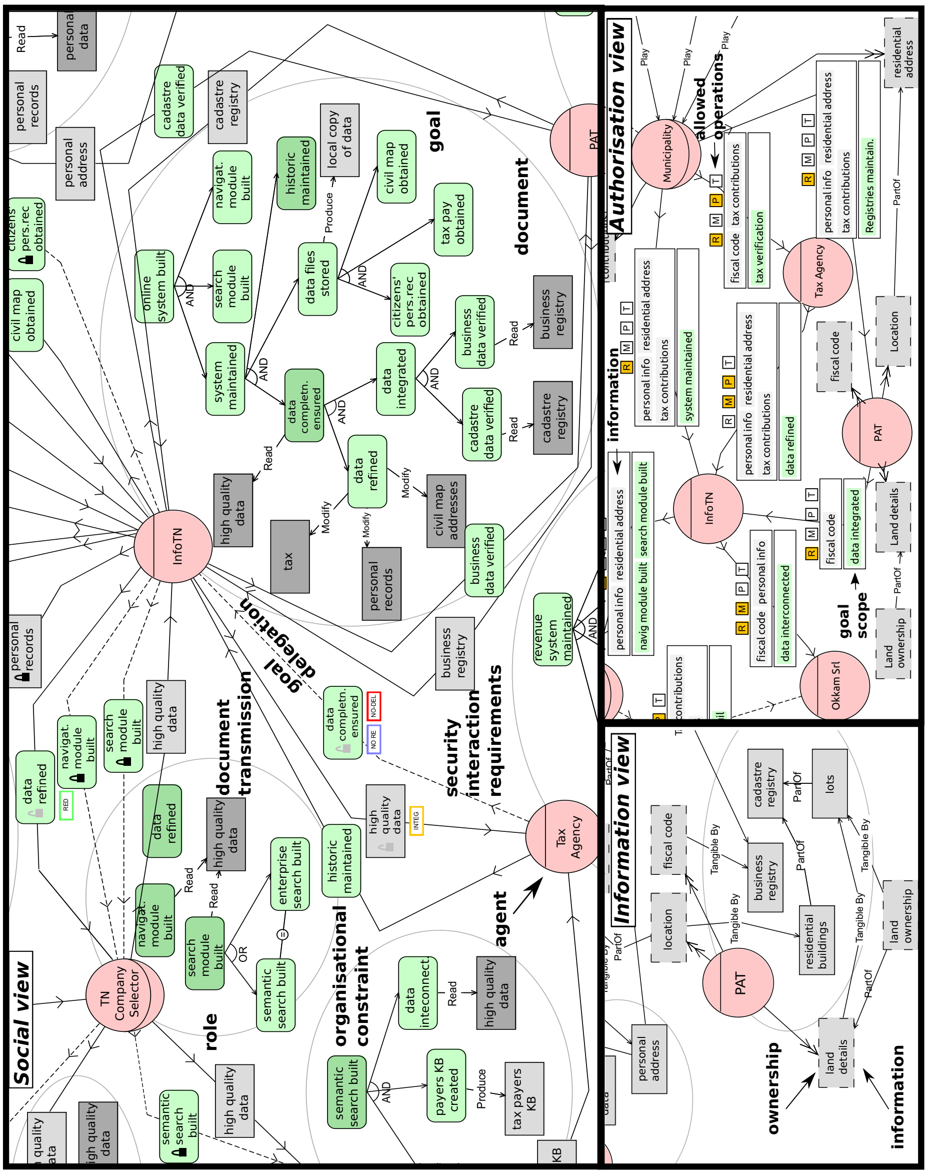

Fig. 1. Partial STS-ml model of the tax collection case study

while Municipality is a role that can be adopted by any municipality in the province (Figure 1). InfoTN has goals online system built, data completn. ensured, and so on. Actors refine their goals through and/or decompositions: InfoTN and-decomposes goal data completn. ensured into data refined and data integrated. Actors may possess documents; they may read, modify, or produce documents while achieving their goals. InfoTN modifies tax for data refined. Actors can transmit documents to others only if they possess the required document. PAT possesses the business registry and transmits it to InfoTN, which reads this document in order to have business data verified.

An actor's **business policy** defines alternative strategies for an actor to achieve her root goals. It is a sub-model of the social view that includes all the goals and documents in the scope of that actor in the social view, the relationships (and/or-decompose, read, modify, and produce) among those goals and documents, as well as goal delegations and document transmissions that start from that actor and end to another actor.

The business policy of TN Company Selector includes goals data refined, navigat. module built, for which document high quality data is read, and search module built that the actor or-decomposes into semantic search built and enterprise search built (Figure 1). These subgoals denote alternative strategies: the actor can choose either of them.

Information View. This defines the relation between information (the data that actors care of) and documents (artifacts that represent information), and represents information ownership. Information can be represented by one or more documents (Tangible By). One or more information pieces can be part of some document. The linkage between information and documents is key to identify which information actors affect, while *reading*, *modifying*, *producing*, or *transmitting* documents. In Figure 1, information on fiscal code owned by PAT is made tangible by document business registry. Also, the information view structures information and documents through part-of relations. For instance, residential buildings is part of cadastre registry.

Authorisation View. This defines the requirements of the actors about who can access information, for what purpose, and which operations can be performed on documents representing information. Authorisations are granted on a set of information $\mathcal{I}$, specifying the operations the authorisee is allowed to perform from the set $\{\mathtt{R}, \mathtt{M}, \mathtt{P}, \mathtt{T}\}$ (where R-read, M-modify, P-produce, T-transmit), and the purpose (which goals) $\mathcal{G}$ for which the authorisation is passed. If $\mathcal{G} = \emptyset$, no restriction about the purpose exists. STS-ml authorisations also specify if *authority to delegate* authority to others is granted (transferrable). The Municipality authorises InfoTN to read (R) information personal info, residential address, and tax contributions, in the scope of goal system maintained granting authority to delegate (graphically, the authorisation has a continuous arrow line).

4.2 Security Requirements in STS-ml

STS-ml aims to ensure information security and secure interaction among participants. The supported requirements originate from the main security aspects in the NIST glossary [11], they have been refined through collaboration with industry [2], and confirmed by evaluation workshops [17]. STS-ml supports three types of security requirements:

- *Interaction (security) requirements* are security-related constraints actors express over their interactions, namely goal delegations and document transmissions.
- *Organisational constraints* determine another range of requirements that constrain the adoption of roles and the uptake of responsibilities.
- *Authorisation requirements* determine authorisations and prohibitions over information specifying whether they can be used, how, for which purpose, and by whom.

[2] Our partners in the EU FP7 Project Aniketos `http://www.aniketos.eu`

An STS-ml model is *consistent* when there are no conflicts between security requirements and business policies.

Interaction (Security) Requirements. STS-ml supports several interaction security requirements on goal delegations and document transmissions (illustrated in Figure 1). Let *Del* stand for delegate(A_1, A_2, G), where A_1 and A_2 are actors, G is a goal:

- *Non-repudiation of delegation/acceptance* [R$_1$: r-not-repudiated-del(A_2, A_1, Del), R$_2$: r-not-repudiated-acc(A_1, A_2, Del)]: A_2 (A_1) requires A_1 (A_2) not to repudiate the delegation (acceptance of the delegation) *Del*. In Figure 1, Tax Agency wants InfoTN not to repudiate the acceptance of delegation for data completn. ensured.
- *Redundancy* [r-red-ensured]: the delegatee has to adopt redundant strategies for the achievement of a delegated goal, either by (a) relying on other actors, or by (b) using alternative internal capabilities. We consider two types of redundancy: (1) *Fallback redundancy*: a primary strategy is selected to fulfill the goal, while other strategies are maintained as backup, and are used only if the primary strategy fails. (2) *True redundancy*: two or more different strategies are executed simultaneously. By intertwining (a-b) with (1-2), STS-ml supports four mutually exclusive redundancy security requirements: (a1) true redundancy single [R$_3$: r-ts-red-ensured(A_1, A_2, G)], (a2) fallback redundancy single [R$_4$: r-fs-red-ensured(A_1, A_2, G)], (b1) true redundancy multi [R$_5$: r-tm-red-ensured(A_1, A_2, G)], and (b2) fallback redundancy multi [R$_6$: r-fm-red-ensured(A_1, A_2, G)]. For instance, InfoTN requires TN Company Selector to ensure true redundancy single for goal data refined.
- *Not-redelegation* [R$_7$: r-not-redelegated(A_1, A_2, G)]: the delegator wants the delegatee not to further delegate goal fulfilment. InfoTN requires TN Company Selector not to redelegate goal search module built, for instance.

Let *Tx* stand for transmit(A_1, A_2, Doc), where A_1, A_2 are actors, *Doc* is a document:

- *Integrity of transmission* [R$_8$: r-integrity-ensured(A_2, A_1, Tx)] requires the sender to guarantee the integrity of *Doc* while transmitting it. Tax Agency requires InfoTN to guarantee the transmission integrity of high quality data.

Organisational Constraints. These requirements do originate from laws, business rules, and regulations. In these constraints, the *STS* is the legal regulatory context for the considered domain. STS-ml supports two basic types of organisational constraints: *Separation of Duties* (SoD), and *Binding of Duties* (BoD). These constraints dictate restrictions over role-to-role relationships as well as agent-to-role and goal-to-goal relationships:

- *Role-based SoD* [R$_9$: r-not-played-both(STS, A, R_1, R_2)]: defines that two roles are incompatible, i.e., no agent A can play both roles R_1 and R_2.
- *Role-based BoD* [R$_{10}$: r-played-both(STS, A, R_1, R_2)]: defines a binding between roles, i.e., if agent A plays role R_1 (R_2), then A must also play R_2 (R_1).
- *Goal-based SoD* [R$_{11}$: r-not-pursued-both(STS, A, G_1, G_2)]: defines incompatible goals, i.e., every agent A must not pursue both goals G_1 and G_2.
- *Goal-based BoD* [R$_{12}$: r-pursued-both(STS, A, G_1, G_2)]: defines that if an agent A pursues goal G_1 (G_2), A should pursue G_2 (G_1) too. An example of this requirement is expressed between goals semantic search built and enterprise search built.

Authorisation Requirements. These are obtained from authorisations in the authorisation view. If an actor A_2 has no incoming authorisation for any information I from

$\mathcal{I}$, then A_2 has a prohibition for I. Such prohibition is an STS-ml authorisation from the information owner to A_2 where no operation is allowed nor authority to transfer is passed. Let *Auth* stand for authorise$(A_1, A_2, \mathcal{I}, \mathcal{G}, \mathcal{OP}, \mathit{TrAuth})$, where A_1, A_2 are actors, $\mathcal{I}$ is a set of information, $\mathcal{G}$ is a set of goals, $\mathcal{OP}$ is the set of operations $\{\mathtt{R}, \mathtt{M}, \mathtt{P}, \mathtt{T}\}$, and *TrAuth* is a boolean value determining if the authorisation is transferrable:

- $\mathcal{G} \neq \emptyset \rightarrow$ *Need-to-know* [R$_{13}$: r-not-ntk-violated$(A_1, A_2, \mathcal{I}, \mathcal{G})$]: A_2 is required not to perform any operation (read/modify/produce) on documents that make some information in $\mathcal{I}$ tangible, for any goals not included in $\mathcal{G}$. The authorisation from Tax Agency to InfoTN is an example: personal info, residential address and tax contributions shall be used only for goal data refined.
- $\mathtt{R} \notin \mathcal{OP} \rightarrow$ *Non-read* [R$_{14}$: r-not-read$(A_1, A_2, \mathcal{I})$] and *Not-reauthorise-read* [R$_{18}$: r-not-reauthorised$(A_1, A_2, \mathcal{I}, \mathcal{G}, \{\mathtt{R}\})$] : A_2 is required not to read documents representing information in $\mathcal{I}$, or authorise others to do so. Tax Agency expresses such requirement on the authorisation for personal info, residential address and tax contributions to InfoTN, for authority to read those information is not granted.
- $\mathtt{M} \notin \mathcal{OP} \rightarrow$ *Non-modification* [R$_{15}$: r-not-modified$(A_1, A_2, \mathcal{I})$] and *Not-reauthorise-modification* [R$_{18}$: r-not-reauthorised$(A_1, A_2, \mathcal{I}, \mathcal{G}, \{\mathtt{M}\})$]: A_1 wants A_2 not to modify documents that include information in $\mathcal{I}$, or authorise others to do so. Municipality expresses a non-modification requirement over the authorisation towards InfoTN, when authorising it to read personal info, residential address and tax contributions.
- $\mathtt{P} \notin \mathcal{OP} \rightarrow$ *Non-production* [R$_{16}$: r-not-produced$(A_1, A_2, \mathcal{I})$] and *Not-reauthorise-production* [R$_{18}$: r-not-reauthorised$(A_1, A_2, \mathcal{I}, \mathcal{G}, \{\mathtt{P}\})$]: A_1 requires A_2 not to produce any documents that include information in $\mathcal{I}$, or authorise others to do so. For example, PAT requires InfoTN not to produce information fiscal code.
- $\mathtt{T} \notin \mathcal{OP} \rightarrow$ *Non-disclosure* [R$_{17}$: r-not-disclosed$(A_1, A_2, \mathcal{I})$] and *Not-reauthorise-disclosure* [R$_{18}$: r-not-reauthorised$(A_1, A_2, \mathcal{I}, \mathcal{G}, \{\mathtt{T}\})$]: A_1 requires A_2 not to transmit to other actors any document that includes information in $\mathcal{I}$, or authorise others to do so. For instance, Municipality expresses such requirement in the authorisation over information fiscal code and tax contributions granted to PAT.
- $\mathit{TrAuth} = \mathtt{false} \rightarrow$ *Not-reauthorised* [R$_{18}$: r-not-reauthorised$(A_1, A_2, \mathcal{I}, \mathcal{G}, \{\mathtt{R}, \mathtt{M}, \mathtt{P}, \mathtt{T}\})$]: A_1 requires A_2 not to further transfer any rights when transferability of the authorisation is false (A_2 cannot transfer any permission on $\mathcal{I}$ and for $\mathcal{G}$). In Figure 1, an example is the authorisation from Citizen to Municipality (dashed arrow).

Among these requirements, R$_1$, R$_2$, and R$_8$ can be verified only at runtime, for they require checking runtime actions that actors carry out (e.g., repudiating a delegation).

5 Detecting Conflicts at Design-Time

STS-ml models can be inconsistent. After describing the possible types of inconsistencies, we show how our reasoning framework detects conflicts among authorisations (Section 5.1), and between business policies and security requirements (Section 5.2).

5.1 Conflicts among Authorisations

We check if the stakeholders have expressed conflicting authorisations. This is non-trivial, for STS-ml models contain multiple authorisations over the same information, and every authorisation expresses a prohibition on the operations for which rights are not transferred. The authorisation from Municipality to InfoTN allows reading (R is selected, Figure 1) information personal info, residential address and tax contributions, but prohibits the modification, production, and transmission of the given information.

Def. 1 (Authorisation conflict). *Two authorisations* $Auth_1 = \mathsf{authorise}(A_1, A_2, \mathcal{I}_1, \mathcal{G}_1, \mathcal{OP}_1, TrAuth_1)$*, and* $Auth_2 = \mathsf{authorise}(A_3, A_2, \mathcal{I}_2, \mathcal{G}_2, \mathcal{OP}_2, TrAuth_2)$*, are conflicting (*$\mathsf{a\text{-}conflict}(Auth_1, Auth_2)$*) iff* $\mathcal{I}_1 \cap \mathcal{I}_2 \neq \emptyset$*, and either:*

1. $\mathcal{G}_1 \neq \emptyset \ \wedge \ \mathcal{G}_2 = \emptyset$ *, or vice versa; or,*
2. $\mathcal{G}_1 \cap \mathcal{G}_2 \neq \emptyset$*, and either (i)* $\mathcal{OP}_1 \neq \mathcal{OP}_2$*, or (ii)* $TrAuth_1 \neq TrAuth_2$*.* □

An authorisation conflict occurs if both authorisations apply to the same information, and either (1) one authorisation restricts the permission to a goal scope, while the other does not (one implies an r-not-ntk-violated requirement, the other grants rights for any purpose); or, (2) the scopes are intersecting, and different permissions are granted (on operations, or authority to transfer). There are two authorisations to InfoTN on personal info, residential address and tax contributions: that from Municipality grants R, but prohibits M, P, and T; that from Tax Agency grants M and P, but prohibits R and T. The authorisations' scopes intersect: the goal data refined of the second authorisation is a subgoal of system maintained of the first authorisation. Thus, they are conflicting with respect to reading, modification, and production of the specified information.

5.2 Conflicts between Business Policies and Security Requirements

Given an STS-ml model without authorisation conflicts, we verify the existence of security requirements that are violated by some actor's business policy. For instance, requirement r-not-modified(Municipality, InfoTN, {personal info}) conflicts with a business policy for InfoTN that includes the relationship modify(InfoTn, goal, personal info).

Business policies define alternative strategies for an actor to fulfil her root goals. Alternatives are introduced by (i) choosing one subgoal in an or-decomposition; and (ii) deciding whether to pursue root goals that are delegated from other actors. Formally:

Def. 2 (Actor strategy). *Given a business policy for an actor A,* $P(A)$*, an actor strategy* $S_{P(A)}$ *is a sub-model of* $P(A)$ *obtained by pruning* $P(A)$ *as follows:*

- *for every or-decomposition, only one subgoal is kept. All other subgoals are pruned, along with the elements that are reachable from the pruned subgoals only (via and/or-decompose, read/produce/modify, transmit, and delegate relationships);*
- *for every root goal G that is delegated to A, G can optionally be pruned.* □

In Figure 1, the business policy for TN Company Selector includes only delegated root goals. One strategy involves keeping only search module built. This goal is or-decomposed; by Def. 2, one subgoal is kept in the strategy (e.g., semantic search built). The read relationship to document high quality data is retained, as well as the document itself. An alternative strategy could, however, involve not building the search module.

We define a *variant* to enable identifying conflicts that occur only when the actors choose certain strategies. A variant combines consistently actors' strategies (each actor fulfills the root goals in her strategy), by requiring that delegated goals are in the delegatee's strategy. Also, a variant includes the authorise relationships in the model.

Def. 3 (Variant). *Let M be an authorisation-consistent STS-ml model, $P(A_1), \ldots, P(A_n)$ be the business policies for all actors in M. A variant of M ($\mathcal{V}_M$) consists of:*

- *a set of strategies $\{S_{P(A_1)}, \ldots, S_{P(A_n)}\}$ such that, for each A_i, A_j, G, if* delegate *(A_i, A_j, G) is in $S_{P(A_i)}$, then G is in $S_{P(A_j)}$, and*
- *all the* authorise *relationships from M.* □

Variants constrain the strategies of the actors. In Figure 1, every variant includes TN Company Selector pursuing goal search module built, for InfoTn's root goal online system built is not delegated to it by others (thus, it has to be in her strategy), and the only possible strategy involves delegating goal search module built to TN Company Selector. Thus, there exists no variant where the latter actor does not pursue that goal.

An STS-ml model can contain more variants. TN Company Selector can choose to achieve search module built through semantic search built or enterprise search built.

Variants enable detecting conflicts between business policies and security requirements. The latter define (dis)allowed relationships for the actors' business policies.

Def. 4 (Bus-Sec conflict). *Given a variant $\mathcal{V}_M$, a conflict between business policies and security requirements exists if and only if, for every actor A:*

- *The strategy of A in $\mathcal{V}_M$ contains one or more relationships that are prescribed by a security requirement requested from another actor A' to A;*
- *The strategy of A in $\mathcal{V}_M$ does not contain any relationships required by some requirement requested from another actor A' to A.* □

Table 1 describes semi-formally how these conflicts are verified for the different types of security requirements that STS-ml supports. Below, we provide some more details.

Security Requirements. Redundancy requirements (R_3 to R_6) can be partially checked. The existence of redundant alternatives is possible, but a variant does not allow distinguishing true and fallback redundancy. Thus, true and fallback redundancy are checked the same way. Single-agent redundancy (R_3 and R_4) is fulfilled if A_2 has at least two disjoint alternatives (via or-decompositions) for G. Multi-actor redundancy (R_5 and R_6) requires that at least one alternative involves another actor A_3. Not-redelegation (R_7) holds if there is no delegation of G or its subgoals from A_2 to other actors in the variant.

Organisational Constraints. R_9 and R_{10} require A *not to* or *to* play two roles through play relationships, respectively. R_{11} is verified if A is not the final performer for both G_1 and G_2 or their subgoals. R_{12} is verified in a similar way, but A has to be the final performer (i.e., does not delegate) for both goals.

Authorisation Requirements. These prescribe relationships that shall not be in A_2's strategy in the variant. Need-to-know (R_{13}) requires no read, modify, or produce relationship on documents that make tangible some information in $\mathcal{I}$ for some goal G' that is not in $\mathcal{G}$ or in descendants of some goal in $\mathcal{G}$. R_{14} to R_{16} are verified if A_2's strategy in the variant includes no read, modify, or produce relationships on documents that make tangible part of $I \in \mathcal{I}$, respectively. Non-disclosure (R_{17}) does a similar check but looking at transmissions. Non-reauthorisation (R_{18}) is fulfilled if there is no authorise relationship from A_2 to others on any operation in $\mathcal{OP}$ over $\mathcal{I}$ in the scope of $\mathcal{G}$.

Table 1. Security requirements and their design-time verification against a variant $\mathcal{V}_M$

Requirement	**Verification at design-time**
Interaction requirements	
R_3 : $\mathsf{r\text{-}ts\text{-}red\text{-}ensured}(A_1, A_2, G)$ R_4 : $\mathsf{r\text{-}fs\text{-}red\text{-}ensured}(A_1, A_2, G)$	Partial. A_2 pursues goals in $\mathcal{V}_M$ that define at least two disjoint ways to support G
R_5 : $\mathsf{r\text{-}tm\text{-}red\text{-}ensured}(A_1, A_2, G)$ R_6 : $\mathsf{r\text{-}fm\text{-}red\text{-}ensured}(A_1, A_2, G)$	Partial. Both A_2 and another actor A_3 support G, each in a different way
R_7 : $\mathsf{r\text{-}not\text{-}redelegated}(A_1, A_2, G)$	$\nexists \mathsf{delegate}(A_2, A_3, G') \in \mathcal{V}_M.\ G' = G$ or G' is a subgoal of G
Organisational constraints	
R_9 : $\mathsf{r\text{-}not\text{-}played\text{-}both}(\mathit{STS}, A, R_1, R_2)$	$\{\mathsf{play}(A, R_1), \mathsf{play}(A, R_2)\} \nsubseteq \mathcal{V}_M$
R_{10} : $\mathsf{r\text{-}played\text{-}both}(\mathit{STS}, A, R_1, R_2)$	$\{\mathsf{play}(A, R_1), \mathsf{play}(A, R_2)\} \subseteq \mathcal{V}_M$
R_{11} : $\mathsf{r\text{-}not\text{-}pursued\text{-}both}(\mathit{STS}, A, G_1, G_2)$	A is not the final performer for both G_1 and G_2 or their subgoals
R_{12} : $\mathsf{r\text{-}pursued\text{-}both}(\mathit{STS}, A, G_1, G_2)$	A is the final performer for both G_1 and G_2 or their subgoals
Authorisation requirements	
R_{13} : $\mathsf{r\text{-}not\text{-}ntk\text{-}violated}(A_1, A_2, \mathcal{I}, \mathcal{G})$	$\nexists \mathsf{read/modify/produce}(A_2, G, D) \in \mathcal{V}_M.\ D$ makes tangible (part of) $I \in \mathcal{I}$ and $G \notin \mathcal{G}$
R_{14} : $\mathsf{r\text{-}not\text{-}read}(A_1, A_2, \mathcal{I})$	$\nexists \mathsf{read}(A_2, G, D) \in \mathcal{V}_M.\ D$ makes tangible (part of) $I \in \mathcal{I}$
R_{15} : $\mathsf{r\text{-}not\text{-}modified}(A_1, A_2, \mathcal{I})$	$\nexists \mathsf{modify}(A_2, G, D) \in \mathcal{V}_M.\ D$ makes tangible (part of) $I \in \mathcal{I}$
R_{16} : $\mathsf{r\text{-}not\text{-}produced}(A_1, A_2, \mathcal{I})$	$\nexists \mathsf{produce}(A_2, G, D) \in \mathcal{V}_M.\ D$ makes tangible (part of) $I \in \mathcal{I}$
R_{17} : $\mathsf{r\text{-}not\text{-}disclosed}(A_1, A_2, \mathcal{I})$	$\nexists \mathsf{transmit}(A_2, A_3, D) \in \mathcal{V}_M.\ D$ makes tangible (part of) $I \in \mathcal{I}$
R_{18} : $\mathsf{r\text{-}not\text{-}reauthorised}(A_1, A_2, \mathcal{I}, \mathcal{G}, \mathcal{OP})$	$\nexists \mathsf{authorise}(A_2, A_3, \mathcal{I}, \mathcal{G}, \mathcal{OP}') \in \mathcal{V}_M.\ \mathcal{OP}' \subseteq \mathcal{OP}$

6 Implementation and Evaluation

STS-Tool [3] [15] supports STS-ml modelling, inter-view consistency, requirements document generation, as well as automated analysis for conflict detection. Under the hood, the tool encodes STS-ml models into disjunctive Datalog programs to support our reasoning [14]. The current version of STS-Tool is the result of an iterative development process that intertwined evolutions of the language and continuous evaluations [17].

We evaluate our framework in two ways: (i) we show its effectiveness in identifying conflicts in our case study, and (ii) we conduct experiments to assess its scalability.

Findings from the Case Study. We first modelled the case study using STS-Tool (Figure 1) to then run the automated analysis to identify *authorisation conflicts*. The analysis returned severl conflicts that we had not identified during the modelling, including:

- *On authority to produce*: Tax Agency authorises InfoTN to produce documents with information personal info, residential address and tax contributions to obtain refined data, whereas Municipality requires read-only access, and not production.

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

- *On authority to modify*: InfoTN grants Okkam Srl the authority to modify documents with information personal info to obtain interconnected data, whereas TN Company Selector requires no document representing this information is modified.

These conflicts, which went unnoticed while modelling, originate from the stakeholders' authorisation policies. The former conflict can be resolved by negotiating the provision of adequate rights with the Municipality, while the latter can be fixed by revoking the authorisation, given that Okkam Srl does not need it (from the social view).

After fixing authorisation conflicts, we used the tool's capabilities to identify *Bus-Sec conflicts*. This activity provided us with further useful insights:

- r-not-redelegated: TN Company Selector relies on Okkam Srl to build a semantic search module (delegation of semantic search built). However, while relying on TN Company Selector, InfoTN wants this company to build the search modules, requiring it not to redelegate goal semantic search built. This interaction requirement is in conflict with the business policy on delegating semantic search built.
- r-not-modified: Engineering Tribute Srl makes an unauthorised modification of Citizen's personal info, violating the authorisation requirement r-not-modified specified by Citizen and passed on by TN Company Selector.
- r-not-produced: Citizen makes an unauthorised production of addresses, for this information is owned by the Municipality and no authorisation is granted to Citizen.
- r-not-reauthorised: Citizen wants only the Municipality to read and produce her personal info and does not allow transfer of authority, however the Municipality further authorises InfoTN to read documents with this information.
- r-pursued-both: goals semantic search built and enterprise search built should be pursued by the same actor, since a r-pursued-both normative requirement is specified between these goals. A conflict occurs because TN Company Selector is not the final performer for both goals (semantic search built is delegated to Okkam Srl).

These conflicts are due to the different policies of the companies. They can be resolved through trade-offs [4] between business policies and security requirements.

Scalability Study: Design. We have investigated how the reasoning execution time is affected by the size of the model. Taking the model in Figure 1 as a basic building block, we cloned it to obtain larger models in terms of (i) *number of elements* (nodes and relationships); and (ii) *number of variants*. The latter is motivated by our reasoning techniques, which generate STS-ml model variants (Def. 3). For details, see [14].

We ran tests on models with *zero*, *medium* and *high* variability, by customising the decomposition types in the original model. For each model, we ran the analysis 7 times, discarded the fastest and slowest executions, and computed the average execution time.

Scalability Study: Results. We have conducted our experiments on a DELL Optiplex 780 desktop PC, Pentium(R) Dual-Core CPU E5500 2.80GHz, 4Gb DDR3 399, powered by Windows 7. Below, we detail the results (summarized in Figure 2) and draw conclusions for the two scalability dimensions we have considered:

- *Number of elements* [Figure 2(a)]: we present results for all the conflict types we can detect, i.e., authorisation conflicts, violation of interaction and authorisation requirements, as well as of organisational constraints. As noticeable by the plot,

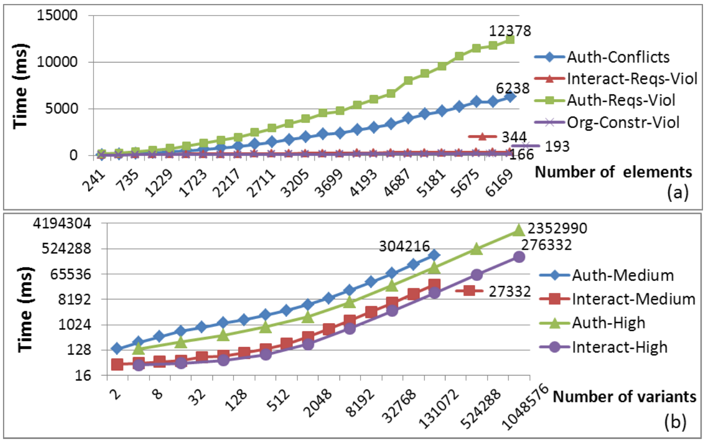

Fig. 2. Scalability results with increasing number of elements and number of variants

all techniques scale very well (linear growth). Furthermore, the tool managed to reason over our extra-large models (6,000+ elements) in about twelve seconds.

– *Number of variants* [Figure 2(b)]: this dimension affects execution time the most. We show only violations of authorisation and interaction requirements; the other checks do not increase the number of variants. While the growth is still linear in the number of variants, it is exponential in the number of elements (the model with 1,048,576 variants consists of 2,500 elements). Some medium-variability tests take longer than high-variability because, for a given number of variants, a medium-variability model contains twice the elements than a high-variability model.

The results are very promising, especially when considering that the size of real world scenarios is smaller than the extra-large models we produced with our cloning strategy.

7 Conclusions

We have proposed a framework to detect conflicts in security requirements adopting a socio-technical perspective on requirements models. Our framework is based on a revised version of STS-ml [2], the security requirements modelling language for STSs. STS-ml supports a rich set of security requirements: interaction security requirements, fine-grained authorisation requirements, and organisational constraints.

We have shown how to detect two types of conflicts: (1) among authorisation requirements; and (2) between business policies and security requirements. We have illustrated the effectiveness of our conflict identification techniques on an industrial case study, and we have reported on promising scalability results of our implementation.

Our future work includes: (1) devising further reasoning techniques to identify conflicts among security requirements (so far, we identify conflicts only among authorisation requirements); and (2) exploring possible ways to resolve the identified conflicts.

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. Bertino, E., Jajodia, S., Samarati, P.: A flexible authorization mechanism for relational data management systems. ACM Transactions on Information Systems 17(2), 101–140 (1999)
2. Dalpiaz, F., Paja, E., Giorgini, P.: Security requirements engineering via commitments. In: Proc. of STAST 2011, pp. 1–8 (2011)
3. De Landtsheer, R., van Lamsweerde, A.: Reasoning about confidentiality at requirements engineering time. In: Proc. of FSE 2005, pp. 41–49 (2005)
4. Elahi, G., Yu, E.: A goal oriented approach for modeling and analyzing security trade-offs. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 375–390. Springer, Heidelberg (2007)
5. Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J., Nuseibeh, B.: Inconsistency handling in multiperspective specifications. IEEE TSE 20(8), 569–578 (1994)
6. Fuxman, A., Pistore, M., Mylopoulos, J., Traverso, P.: Model checking early requirements specifications in tropos. In: Proc. of RE 2001, pp. 174–181 (2001)
7. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Modeling security requirements through ownership, permission and delegation. In: Proc. of RE 2005, pp. 167–176 (2005)
8. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Reasoning with goal models. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 167–181. Springer, Heidelberg (2002)
9. Haley, C.B., Laney, R., Moffett, J.D., Nuseibeh, B.: Security requirements engineering: A framework for representation and analysis. IEEE TSE 34(1), 133–153 (2008)
10. Horkoff, J., Yu, E.: Finding solutions in goal models: An interactive backward reasoning approach. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 59–75. Springer, Heidelberg (2010)
11. Kissel, R.: Glossary of key information security terms. Technical Report IR 7298 Rev 1, NIST (2011)
12. Liu, L., Yu, E., Mylopoulos, J.: Security and privacy requirements analysis within a social setting. In: Proc. of RE 2003, pp. 151–161 (2003)
13. Mouratidis, H., Giorgini, P.: Secure Tropos: A security-oriented extension of the tropos methodology. IJSEKE 17(2), 285–309 (2007)
14. Paja, E., Dalpiaz, F., Giorgini, P.: Identifying conflicts in security requirements with STS-ml. Technical Report DISI-12-041, University of Trento (2012)
15. Paja, E., Dalpiaz, F., Poggianella, M., Roberti, P., Giorgini, P.: STS-Tool: socio-technical security requirements through social commitments. In: Proc. of RE 2012, pp. 331–332 (2012)
16. Shvaiko, P., Mion, L., Dalpiaz, F., Angelini, G.: The TasLab portal for collaborative innovation. In: Proc. of ICE 2010 (2010)
17. Trösterer, S., Beck, E., Dalpiaz, F., Paja, E., Giorgini, P., Tscheligi, M.: Formative user-centered evaluation of security modeling: Results from a case study. IJSSE 3(1), 1–19 (2012)
18. van Lamsweerde, A., Darimont, R., Letier, E.: Managing conflicts in goal-driven requirements engineering. IEEE TSE 24(11), 908–926 (1998)
19. Whitman, M.E., Mattord, H.J.: Principles of Information Security, 4th edn. Course Technology Press (2011)
20. Yu, E.: Modelling strategic relationships for process reengineering. PhD thesis, University of Toronto, Canada (1996)