Data & Knowledge Engineering 98 (2015) 123–143

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Modelling and reasoning about security requirements in socio-technical systems

Elda Paja [a,*], Fabiano Dalpiaz [b], Paolo Giorgini [a]

[a] Dept. of Information Engineering and Computer Science, University of Trento, Italy
[b] Dept. of Information and Computing Sciences, Utrecht University, The Netherlands

ARTICLE INFO

Available online 14 July 2015

Keywords:
Security requirements
Automated reasoning
Requirements models

ABSTRACT

Modern software systems operate within the context of larger socio-technical systems, wherein they interact—by exchanging data and outsourcing tasks—with other technical components, humans, and organisations. When interacting, these components (*actors*) operate autonomously; as such, they may disclose confidential information without being authorised, wreck the integrity of private data, rely on untrusted third parties, etc. Thus, the design of a secure software system shall begin with a thorough analysis of its socio-technical context, thereby considering not only technical attacks, but also social and organisational ones.

In this paper, we propose the STS approach for modelling and reasoning about security requirements. In STS, security requirements are specified, via the STS-ml requirements modelling language, as contracts that constrain the interactions among the actors in the socio-technical system. The requirements models of STS-ml have a formal semantics which enables automated reasoning for detecting possible conflicts among security requirements as well as conflicts between security requirements and actors' business policies. We apply STS to a case study about e-Government, and report on promising scalability results of our implementation.

1. Introduction

Most of today's software systems are part of larger socio-technical systems [1–3] that include humans and organisations in addition to technical components. Socio-technical systems consist of autonomous subsystems (*participants*) that interact (via messages) to achieve their objectives and to exchange information. Examples include smart cities, e-commerce, healthcare systems, etc. When designing a secure software system, one should start from an analysis of the enclosing socio-technical system, and the security requirements should include social and organisational considerations, in addition to technical aspects [4,5].

Many methods for security requirements have been proposed (see [6] for a review). While a considerate number of mainstream approaches does not take into account social aspects, focusing strictly on technical mechanisms [7–12], there are others that recognize the importance of considering security from a social and organisational perspective [4,5].

Goal-oriented approaches [13,14,4] explicitly acknowledge the importance of social factors, and they rely on a model of a socio-technical system that comprises actors (representing participants) that are *intentional*—they have objectives—and *social*—they interact with others to achieve their objectives. However, security requirements are expressed at a too high level of abstraction (e.g., [4,13]), making them difficult to operationalise to technical requirements for the system-to-be. Also, the underlying ontologies are not expressive enough to effectively represent real-world security requirements.

[*] Corresponding author.
E-mail addresses: elda.paja@unitn.it (E. Paja), f.dalpiaz@uu.nl (F. Dalpiaz), paolo.giorgini@unitn.it (P. Giorgini).

http://dx.doi.org/10.1016/j.datak.2015.07.007

To overcome this limitation, we have previously proposed the Socio-Technical Security modelling language (STS-ml) for socio-technical systems [15]. STS-ml models are also actor- and goal-oriented, but the language relies on a more expressive ontology and its security requirements are relationships between couples of actors in the socio-technical system, where a *requester* actor requires a *requestee* actor to comply with a security need. Each participant can express its own requirements, and the STS-ml model represents the business policies of the participants as well as their security requirements over information and goals.

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 socio-technical system (one or more requirements would be necessarily violated). The detection and handling of conflicts between requirements is a difficult task [16] (goal-models tend to become large 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 the STS approach for *modelling and reasoning* about security requirements in socio-technical systems. The STS approach is founded on the STS-ml modelling language. We provide a comprehensive account on STS-ml, the formal framework for automated reasoning, and their implementation in STS-Tool. This work builds on top of previous research: the initial version of the STS-ml [15], demonstrations of the tool that supports STS-ml [18,19], and the work on conflict identification mechanisms [20]. Specifically, the contributions of this paper are as follows:

- an extended version of the STS-ml modelling language including a rich set of security requirements that covers the major information security aspects;
- a formal framework that supports automated reasoning techniques for detecting conflicting requirements at design time;
- an implementation of the reasoning techniques and evaluation with a case study on e-Government to assess their usefulness and scalability.

The rest of the paper is organised as follows. Section 2 presents the case study on e-Government. Section 3 presents the extended version of STS-ml and its supported security requirements. Section 4 introduces the formal framework for identifying conflicts, while Section 5 evaluates it on the case study and reports on the findings and scalability results. Section 6 reviews related work. Finally, Section 7 presents conclusions and future directions.

2. 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 province of Trentino [21]. 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 *Citizens* and *Organisations*, provides a complete profile of taxpayers, generates reports, and enables online tax payments.

This is a socio-technical system 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).

Thus, the income tax system is not monolithic: its operation depends by the successful interaction among taxpayers, the municipality, InfoTN, the Tax Agency, and the TasLab website. 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. Conflicts concerning security are possible:

- *Security requirements can be conflicting*: 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.
- *Business policies can clash with security requirements*: Tax Agency's security requirements may include authorising InfoTN to read some data, but prohibiting transmission of such data. If InfoTN's business policy includes relying upon an external provider to polish data, a conflict would occur.

3. The STS-ml language for security requirements modelling

The initial version of STS-ml was presented in [15]. Here, we present an extended version of STS-ml, which not only supports a richer set of security requirements, but comes with a precise semantics of the language constructs.

STS-ml provides a set of modelling primitives to represent socio-technical systems, their participants, their important assets, and their security needs. Most importantly, STS-ml employs multi-view modelling by arranging its modelling primitives into three different yet complementary views that, together, represent the overall STS-ml model for the system-to-be. The advantage of this

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

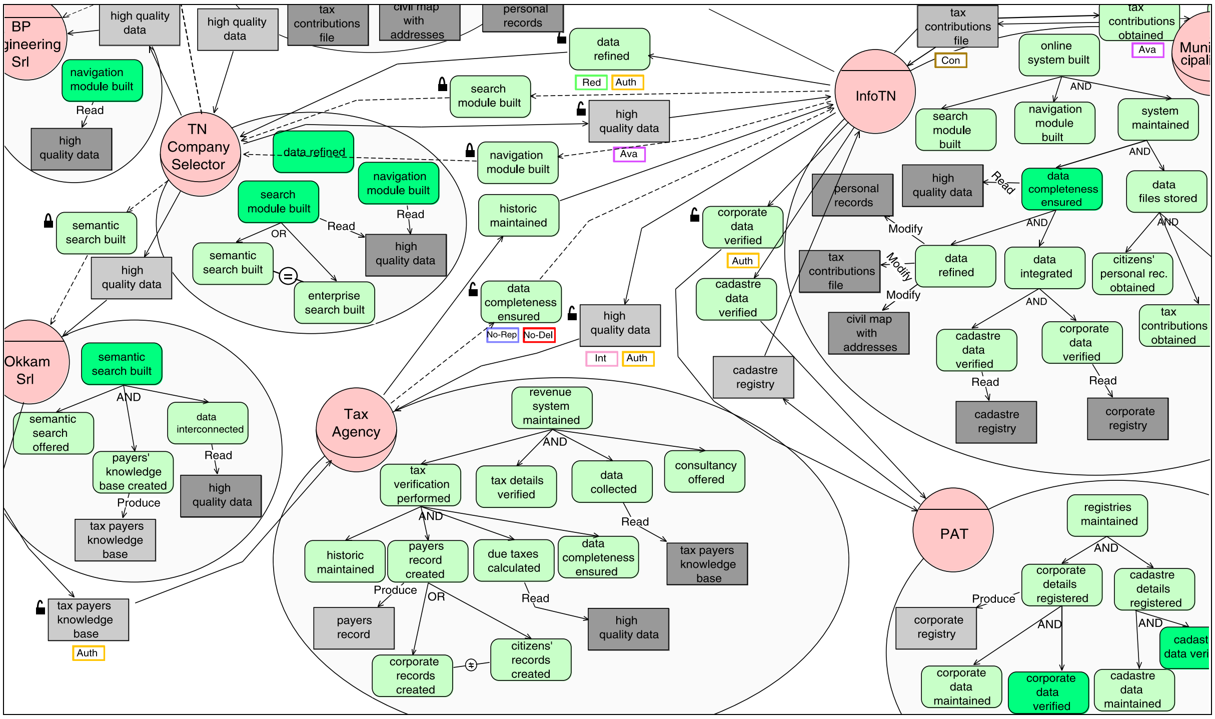


Fig. 1. Partial STS-ml social view of the tax collection scenario.

choice is that the analyst(s) can examine a specific aspect of the system at a time. Clearly, these views are interrelated—they describe facets of the *same* system—and changes in one view may require modifications in the other views.

Based on these premises, STS-ml modelling is performed by focusing on three views, each representing a specific perspective of the system. Together, these views provide a comprehensive representation of the important security dimensions of a socio-technical system:

- *Social view*: captures *social* and *organisational* aspects, in which actors are modelled along with their objectives and interactions with others;
- *Information view*: represents informational assets, that is, information actors own and want to protect;
- *Authorisation view*: models the flow of permissions and prohibitions regarding information among the actors that participate in the system.

Inter-model consistency, which is threatened by the existence of multiple views, is ensured by STS-Tool (details in Section 5).

This separation of concerns facilitates also the collaboration between requirements analysts and security engineers, allowing each of them to focus on modelling the aspects related to their expertise. For instance, a requirements analyst would typically contribute mostly with information regarding the social and organisational aspects (social view), while a security engineer would mainly model the permissions and prohibitions among actors (authorisation view).

The following subsections describe which modelling primitives are used to create an STS-ml model through each view, and show the resulting models from the modelling of each view (Section 3.1), to then present the set of supported security requirements (Section 3.2).

3.1. Modelling with STS-ml

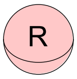
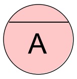
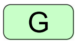
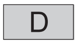
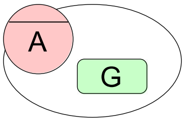
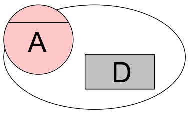
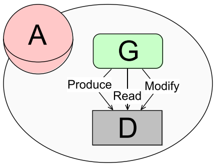
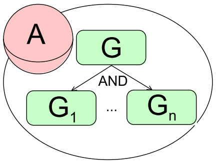
We present the modelling primitives through the various views, to better explain what the focus in each of them is.

3.1.1. Social view

The *social view* represents the stakeholders and their interactions. Stakeholders are intentional—they have objectives they aim to attain—, and social—they interact with others to achieve their objectives. A partial social view for the motivating case study is shown in Fig. 1, while the primitives in the social view are summarised in Tables 1 and 2. Details about security requirements and their representation, already visible in Fig. 1, will be provided in Section 3.2.

3.1.1.1. Concepts and intentional relationships. STS-ml supports the modelling of stakeholders (system participants) in terms of *roles* and *agents* (see Table 1). For instance, in Fig. 1, we have modelled the company *Informatica Trentina* (`InfoTN`) as an agent, since we know it will be part of the system already at design time. `TN Company Selector`, on the other hand, is modelled as a role, for

Table 1
Social view: concepts and intentional relationships.

Graphical notation	Syntax and description
Concepts	
	role (R): an abstract characterization used to model a class of participants, defining a set of responsibilities for the said participants (e.g., professor, student)
	agent (A): a concrete participant known to be in the system already at design time (e.g., John, Laura)
	goal (G): represents stakeholders' objectives (e.g., exam taken)
	document (D): represents information (e.g. transcripts file)
Intentional relationships	
	wants (A, G): actor A wants to achieve goal G, models the actor's intention to achieve the goal (e.g. student wants to pass the exam)
	possesses (A, D): actor A possesses document D, models the possession of the document by an actor (e.g. professor has the exam paper)
	reads/modifies/produces (A, G, D): actor A reads/modifies/produces document D when fulfilling goal G (e.g. professor reads student exams to grade them)
	decomposes (A, G, $\mathcal{S}$, DecT): A decomposes root goal G into sub-goals from $\mathcal{S}$, where $\mathcal{S} = \{G_1,\ldots,G_n\}$ and $\lvert\mathcal{S}\rvert \geq 2$, and the decomposition is of type DecT, such that DecT $\in$ {and, or} (e.g. a student passes the exam if she sits in for the exam and gets a grade higher than 65%)

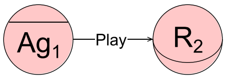
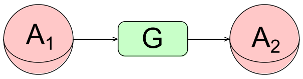
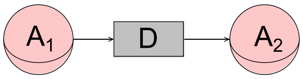
we do not know yet which company will take over this responsibility, but we do know the responsibilities encapsulated in this role. Whenever we do not need to distinguish between role and agent (for properties or relationships applying to both), we will use the general term *actor* to refer to either a role or an agent.[2]

STS-ml captures stakeholders' objectives via the concept of *goal* (see Table 1). Goals are an actor's *intentional assets* for actors enter the socio-technical system with the aim to achieve their desired goals, and thus, their goals are their intentional assets. Goals are different from activities and processes: goals are part of the motivational component of an agent, and express *why* and *what* an agent aims to achieve, rather than *how* the agent is to achieve its objectives. High-level (*root*) goals are refined (decomposed) into subgoals to express how an actor intends to fulfil the root goals. STS-ml supports (i) *and-decomposition*—the achievement of all the subgoals implies the achievement of the decomposed goal, and (ii) *or-decomposition*—the achievement of at least one subgoal implies the achievement of the decomposed goal. In the TasLab case study, `InfoTN` wants to achieve goal `online system built`, which is and-decomposed into `search module built`, `navigation module built`, and `system maintained`.

STS-ml supports the modelling of *documents*—containing information,[3] see Table 1—that actors manipulate while achieving their goals. STS-ml represents the relationships among an actor's goals and documents within that actor's scope (the grey oval around the role or agent shape, see Table 1). Actors might *possess* documents, and they might *read*, *modify*, or *produce* documents while achieving

[2] In Tables 1–4, for illustrative purposes, we interchangeably use roles and agents.
[3] A detailed explanation on information and documents is provided in Section 3.1.2.

Table 2
Social view: social relationships.

Graphical notation	Syntax and description
	plays (Ag$_1$, R_2): models the adoption of roles by agents, i.e., agent Ag$_1$ plays role R_2
	delegates (A_1;A_2;G): models the transfer of responsibilities between two actors; a delegator actor delegates the fulfilment of a goal (delegatum) to a delegatee actor, i.e., actor A_1 delegates goal G to actor A_2
	transmits (A_1;A_2;D): specifies the exchange of documents between two actors; a sender actor transmits a document to a receiver actor, i.e., actor A_1 transmits document D to actor A_2

their goals. Possession implies the actor's capability of transmitting the document to other actors. Note that possession is different from ownership, for it refers only to the actor having a document, not necessarily owning it. Documents are secondary assets, for attackers can try to alter or destroy them to threaten the primary asset that they contain, i.e., information. In Fig. 1, `InfoTN` possesses document `local copy of data`, it reads document `high quality data` to have `data completeness ensured`, while it modifies `personal records` for goal `data refined`.

3.1.1.2. Social relationships. The social view represents the social relationships between actors in the considered socio-technical system. The language supports three social relationships: *play*, *goal delegation* and *document transmission* (see Table 2). In the TasLab case study, we do not have any knowledge at design time about agents adopting the identified roles, i.e., no examples of *play*. An example of *goal delegation* is that of InfoTN delegating goal `search module built` to `TN Company Selector`, while an example of *document transmission* is that of `InfoTN` transmitting the document `high quality data` to `Tax Agency`, see Fig. 1.

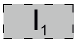
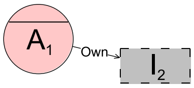
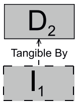
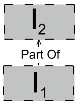
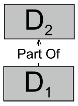
3.1.2. Information view

STS-ml distinguishes between *information*—the data that actors own, care about, and may deem confidential—and its representation via *documents*. The latter, intended in a broad sense (e.g., an email or a text message are documents too), are the means through which actors transfer information. This is a distinguishing feature of STS-ml, and it is the key enabler for specifying sophisticated security requirements concerning authorisations (see Section 3.1.3).

While the social view includes documents and their transmission, we do not know what is the informational content of the exchanged documents. This is useful in an analysis of security requirements to determine whether information was exchanged among authorised users for instance. For this, the *information view* represents the informational content of the documents in the social view. The model indicates information entities, their owners, and provides a structured representation of information and documents.

The concepts and relationships of the information view are summarised in Table 3. Documents are possessed by actors and STS-ml represents this through the `possesses` relationship, which denotes that the actor has a document in the socio-technical system. For instance, actor `Citizen` possesses documents `personal data` and `personal address`, while actor `PAT` possesses the four documents in its scope, namely `corporate registry`, `cadastre registry`, `residential buildings`, and `land lots`. Note

Table 3
Information view: concepts and relationships.

Graphical notation	Syntax and description
	information (I_1): informational entities (e.g. name, student grade)
	owns (A^1; I^2): actor A^1 is the legitimate owner of information I^2, i.e., it has full rights over that information (e.g. students own their personal information)
	makes-tangible (I_1, D_2): document D_2 materializes information I_1 (e.g. transcripts materialize information about course results)
	part-of-i (I_1, I_2): information I_1 is part of information I_2 (e.g. course description is part of course syllabus)
	part-of-d (D_1, D_2): information D_1 is part of information D_2 (e.g. student file is part of student registry)

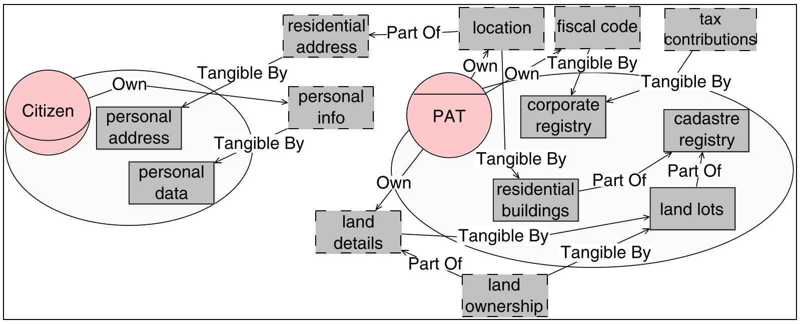


Fig. 2. Partial STS-ml information view of the tax collection scenario.

that details about possession are preserved by the social view, but new documents can be added when creating the information view as well. Actors and documents are the primitives used to connect the social with the information view.

Information entities are denoted by the concept of `information`. Information ownership is denoted with an arrow labelled *own* from the actor to the information it owns. For instance, actor `Citizen` is the owner of `personal info` (see Fig. 2). Note that information may have one or more legitimate owners, although we do not have an example of such situation in our motivating case study. In STS-ml, the relation owns indicates that one actor (a role or an agent) is the legitimate owner of some information and can freely manipulate it (has full rights over the said information), as well as decide to transfer rights about it to others.

The relationship `Tangible By`, a labelled arrow from an information to a document, indicates that the document contains/represents the information. In other words, when the document is read, the reader becomes aware of the information. For instance, information `personal info` is made tangible by `Citizen's personal data` (Fig. 2). Information can be represented by one or more documents (through multiple `Tangible By` relationships). On the other hand, one or more information entities can be made tangible by the same document. For instance, `location and fiscal code` are information entities owned by `PAT`; `location` is made tangible by `residential buildings`, while `fiscal code` together with `tax contributions` is made tangible by document `corporate registry`, see Fig. 2.

We emphasize again that all operations over information are done through documents in the social view.

Another feature of the information view is to support composite information (documents). The structuring of information and documents is done via `part-of` relationships, allowing designers to build a hierarchy of information entities and documents, respectively. For instance, this allows representing that information location is part of the information `residential address`, while document `land lots` is part of document `cadastre registry` in Fig. 2.

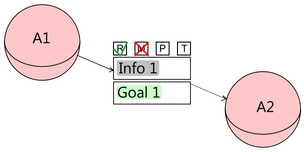
3.1.3. Authorisation view

This model shows the authorisations that actors grant to others over the informational entities that are represented in the information view.

In STS-ml, an authorisation is a directed relationship between two actors, where one actor grants or prohibits certain rights to another actor on some information, see Table 4. Authorisations are defined along four orthogonal dimensions:

- *Allowed/prohibited operations*: the transferred rights relate to different operations that an actor can perform (tick symbol) or is prohibited (cross symbol) to perform on the information (see Fig. 3) through the documents that represent it. STS-ml supports four basic operations that originate from how information is manipulated in the social view through documents: authority to read (`R`) goes in parallel with the read relation; authority to modify (`M`) relates to the modify relation, authority to produce (`P`) is reflected by the produce relation, and authority to transmit (`T`) corresponds to the document transmission relation.
- *Information*: authorisation is granted over at least one information entity. Given the structuring of information in the *information view* via `part-of` relationships, authorising some actor over some information means that the actor is authorised for parts of information as well.

Table 4
Authorisation view: social relationships.

Graphical notation	Syntax and description
	authorises (A_1, A_2, $\mathcal{I}$, $\mathcal{G}$, $\mathcal{OP}$, TrAuth): actor A_1 authorises/prohibits actor A_2 to perform operations $\mathcal{OP}$ ({R, M, P, T} ∪ {$\overline{\text{R}}$, $\overline{\text{M}}$, $\overline{\text{P}}$, $\overline{\text{T}}$}) on the information in $\mathcal{I}$, in the scope of the goals in $\mathcal{G}$ and allows (prohibits) A_2 to transfer the authorisation to others if TrAuth is true (false); (e.g. A_1 authorises A_2 to read information Info 1 (R is checked), but prohibits modifying such information (M crossed over), in the scope of Goal 1 allowing transferability (continuous arrow line))

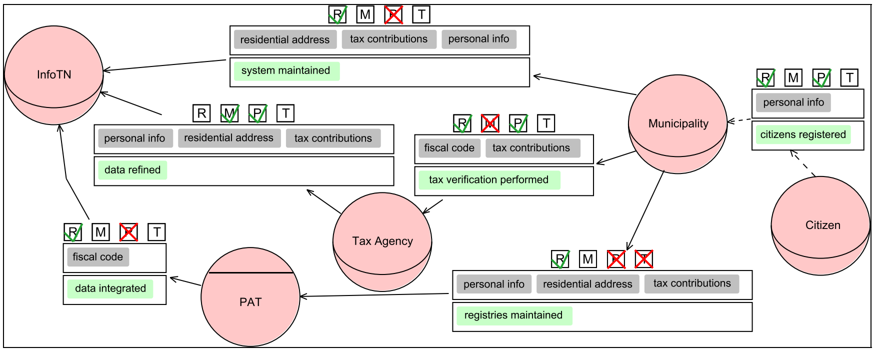


Fig. 3. Partial STS-ml authorisation view of the tax collection scenario.

- *Scope of authorisation*: authority over information can be limited to the scope of a certain goal. Our notion of goal scope adopts the definition in [22], which includes the goal tree rooted by that goal. As a result, if a goal is specified in the scope of authority, authority is given to make use of the information not only for the specified goal, but also for all its sub-goals. We assume that goal decompositions are part of the domain knowledge: there is no dispute between actors about how goals are hierarchically structured.
- *Transferability of permissions*: it specifies whether the authorisee actor is in turn entitled to transfer the received permissions or specify prohibitions (on the received permissions) to other actors.

Graphically, an authorisation is a box that contains three slots to support the first three dimensions (see Table 4), while the forth dimension, transferability, is graphically represented via the authorisation line: transferability of the authorisation is allowed when the authorisation arrow line is solid, and is not granted when the arrow line is dashed. In the TasLab case study, Fig. 3, `Municipality` authorises `InfoTN` to read information `personal info`, `residential address`, and `tax contributions`, but it prohibits any production of such information, in the scope of goal `system maintained`, while granting a transferable authorisation.

Note that the information owner has all permissions over her information, while prohibitions over this information do not apply to such actor. She is the legitimate actor for passing authorisations to others in the socio-technical system.

For the other actors, authorisations are summed up, that is, whenever different authorisation relationships are drawn towards a given actor, it is enough that at least one authorisation grants a given right (e.g., to read) and none prohibits this right, then the authorised actor has the right to read the specified information. For instance, `InfoTN` has the right to read information `personal info`, `residential address`s, and `tax contributions` for goal `data refined`, since there are two authorisations towards this actor, one from `Municipality` and the other one from the `Tax Agency`, the first grants the right to read, while the other does not specify anything on this operation. The same applies for the modification operation as well. However, `InfoTN` does not have the right to transmit the said information entities, since no authoriser actor (neither `Municipality` nor `Tax Agency`) granted the permission to transmit (`T`).

3.2. Security requirements in STS-ml

STS-ml aims to comply with the terminology used by security experts. As such, we align our work with the principles adopted by the (information) security community. Although there is no agreed upon taxonomy of security principles, our proposal relies on six principles that mainstream taxonomies and security standards do acknowledge [23–26]: *confidentiality*, *integrity*, *availability*, *authenticity*, *reliability*, and *accountability*. To such extent, we have devised a taxonomy—see Fig. 4—that represents the supported security requirements. This taxonomy refines the core security principles in information and computer security, and serves as a checklist for requirements analysts and security engineers to keep track of which security needs have been considered or are still outstanding. While some requirements apply to all domains (e.g., confidentiality), others are domain-specific (e.g., reliability, applicable to systems offering real-time services for instance). The supported requirements have been refined through collaboration with industry,[4] and confirmed by evaluation workshops [17].

As STS-ml relates security to the interactions among actors, security requirements are specified by allowing actors to express their concerns with regard to security (security needs) over the interactions they participate, namely *goal delegations*, *document transmissions*, role adoption (*play*), and *authorisations*. Therefore, security requirements in STS-ml are specified through the *social* and *authorisation* models, while the *information view* serves as a bridge between the two to support expressing a richer set of information security requirements.

Note that in STS-ml there is a one-to-one mapping of security needs and security requirements, that is, each security need expressed by one actor to another results in a security requirement from the second actor to the first for the satisfaction of the security need. Security needs are expressed through the various security requirements types, and thus, determine the

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

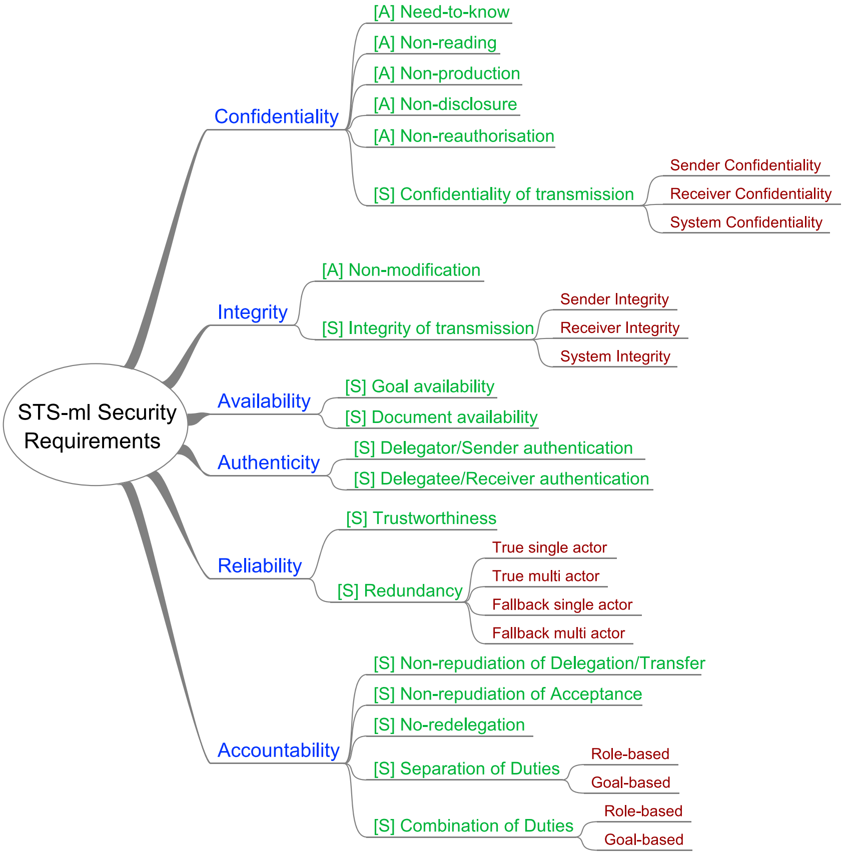


Fig. 4. Security requirements types in STS-ml, classified according to information security principles, and indicating the view that enables expressing them (A = Authorisation, S = Social).

type of security requirement to be satisfied by the responsible actor. For simplicity we explain directly the security requirements types specified with STS-ml.

An important advantage of having the classification of security requirements types is that it facilitates (aided by automated analysis techniques) answering questions such as: "*Are there any violations of citizens' confidentiality requirements?*", "*Is the integrity of tax payers' records preserved?*", etc.

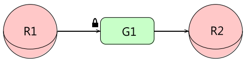
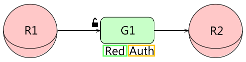
3.2.1. Specifying security requirements in the social view

We present how the security requirements listed in Fig. 4 are expressed over social relationships in the social view, doing this per social relationship.

3.2.1.1. Over goal delegations. Security requirements are specified over goal delegations by selecting the applicable security needs (from now on *security requirements types*). To do so, delegations are annotated via security requirements types that the interacting actors want each other to comply with (see Table 5 for details).

The following are the security requirement types over goal delegations:

Table 5
Delegations and the representation of security requirements types.

Graphical notation	Description
	A delegation over which security needs are specified, represented by the closed padlock sign
	Explicit visualisation of the security needs specified over the goal delegation, represented by an open padlock sign and label below the delegation; in this case redundancy and authentication are specified

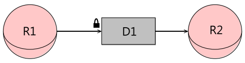
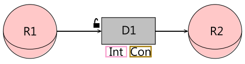
- *Non-repudiation of delegation/acceptance* [R_1, R_2]: the delegator (the delegate) shall ensure non-repudiation of the delegation (non-repudiation of the acceptance of delegation). In Fig. 1, Tax Agency wants InfoTN not to repudiate the acceptance of delegation for data completeness ensured.
- *Redundancy*: the delegator shall adopt redundant strategies for the achievement of a delegated goal, either by: (a) using alternative internal capabilities; or (b) relying on other actors. We consider two types of redundancy: (1) *True redundancy*: two or more different strategies are executed simultaneously; or (2) *Fallback redundancy*: a primary strategy is selected to fulfil the goal, while other strategies are maintained as backup, and are used only if the primary strategy fails. By intertwining (a–b) with (1–2), STS-ml supports four mutually exclusive redundancy security requirements types: (a1) true redundancy single [R_3], (a2) fallback redundancy single [R_4], (b1) true redundancy multi [R_5], and (b2) fallback redundancy multi [R_6]. For instance, in the TasLab case study InfoTN requires TN Company Selector to ensure true redundancy single for goal data refined.
- *No-redelegation* [R_7]: the delegatee actor shall not further delegate goal fulfilment. In Fig. 1, for instance, InfoTN requires TN Company Selector not to redelegate goal search module built.
- *Trustworthiness* [R_8]: the delegatee actor shall be trustworthy for the delegation to take place. The delegatee shall provide a proof of trustworthiness, say issued by a certification authority. In Fig. 1, InfoTN imposes such a requirement when delegating tax contributions obtained to Municipality.
- *Goal availability* [R_9]: the delegatee shall guarantee a minimum availability level for the delegated goal. For instance, InfoTN requires the Municipality to guarantee an availability level of 95% for goal tax contributions obtained.
- *Delegator/delegatee authentication* [R_{10}, R_{11}]: the delegator (delegatee) shall authenticate herself for the delegation to take place. The delegation of goal data refined from InfoTN to TN Company Selector includes a delegator authentication requirement. The delegation of goal corporate data verified from InfoTN to PAT, on the other hand, includes a delegatee authentication requirement: InfoTN wants to ensure that the verification is performed by the same person that maintains the data record.

3.2.1.2. Over document transmissions. Security requirements types over document transmissions are specified in a similar way to those over goal delegations. That is, document transmissions are annotated via security requirements types that the interacting actors want each other to comply with (see Table 6 for details).

The following are the security requirement types to constrain how documents are transmitted among actors:

- *Non-repudiation of transfer/acceptance* [R_{12}, R_{13}]: the sender (receiver) shall not repudiate that the document transfer (acceptance of the transmission) has taken place. In Fig. 1, Okkam Srl requires Tax Agency not to repudiate the acceptance of transmission of document tax payers knowledge base.
- *Integrity of transmission*: requires that the integrity of a document is preserved while being transmitted. Integrity of transmission might be required by the sender, the receiver, as well as by the system (meaning the socio-technical system), thus the requirement is specialized into: (1) *Sender Integrity* [R_{14}]: the sender shall ensure the integrity of transmission for the given document is preserved. For instance, in the TasLab case study Tax Agency requires InfoTN to guarantee the transmission integrity of high quality data; (2) *Receiver Integrity* [R_{15}]: the receiver shall ensure the integrity of transmission for the given document *D* is preserved while being transmitted; (3) *System Integrity* [R_{16}]: the system shall ensure that the integrity of transmission of a document in transmission is preserved. This indicates that the requirement is imposed by the organisation itself deeming the document as important to preserve integrity during transmission.
- *Document availability* [R_{17}]: the sender shall guarantee an availability level expressed in percentage for the transmitted document. For instance, InfoTN requires TN Company Selector to ensure an availability level of 70% for the document high quality data, see Fig. 1.
- *Sender/receiver authentication* [R_{18}, R_{19}]: the sender (receiver) shall authenticate herself for the document transmission to take place. In the TasLab scenario in Fig. 1, InfoTN expresses the requirement that the transmission of document high quality data necessitates the receiver's authentication, while Okkam Srl expresses the requirement that the transmission of document tax payers knowledge base necessitates sender's authentication.
- *Confidentiality of transmission*: the confidentiality of some information shall be preserved while it is transmitted from one actor to another. Confidentiality of transmission might be required by the sender, the receiver, as well as by the system (meaning the socio-technical system), thus the requirement is specialized into: (1) *Sender Confidentiality* [R_{20}]: the sender shall ensure that the confidentiality of transmission for the given document is preserved; (2) *Receiver Confidentiality* [R_{21}]: the receiver shall ensure the confidentiality of transmission for the given document is preserved; (3) *System Confidentiality* [R_{22}]: the system shall ensure that the confidentiality of transmission of a document in transit is preserved. This indicates that the requirement is imposed

Table 6
Document transmissions and the representation of security requirements types.

Graphical notation	Description
	A document transmission over which security needs are specified, closed padlock sign
	Explicit visualisation of security needs specified over the document transmission, represented by an open padlock sign and labels for security needs below the transmission. In this case integrity and confidentiality are specified.

by the organisation itself deeming the document as important to preserve confidentiality while being transmitted. For instance, `Municipality` shall ensure the sender confidentiality of document `tax contributions file` when transmitting it to `InfoTN`.

3.2.1.3. Over responsibility uptake. This type of security requirements constrains the uptake of responsibilities, i.e., the adoption of roles and the pursuit of goals.

- *Separation of duties* (SoD): (1) *Role-based SoD* [R_{23}]: defines that two roles are incompatible, i.e., all agent shall obey not adopting (playing) both roles simultaneously. (2) *Goal-based SoD* [R_{24}]: defines incompatible goals, i.e., all agents shall obey not pursuing both goals G_1 and G_2 should they pursue one of them. The requirement is graphically shown as a line (no arrows due to symmetry) annotated with a circle and the ≠ symbol. An example is that between goals `corporate records created` and `citizens' records created` of `Tax Agency`, see Fig. 1.
- *Combination of duties* (CoD): (1) *Role-based CoD* [R_{25}]: defines a binding (combination) between roles, i.e., all agents shall obey adopting both roles simultaneously, if they adopt one of them. (2) *Goal-based CoD* [R_{26}]: all agents shall obey pursuing both goals if they pursue one of them. The requirement is graphically shown as a line (no arrows due to symmetry) annotated with a circle and the = symbol. An example of this requirement is expressed between goals `semantic search built` and `enterprise search built` of `TN Company Selector`, see Fig. 1.

3.2.2. Specifying security requirements over authorisations

Let *Auth* stand for `authorise` $(A_1, A_2, \mathcal{I}, \mathcal{G}, \mathcal{OP}, TrAuth)$, 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 allowed and prohibited operations $\{\mathtt{R}, \mathtt{M}, \mathtt{P}, \mathtt{T}\} \cup \{\overline{\mathtt{R}}, \overline{\mathtt{M}}, \overline{\mathtt{P}}, \overline{\mathtt{T}}\}$, and *TrAuth* is a boolean value determining transferability:

- $\mathcal{G} \neq \emptyset \rightarrow$ *Need-to-know* [R_{27}]: the authorisee A_2 shall not 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 read only for goal `data refined`.
- $\overline{\mathtt{R}} \in \mathcal{OP} \rightarrow$ *Non-reading* [R_{28}]: the authorisee shall not read documents representing information in $\mathcal{I}$. There is no example of non-reading in the TasLab case study, for authority to read has not been prohibited to any actor.
- $\overline{\mathtt{M}} \in \mathcal{OP} \rightarrow$ *Non-modification* [R_{29}]: the authorisee A_2 shall not modify documents that include information in $\mathcal{I}$. `Municipality` requires `InfoTN` not to modify personal info, `residential address` and `tax contributions`.
- $\overline{\mathtt{P}} \in \mathcal{OP} \rightarrow$ *Non-production* [R_{30}]: the authorisee A_2 shall not produce any documents that include information in $\mathcal{I}$. For example, the PAT expresses a non-production requirement on `fiscal codes` to `InfoTN`, by prohibiting the production operation, see Fig. 3.
- $\overline{\mathtt{T}} \in \mathcal{OP} \rightarrow$ *Non-disclosure* [R_{31}]: the authorisee A_2 shall not transmit (disclose) to other actors any document that includes information in $\mathcal{I}$. For instance, `Municipality` requires this in the authorisation over information `personal info`, `residential address` and `tax contributions` to PAT.
- *Non-reauthorisation* [R_{32}]: the authorisee A_2 shall not redistribute the permissions to other actors. If the authorisee receives an authorisation that contains only prohibitions, then not-reauthorisation does not apply. A_2 is subject to this requirement in two cases:
  - *explicitly*, when the authorisee receives an authorisation that is non-transferable, i.e., $TrAuth = \mathtt{false}$. Graphically, the authorisation relationship is represented by a dashed arrow line. An example is the authorisation from `Citizen` to `Municipality`, see Fig. 3.
  - *implicitly*, when no actor specifies permissions or prohibitions for performing a certain operation on a given information. For instance, `Tax Agency` has no incoming authorisation for information `personal info`.

4. Automated reasoning support

Modelling languages are useful means to represent knowledge through models, but as the latter grow in size, they might become inconsistent. STS-ml models are no exception to the rule. Automated reasoning techniques come to help in identifying potential inconsistencies.

In this section, we present the formal framework of STS-ml that defines the semantics of the language unambiguously (Section 4.1); then, we propose a number of different analysis techniques over STS-ml models (Section 4.2).

4.1. Formal framework

We introduce the formal framework for STS-ml models, and illustrate it on the motivating scenario modelled in Section 3. We employ the following notation: atomic variables are strings in italic with a leading capital letter (e.g., G, I); sets are strings in the calligraphic font for mathematical expressions (e.g., $\mathcal{G}, \mathcal{I}$); relation names are in sans-serif with a leading non-capital letter (e.g., wants, possesses); constants are in typewriter style with a leading non-capital letter (e.g., `and`, `or`).

The elements of an STS-ml model (e.g., goal, goal delegation) are represented by the predicates in Tables 1–4 (e.g., `goal`, `delegates`), including the notions of intentional relationship ($\mathcal{IRL}$) and social relationship ($\mathcal{SR}$). For simplicity of reading, we do not redefine them here.

The notion of an *actor model* formalises the contents of the oval balloon that is associated with a role or an agent shape in the social view. The goals and documents of a specific actor in a socio-technical system relate one to another. There are important relationships that need to be captured, and that enable defining the rationale of an actor, i.e., how it aims to attain its goals. This is the purpose of defining *actor model*, for it consists of an actor's intended goals, its possessed documents, and the relationships between these elements, aka its intentional relationships. For instance, the actor model of InfoTN in Fig. 1 includes: InfoTN wants to fulfil goal online system built, for which it has to fulfil goals search module built, navigation module built, and system maintained. The fulfilment of the latter goal requires the fulfilment of goal data completeness ensured and data files stored. The fulfilment of data completeness ensured requires reading the document high quality data, and so on. Definition 1 ensures that all the relationships in an actor model are among elements of the same actor.

Definition 1. An actor model AM is a tuple $\langle A, \mathcal{G}, \mathcal{D}, \mathcal{IRL}, T\rangle$ where A is an actor, $\mathcal{G}$ is a set of goals, $\mathcal{D}$ is a set of documents, $\mathcal{IRL}$ is a set of intentional relationships (Table 1) over goals in $\mathcal{G}$ and documents in $\mathcal{D}$, and T is an actor type (role or agent).

Given an intentional relationship IRL in $\mathcal{IRL}$:

- if IRL $=$ decomposes$(A', G, \mathcal{S}, DecT)$, then $A' = A$, and both G and all goals in $\mathcal{S}$ are in $\mathcal{G}$;
- if IRL $=$ reads/modifies/produces(A', G, D), then $A' = A$, G is in $\mathcal{G}$, and D is in $\mathcal{D}$. □

Definition 1 states that an actor model is defined by the role or agent (A, defined by the type T), the goals it wants to achieve $\mathcal{G}$, the documents $\mathcal{D}$, together with goal decompositions, and the goal-document relationships determining whether the actor possesses the document or reads/modifies/produces it (in $\mathcal{IRL}$). Given the intentional relationships in an actor model, Definition 1 limits their creation only within an actor's scope: (i) given that all goals are in the set of goals of the actor, decompositions, also, are among these goals; and (ii) given goal-document relationships (reads/modifies/produces), their goals are in the set of the goals of the actor ($\mathcal{G}$) and documents are in the set of documents of the actor ($\mathcal{D}$).

The social view for the TasLab case study in Fig. 1 includes multiple actor models, one per actor. An excerpt of an actor model is as follows: $A =$ InfoTN, $\mathcal{G}$ includes online system built, data refined, etc., $\mathcal{IRL}$ includes decomposes(InfoTN, data completeness ensured, {data refined, data integrated}, and) and modifies(InfoTN, data refined, tax contribution file), and $T =$ agent.

We denote the set of actor models as $\mathcal{AM}$. We tie together all the elements in the social, information, and authorisation views to define an STS-ml model.

Definition 2. An STS-ml model M is a tuple $\langle \mathcal{AM}, \mathcal{SR}, IM, \mathcal{SRQ}\rangle$ where $\mathcal{AM}$ is a set of actor models, $\mathcal{SR}$ is a set of social relationships, *IM* is an information view model, and $\mathcal{SRQ}$ is a set of security requirements of the categories R_1–R_{32}. □

An STS-ml model is composed of all the actor models ($\mathcal{AM}$) of the identified actors, the social relationships among them ($\mathcal{SR}$), the information view model (*IM*) tying together actors' information and documents, and the security requirements expressed by actors ($\mathcal{SRQ}$) over social relationships. The STS-ml model for the TasLab case study is composed of all the actor models of the actors drawn in the social view (Fig. 1), the social relationships drawn in the social as well as the authorisation view (Fig. 3), the information view (Fig. 2), and the set of security requirements R_1–R_{32}.

When tying together the different elements in an STS-ml model, we need to ensure its well-formedness. Definition 3 lays down the constraints on the well-formedness of an STS-ml model.

Definition 3. An STS-ml model $M = \langle \mathcal{AM}, \mathcal{SR}, IM, \mathcal{SRQ}\rangle$ is well-formed if:

1. social relationships are only over actors with models in $\mathcal{AM}$, and;
2. only leaf goals are delegated; for each delegates (A, A', G) in $\mathcal{SR}$, there is an actor model $\langle A, \mathcal{G}, \mathcal{D}, \mathcal{IRL}, T\rangle$ in $\mathcal{AM}$ such that $G \in \mathcal{G}$ and there is no decomposition of G in $\mathcal{IRL}$;
3. delegated goals appear in the delegatee's actor model: for each delegates (A', A, G) in $\mathcal{SR}$, there is an actor model $\langle A, \mathcal{G}, \mathcal{D}, \mathcal{IRL}, T\rangle$ in $\mathcal{AM}$ s.t. $G \in \mathcal{G}$;
4. the transmitter must possess the document for the document transmission to take place: for each transmits (A, A', D) in $\mathcal{SR}$, there is an actor model $\langle A, \mathcal{G}, \mathcal{D}, \mathcal{IRL}, T\rangle \in \mathcal{AM}$ s.t. $D \in \mathcal{D}$. An actor possesses a document if:
   - it has the document since scratch: possesses$(A, D) \in \mathcal{IRL}$, or
   - it creates the document, i.e., $\exists\ G \in \mathcal{G}$. Wants$(A, G) \wedge$ produces$(A, G, D) \in \mathcal{IRL}$, or
   - there is an actor A'' such that transmits$(A'', A, D) \in \mathcal{SR}$.
5. transmitted documents appear in the receiver's actor model: for each transmits(A′, A, D) in $\mathcal{SR}$, there is an actor model $\langle A, \mathcal{G}, \mathcal{D}, \mathcal{IRL}, T\rangle$ in $\mathcal{AM}$ such that $D \in \mathcal{D}$;
6. authorisations are syntactically well-formed: for each authorises$(A, A', \mathcal{I}, \mathcal{G}, \mathcal{OP}, TrAuth) \in \mathcal{SR}$, there must be at least one information entity specified ($|\mathcal{I}| \geq 1$), and at least one prohibition or permission is specified;
7. all security requirements in $\mathcal{SRQ}$ are over social relationships in $\mathcal{SR}$, actors with models in $\mathcal{AM}$, or over their goals;
8. the model M complies with the following constraints:
   (a) delegations have no cycles: for each delegates(A, A', G) in $\mathcal{SR}$, there is no A″ such that delegates(A'', A, G) in $\mathcal{SR}$ or delegates(A'', A, G_i) in $\mathcal{SR}$, where G_i is a descendant of G in the goal tree of A'';
   (b) part-of relationships (either over information or documents) have no cycles. □

Following Definition 3 in the social view in Fig. 1 we cannot have, for instance, a delegation of goal search module built of TN Company Selector, rather its subgoals semantic search built and enterprise search built are delegated (complying with

2); TN Company Selector's delegated goal semantic search built appears in Okkam Srl's actor model (3); Okkam Srl would not be able to transmit document tax payers knowledge base to the Tax Agency if it did not produce this document when pursuing goal payers' knowledge base created (4); Okkam Srl has document high quality data since it was transmitted from TN Company Selector (5); in Fig. 3 the authorisation from Tax Agency to InfoTN would be not well-formed, if the rights to read and produce would not be specified (6).

The imposed constraints (8) are necessary as they determine exceptions that would not allow us to reason over STS-ml models. For instance, delegation cycles would result in an STS-ml model in which it is not clear which actor is responsible for fulfilling the delegated goal. For instance, in Fig. 1 there cannot be a delegation from Okkam Srl back to TN Company Selector of goals semantic search offered, payers' knowledge base created, or data interconnected (8a). part-of cycles would result in ambiguities over information ownership (when specified over information) or ambiguities in reasoning over violation of security requirements (more specifically confidentiality requirements, such as non-reading, non-disclosure, etc.) when over documents. For instance, in Fig. 2 there cannot be a part-of relationship from information land details to information land ownership (8b). As we will see in Section 5, the well-formedness of an STS-ml model can be verified using STS-Tool, which integrates these checks.

We define *authorisation closure* in order to determine actors' final rights and the security requirements they have to comply with. Authorisation closure formalises the intuition that, if an actor A_2 has no incoming authorisation for an information I, then A_2 has a prohibition for I. Such prohibition is an authorisation from the information owner that prohibits all operations as well as the right to transfer authorisations. This is formally specified in Definition 4.

Definition 4. Let $M = \langle \mathcal{AM}, \mathcal{SR}, IM, \mathcal{SRQ} \rangle$ be a well-formed STS-ml model. The authorisation closure of $\mathcal{SR}$, denoted as $\Delta_{\mathcal{SR}}$, is a superset of $\mathcal{SR}$ that makes prohibitions explicit, when no authorisation is granted by any actor. Formally, $\forall A, A'$ with an actor model in $\mathcal{AM}$, $\forall \text{owns}(A, I) \in \mathcal{SR}. \nexists A''.\mathtt{authorises}(A'', A', \mathcal{I}, \mathcal{G}, \mathcal{OP}, \mathtt{TrAuth}) \in \mathcal{SR} \wedge I \in \mathcal{I} \rightarrow \mathtt{authorises}(A, A'', I, \mathcal{G}', \overline{\mathcal{OP}}, \mathtt{false}) \in \Delta_{\mathcal{SR}}$ where $\mathcal{G}'$ is the set of goals of A'. □

As described in Section 3.1.3, security requirements are generated over explicit prohibitions. But, since authorisations are summed up, it is not always clear from the models, what operations actors are granted over a given information. Whenever there are no incoming authorisation (no single authorisation relation has been drawn towards the actor), or no incoming authorisations grant a given operation from $\mathcal{OP}$ over I ($\nexists A''.\mathtt{authorises}(A'', A', \mathcal{I}, \mathcal{G}, \mathcal{OP}, \text{TrAuth}) \in \mathcal{SR}$), we generate an authorisation from the information owner that prohibits all operations, for which no permissions were granted, and sets transferability to false ($\mathtt{authorises}(A, A'', I, \mathcal{G}', \overline{\mathcal{OP}}, \mathtt{false}) \in \Delta_{\mathcal{SR}}$, where A is the information owner). We do this to reason over possible unauthorised operations over information. For instance, in Fig. 3, the lack of incoming authorisations towards the Tax Agency for information land details, implies authorises(PAT, Tax Agency, land details, $\mathcal{G}$, $\overline{\mathcal{OP}}$, false) $\in \Delta_{\mathcal{SR}}$, where $\mathcal{G}$ is the set of goals of the Tax Agency, and PAT is the owner of information land details, see Fig. 2.

Notice the key difference between the explicit prohibitions and the implicit prohibitions derived from authorisation closure. In the former case, an actor wants to ensure that another cannot do specific operations on some information. In the latter case, the lack of permissions can be a temporary situation, which could be changed by any actor having permission and authority to transfer such permission, and using this authority by specifying an authorisation that grants the said permission. We apply Definition 4 when executing the automated reasoning. This ensures that changes over the model are taken into account, and a recalculation of the authorisation closure takes place.

4.2. STS automated reasoning

The formal framework in Section 4.1 allows us to perform automated analysis to verify if: (i) the STS-ml model is well-formed, and (ii) any security requirements are violated. We focus here on the second analysis, known as *security analysis*, given that the first is covered by Definition 3.

Security analysis verifies possible violations at two levels: (i) identifying possible conflicts among security requirements, i.e., two or more requirements cannot be all fulfilled by the same system, and (ii) identifying conflicts between actors' business policies and the security requirements imposed on them.

4.2.1. Security requirements conflicts: 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 both permissions and prohibition. For instance, in Fig. 3, the authorisation from Municipality to InfoTN allows reading information personal info, residential address and tax contributions, it prohibits the production of the given information, and specifies nothing on modification and transmission. The authorisation is limited to the scope of goal system maintained. If we had another authorisation towards InfoTN for the same information which were not restricted to a goal scope, then we would have an authorisation conflict. Similarly, if we had more authorisations towards InfoTN prohibiting the granted operation (in this case R) or granting the right to perform any of the prohibited operations (in this case P) over any of the specified information entities (personal info, residential address or tax contributions) or parts of these information entities, in the scope of the same goal (i.e. including its subgoals), then we would have an authorisation conflict. This is formalised by Definition 5.

Definition 5. Two authorisations $\mathtt{authorises}(A_1, A_2, \mathcal{I}_1, \mathcal{G}_1, \mathcal{OP}_1, \mathit{TrAuth}_1)$, and $\mathtt{authorises}(A_3, A_2, \mathcal{I}_2, \mathcal{G}_2, \mathcal{OP}_2, \mathtt{TrAuth}_2)$ are conflicting (a-conflict(Auth$_1$, Auth$_2$)) if and only if they both regulate the same information ($\mathcal{I}_1 \cap \mathcal{I}_2 \neq \emptyset$), 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) $\exists \texttt{op}.\ \texttt{op} \in \mathcal{OP}_1 \wedge \overline{\texttt{op}} \in \mathcal{OP}_2$, or vice versa (where op is one of {R, M, P, T}); or (ii) $\texttt{TrAuth}_1 \neq \texttt{TrAuth}_2$. □

Definition 5 states that an authorisation conflict occurs if both authorisations apply to the same information, and either (1) one authorisation restricts the permission to a goal scope ($\mathcal{G}_1 \neq \emptyset$), while the other does not ($\mathcal{G}_2 = \emptyset$), that is, one implies a need-to-know requirement, while the other grants rights for any purpose; or, (2) the scopes are intersecting ($\mathcal{G}_1 \cap \mathcal{G}_2 \neq \emptyset$), and contradictory permissions are granted (on operations op, or authority to transfer TrAuth).

In Fig. 3, there are two authorisations towards InfoTN on personal info, residential address and tax contributions: that from Municipality, which grants R, but prohibits P, and says nothing on M and T; that from Tax Agency, which grants M and P, and says nothing on 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, following Definition 5, they are conflicting with respect to the production of the specified information.

An STS-ml model is *authorisation-consistent* when no authorisation conflicts exist.

4.2.2. Conflicts between business policies and security requirements

Each actor model defines a specific actor's business policy, i.e., its goals and the alternative strategies to achieve these goals. Given an authorisation-consistent STS-ml model, we verify if any security requirement is in conflict with the business policy of any actor in the model. For instance, a requirement such as non-modification(Municipality, InfoTN, {personal info}) conflicts with InfoTN's business policy that includes the relationship modifies(InfoTn, goal, personal info).

An actor's *business policy* defines alternative strategies for an actor to achieve its 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-decomposes, reads, modifies, and produces) among those goals and documents, as well as goal delegations and document transmissions that start from that actor. For instance, the business policy of TN Company Selector includes goals data refined, navigation module built, for which document high quality data needs to be read, and search module built that the actor or-decomposes into semantic search built and enterprise search built (see Fig. 1). The or-subgoals denote alternative strategies, i.e., the actor can choose either of them.

Definition 6. Given a business policy for an actor A (denoted as 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-decomposes, reads/produces/modifies, transmits, and delegates relationships);
- for every root goal G that is delegated to A, G can optionally be pruned. □

Definition 6 formalises the intuition that alternative strategies are introduced by: (i) choosing one subgoal in an or-decomposition; and (ii) deciding whether to pursue root goals delegated by other actors. In Fig. 1, the business policy for TN Company Selector includes only delegated root goals. One strategy involves pruning all root goals but search module built. This goal is or-decomposed; by Definition 6, one subgoal is retained in the strategy (e.g., semantic search built), while other goals are pruned (e.g., enterprise search built). The reads 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 the notion of a *variant* to: (i) consistently combine actors' strategies (each actor fulfils the root goals in its strategy), by requiring that delegated goals are in the delegatee's strategy; and to (ii) include the authorises relationships in the STS-ml model. This enables us to identify conflicts that occur only when the actors choose certain strategies, see Definition 7.

Definition 7. 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 for M (denoted as $\mathcal{V}_M$) consists of:

- a set of strategies $\{S_{P_{A_1}}, \ldots, S_{P_{A_n}}\}$ such that, for each A', A'', G, if delegates(A', A'', G) is in $S_{P_{A'}}$, then G is in $S_{P_{A''}}$, and
- all the authorises relationships from M. □

Variants constrain the strategies of the actors. In Fig. 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 its 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. Note that an STS-ml model may have multiple variants. For example, 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 in the actors' business policies, and at the interplay of the two we identify conflicts, see Definition 8.

Definition 8. Given a variant $\mathcal{V}_M$, a conflict between business policies and security requirements (Bus–Sec conflict) exists if there is an actor A such that:

- the strategy of A in $\mathcal{V}_M$ contains one or more relationships (as denoted by its business policy) that are prohibited by a security requirement requested (prescribed) by another actor A' to A;
- the strategy of A in $\mathcal{V}_M$ does not contain any relationships required by some requirement requested by another actor A' to A. □

Table 7
Security requirements and their design-time verification against a variant $\mathcal{V}_{\mathrm{M}}$.

Requirement	Verification at design-time
Accountability requirements	
non-redelegation (A_1, A_2, G) [R_7]	$\nexists$delegates$(A_2, A_3, G') \in \mathcal{V}_M$. $G' =$ G or G' is a subgoal of G
role-sod (STS, Ag, R_1, R_2) [R_{23}]	{plays(Ag, R_1), plays(Ag, R_2)} $\nsubseteq \mathcal{V}_M$
goal-sod (STS, Ag, G_1, G_2) [R_{24}]	Ag should not pursue both G_1 and G_2 or their subgoals
role-cod (STS, Ag, R_1, R_2) [R_{25}]	{plays(Ag, R_1), plays(Ag, R_2)} $\subseteq \mathcal{V}_M$
goal-cod (STS, Ag, G_1, G_2) [R_{26}]	Ag should pursue both G_1 and G_2 or their subgoals
Reliability requirements	
true-single-red (A_1, A_2, G) [R_3]	Partial. A_2 pursues goals in $\mathcal{V}_M$ that define at least two disjoint ways to support G
fback-single-red (A_1, A_2, G) [R_4]	
true-multi-red (A_1, A_2, G) [R_5]	Partial. Both A_2 and another actor A_3 support G, each in a different way
fback-multi-red (A_1, A_2, G) [R_6]	
Integrity requirements	
Non-modication $(A_1, A_2, \mathcal{I})$ [R_{29}]	$\nexists$modifies$(A_2, G, D) \in \mathcal{V}_M$. D makes tangible (part of) $I \in \mathcal{I}$
Confidentiality requirements	
need-to-know $(A_1, A_2, \mathcal{I}, \mathcal{G})$ [R_{27}]	$\nexists$reads/modifies/produces$(A_2, G, D) \in \mathcal{V}_M$. D makes tangible (part of) $I \in \mathcal{I}$ and $G \notin \mathcal{G}$
non-reading $(A_1, A_2, \mathcal{I})$ [R_{28}]	$\nexists$reads$(A_2, G, D) \in \mathcal{V}_M$. D makes tangible (part of) $I \in \mathcal{I}$
non-production $(A_1, A_2, \mathcal{I})$ [R_{30}]	$\nexists$produces$(A_2, G, D) \in \mathcal{V}_M$. D makes tangible (part of) $I \in \mathcal{I}$
non-disclosure $(A_1, A_2, \mathcal{I})$ [R_{31}]	$\nexists$transmits$(A_2, A_3, D) \in \mathcal{V}_M$. D makes tangible (part of) $I \in \mathcal{I}$
not-reauthorised $(A_1, A_2, \mathcal{I}, \mathcal{G}, \mathcal{OP})$ [R_{32}]	$\nexists$authorises$(A_2, A_3, \mathcal{I}, \mathcal{G}\prime, \mathcal{OP}\prime) \in \mathcal{V}_M$. $\mathcal{G}' \subseteq \mathcal{G} \wedge \mathcal{OP}' \subseteq \mathcal{OP}$

Table 7 describes semi-formally how these conflicts are verified at design time for the different types of security requirements that STS-ml supports. Below, we provide some more details. Notice that not all security requirements are enlisted in this table, this is done on purpose, leaving out all those requirements that require runtime monitoring and verification. We discuss below the security requirements for each category (see Fig. 4 in Section 3.2 for the complete list) and give insights on the verification of runtime requirements.

4.2.2.1. Accountability requirements. Non-repudiation requirements (R_1, R_2, and R_{12}, R_{13}) expressed over goal delegations and document transmissions respectively, cannot be verified at design time for they require actions such as proof of fulfilment for the goal in case of non-repudiation of acceptance.

Not-redelegation (R_7) is verified if there is no delegation (of the delegated goal or its subgoals) relationship from the delegatee to other actors in the variant.

Role-based separation and combination of duties (R_{23} and R_{25} respectively) require all agents *not to* or *to* play two roles through `play` relationships, respectively. Goal-based separation of duties (R_{24}) is verified if no agent pursues both goals among which `goal-sod` is specified, or their subgoals. Finally, goal-based combination of duties (R_{26}) is verified in a similar way, but the agent should be the final performer (i.e., does not delegate to other actors) of both goals.

4.2.2.2. Reliability requirements. Redundancy requirements (R_3 to R_6) can be partially checked at design time. The existence of redundant alternatives can be verified, but a variant does not tell how alternatives are interleaved, i.e., if they provide true or fallback redundancy. Thus, these two types are checked the same way. Single-agent redundancy (R_3 and R_4) is fulfilled if the delegatee has at least two disjoint alternatives (via or-decompositions) for the delegated goal. Multi-actor redundancy (R_5 and R_6) requires that at least one alternative involves a third actor.

Trustworthiness requirements (R_8) cannot be verified at design time, as they require the delegatee to provide proof of trustworthiness, e.g., certification.

4.2.2.3. Authenticity requirements. Delegator/sender authentication (R_{10}, R_{18}) is typically implemented in e-commerce, wherein a certification authority guarantees the authenticity of the sellers' website. Therefore, the verification of such requirement involves actions and mechanisms that can be verified only at runtime.

Delegatee/receiver authentication (R_{11}, R_{19}), on the other hand, is one we encounter everyday when browsing the web and using our credentials (username/password) to access web information such as our email. The fulfilment of these types of requirements cannot be verified at design time.

4.2.2.4. Availability requirements. Verifying availability requirements (both goal and document availability—R_9, R_{17}) calls for a way to measure the availability level. Notice that goal availability is highly related to the notion of service availability, where a provider specifies an uptime level for the service. In service-oriented settings, availability levels often become integral part of service-level agreements. These requirements cannot be verified at design time.

4.2.2.5. Integrity requirements. Non-modification (R_{29}) requires that the authorisee's strategy in the variant includes no modifies relationship on documents that make tangible part of the given information.

Integrity of transmission requirements ($R_{14,}$ $R_{15,}$ R_{16}) prescribe that the integrity of transmission of the document is preserved; verifiable only at runtime.

4.2.2.6. Confidentiality requirements. Confidentiality requirements expressed via authorisations prescribe the set of relationships that shall not be present in the authorisee's strategy in the variant. Need-to-know (R_{13}) is verified by the absence of `reads`, `modifies`, or `produces` relationships on documents that make tangible some information from the set of authorised information entities for some goal different from the ones specified in the scope of the authorisation (or any their descendants). Requirements R_{14} to R_{16} are verified if the authorisee's strategy in the variant includes no `reads` or `produces` relationships on documents that make tangible part of the said information. Non-disclosure (R_{17}) does a similar check, but looking at `transmits` relationships. Non-reauthorisation (R_{18}) is fulfilled if there is no `authorises` relationship from the authorisee to others on any operation over the given information in the goal scope of the authorisation.

Confidentiality of transmission (R_{18}, R_{19}, R_{20}) can only be verified at runtime.

More details on the findings of the automated reasoning techniques will be discussed in Section 5.

5. Implementation and evaluation

STS-Tool[5] [27] is the CASE tool for the STS approach. It supports STS-ml modelling through three different views (social, information, and authorisation), it ensures inter-view consistency, supports requirements document generation, as well as automated reasoning for conflict detection. Under the hood, the tool encodes STS-ml models into disjunctive Datalog programs to support automated reasoning [28]. The current version of STS-Tool is the result of an iterative development process that intertwined evolutions of the language and continuous empirical evaluations of each version [17].

Here, we report on the effectiveness of the automated reasoning techniques, i.e., their capability of identifying non-trivial conflicts (Section 5.1), and their scalability when applied to large models (Section 5.2).

5.1. Findings from the TasLab case study

5.1.1. Well-formedness analysis

The execution of automated reasoning is to be performed over well-formed models (as per Definition 3). We verify a model's well-formedness in two steps, depending on the complexity of the check: (i) online or on-the-fly, while the model is being drawn, or (ii) offline, upon user explicit request for computationally expensive checks.

The STS-ml model for the TasLab case study is well-formed.

5.1.2. Security analysis

We assess the ability of the reasoning mechanisms to discover security requirements conflicts (mainly on authorisations) and between business policies and security requirements (*Bus–Sec conflict*). We provide evidence by presenting the findings from the use of the STS-Tool in modelling and analysing the TasLab case study. Based on the models that we created with the stakeholders (Figs. 1–3), the analysis returned a number of 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` authorises reading only, and requires non-production of the same information, see Fig. 5.
- *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 during the modelling, originate from the different authorisation policies of the stakeholders. Conflict resolution activities are conducting with domain experts to ultimately reach a consistent model. 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 model).

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

- `no-redelegation`: InfoTN relies on `TN Company Selector` to refine the obtained data (delegating `data refined`). On the other hand, `Tax Agency` relies on `InfoTN` for data completeness (delegating `data completeness ensured`) and requires it not to redelegate this goal. This security requirement is in conflict with InfoTN's business policy on delegating data refined, since the latter is a subgoal of `data completeness ensured`, see Fig. 1.

[5] http://www.sts-tool.eu.

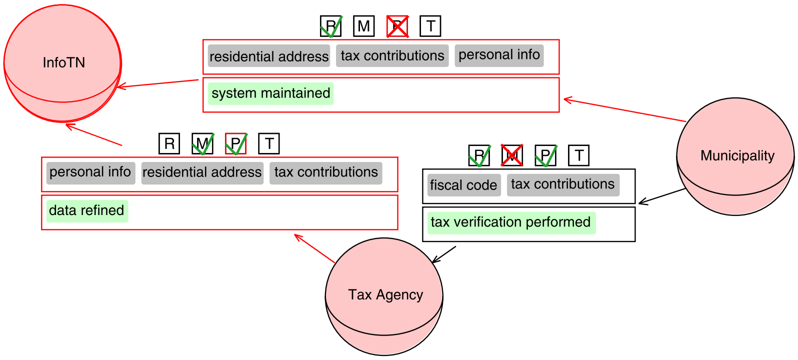


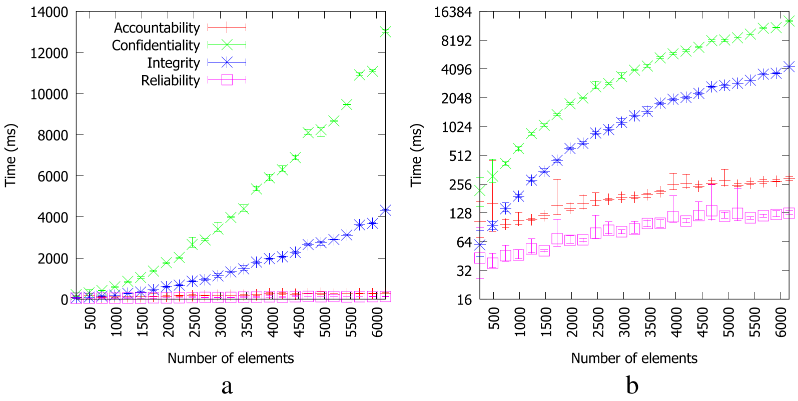
Fig. 5. Visualisation of conflicting authorisations: on authority to produce.

- true-multi-red: TN Company Selector has not employed several different strategies on its own to fulfil goal data refined, for which InfoTN has required multi actor true redundancy.
- goal-cod: goals semantic search built and enterprise search built should be pursued by the same actor, as a combination of duties is specified between them. A conflict occurs because TN Company Selector is not the final performer for both goals (semantic search built is delegated to Okkam Srl).
- non-modification: Engineering Tribute Srl makes an unauthorised modification of Citizen's personal info. There is no incoming authorisation granting Engineering Tribute Srl the right to modify, for which a non-modification requirement specified by Citizen is generated, following Definition 4.
- non-production: PAT makes an unauthorised production of tax contributions, given that this information is owned by the Municipality and the authority to produce is prohibited to PAT.
- non-reauthorisation: Citizens want only the Municipality to read and produce their personal info and do not allow the transfer of authority (explicit specification of non-reauthorisation), however the Municipality further authorises InfoTN and PAT to read this information. Another finding of the same type regards Tax Agency, which has no authority to modify information location (no authorisation granting this operation is passed to InfoTN—implicit specification of non-reauthorisation), however Tax Agency further authorises InfoTN to modify this information (by granting authority to modify over information residential address that contains information location—part-of-i(location, residential address)).

These conflicts are due to the different policies of the companies. They can be resolved through trade-offs [29] between business policies and security requirements. In STS we opt for the most secure option, especially if the security requirement derives from norms in the legal context.

5.2. Scalability study

We report a scalability study to assess how well the automated reasoning copes with large STS-ml models: how is the execution time affected by the model size?



a b

Fig. 6. Scalability analysis for *no-variability*: y-axis in linear (a) and logarithmic (b) scale.

5.2.1. Design of experiments

We take the STS-ml model of the TasLab case study as a basic building block, and clone it to obtain larger models. We increase the size of the model in two ways: (1) we augment the *number of elements* (nodes and relationships) in the model; (2) we increase the *number of variants* in the model. The latter strategy is motivated by our reasoning techniques, which rely upon the generation of STS-ml model variants (Definition 7) for identifying possible conflicts (leading to security requirements violations) in the model.

To obtain larger versions of an STS-ml model M, we perform the following steps: (i) create an identical copy (clone) M' of M; (ii) add a fictitious leaf goal G to a randomly chosen actor A in M; (iii) delegate G to the clone of A (actor A') in M'; (iv) decompose (see below for the decomposition type) the goal G of A' into (a) the root goal of its existing goal model, and (b) a fictitious goal. This process increases the number of variants, for the initial model M already contains variability. We repeat these steps to obtain larger and larger models.

We adopt three different customizations of the procedure above to independently test the effect of increasing the number of elements and that of variants:

- *no-variability*: all decompositions in M and M' are treated as AND-decomposition, thus leading to one variant;
- *medium-variability*: goal G of A' is AND-decomposed, all other decomposition are not modified;
- *high-variability*: goal G of A' is OR-decomposed, all other decomposition are not modified.

The first strategy (no-variability) enables assessing scalability with respect to the number of elements, while the other two strategies (medium- and high-variability) are meant to assess scalability with respect to the number of variants.

We conduct three sets of tests with models of increasing size generated using each of the strategies above (no-, medium-, and high-variability). For each cloned model, we run the analysis 7 times and calculate the average execution time to ensure stability of the latter.

5.2.2. Results

We have conducted experiments on a DELL Optiplex 780 machine, Pentium(R) Dual-Core CPU E5500 2.80 GHz, 4 Gb DDR3 399, powered by Windows 7. Figs. 6–8 summarise the results of our scalability experiments. Below, we detail and discuss the results for each set of tests.

No-variability. The scalability graphs are shown in Fig. 8, and are split according to the time spent to analyse requirements of the different categories in Table 7: accountability, confidentiality, integrity, and reliability. We present the y-axis both in linear scale and logarithmic scale; the latter visualisation serves to take a closer look at reliability and accountability, which take considerably less time than the other types of requirements. As noticeable by the plot, all techniques scale very well (linear or quasi-linear growth). Interestingly, the tool is able to reason about extra-large models (>6000 elements) in about 12 seconds. We conclude that the number of elements does not constitute an obstacle to the scalability of our techniques.

Medium- and high-variability. The graphs in Figs. 7 and 8 present (in $\log_2$ scale) the results of the scalability analysis with respect to the number of variants. This dimension affects the execution time of our techniques the most. The analysis of reliability requirements is not affected by the increasing number of variants: the reason for this is that our tool processes these types of requirements before increasing the number of variants. As far as the other types of requirements are concerned, the growth is still linear in the number of variants, but it is exponential in the number of elements (the model with 1,048,576 variants in Fig. 7 consists of 2500 elements). Notice that the tool deals with dozens of thousands of variants in less than a minute. In this case, we conclude that the tool is adequate to deal with large models, but needs to be improved to deal with extra-large models.

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

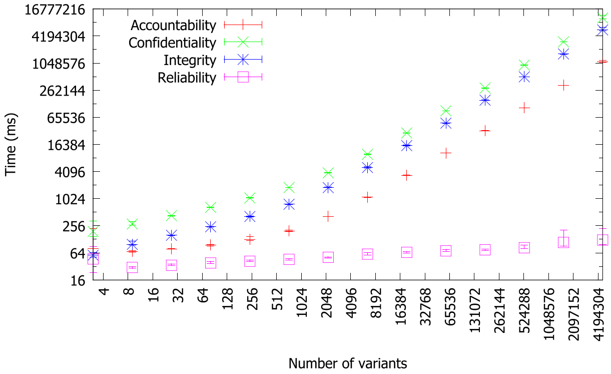


Fig. 7. Scalability analysis for *medium-variability*.

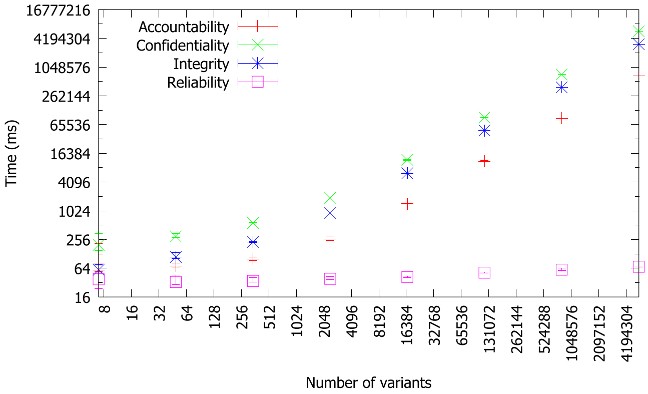


Fig. 8. Scalability analysis for *high-variability*.

6. Related work

We review related work about reasoning with requirements, reasoning on security requirements, and methods for security requirements engineering.

6.1. Reasoning with requirements: conflict identification and resolution

The importance of identifying conflicting requirements is well-known by practitioners and has been widely acknowledged by the research community [30,31].

SAT solving was applied to analyse the satisfaction or denial of goals in goal models [32]. They define a range of satisfaction and denial evidence, and introduce positive and negative contribution relationships among goals. The approach includes both qualitative and quantitative analysis techniques that determine evidence of goal satisfaction/denial by using label propagation algorithms. Conflicts are identified when both positive and negative evidences exist.

The goal model analysis procedures introduced in [32] have been included [33] into the Tropos Framework, with the objective of making the goal analysis process concrete while coping with qualitative relationships and inconsistencies among goals, in order to suggest, explore and evaluate alternative solutions. Goal analysis includes forward and backward reasoning for goal achievement.

This approach inspired further research, such as the iterative interactive identification of conflicts [34,35] for early requirements engineering. The authors propose a qualitative interactive procedure that allows users to systematically evaluate and compare the alternative actions and solutions expressed in models asking “What if?” questions. Evaluation is supplemented with domain knowledge, and human intervention is used to compensate for the incomplete nature of the models. Further work by the authors [35] presents an interactive backward procedure to answer questions “Is this possible?”, “If so, how?” and “If not, why not?”. They deal with conflicts by demanding resolution to the analyst (using human judgement) in an interactive fashion. It would be interesting to investigate how to include these interactive analysis approaches with our work.

Formal Tropos [31] is a formal language which supplements *i** concepts with first-order linear-time temporal logic. When identifying property violations, their analysis returns a concrete conflicts scenario (counterexample) that describes a property violation, to help the analyst understand the problem. Similarly, our techniques visualise and describe the identified conflicts to offer to the security requirements engineers a better understanding.

KAOS [30] includes analysis techniques to identify and resolve inconsistencies that arise from the elicitation of requirements from multiple stakeholders. Emphasis is put on identifying goal conflicts, defining various types of inconsistencies, such as *instance-level deviations*, *terminology clashes*, etc. *Conflict* is defined among several specifications of goals/requirements enhanced with domain knowledge. Most importantly, resolution techniques are discussed, such as finding alternative goal refinements, weakening goals, and using divergence resolution heuristics. While their classifications and resolution strategies are specific to KAOS concepts, future work may involve applying these to STS.

6.2. Reasoning about security requirements

SI* [13] is a security requirements engineering framework that builds on *i** [36] by adding security concepts, including delegation and trust of execution or permission. SI* uses automated reasoning to check the security properties of a model, reasoning on the interplay between execution and permission of trust and delegation relationships. Our approach clearly separates between security requirements and business policies—how actors achieve their desired objectives—, binds all requirements to interactions, and supports a larger set of security requirements.

De Landtsheer and Van Lamsweerde [37] 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.

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.

Elahi et al. [29] extend the *i** framework for security trade-off analysis (to reach a good-enough security level) in a multi-actor setting. This technique offers the possibility to assess the impact of assessing security mechanisms on actors' goals and threats. Vulnerabilities refer to the deficiencies in the structure of goals and activities of intentional agents. Differently from STS-ml, they do not take into account security issues related to actors' interaction.

6.3. Security requirements methods

These are methods to identify possible security conflicts through elicitation techniques, as opposed to using automated reasoning. Secure Tropos [14] models security concerns from early requirements to design. It expresses security requirements as *security constraints*, considers potential threats and attacks, and provides methodological steps to validate these requirements and overcome vulnerabilities. Their analysis identifies secure goals and entities to guarantee the satisfaction of security constraints, and gives secure capabilities to agents to guarantee the satisfaction of security entities. STS clearly separates between security constraints over goal delegations and over information. This allows us to be clearer, while being expressive and supporting more types of security requirements.

Liu et al. [4] extend *i** to deal with security and privacy requirements. Their method defines security- and privacy-specific analysis mechanisms to identify potential attackers, derive threats and vulnerabilities, thereby suggesting countermeasures. We support a significantly larger set of security requirements, while differentiating information flow from permissions and prohibitions represented through sophisticated authorisations.

Haley et al. [12] construct the system context with problem frames [38], security requirements are defined as constraints over functional requirements, and a structure of satisfaction arguments is built to verify the correctness of security requirements. The argument fails if the security requirements are not satisfiable in the context, or the context is insufficient to develop the argument. This approach focuses mainly on system requirements, while ours is centred on user-specific security requirements emerging from the interactions among actors.

7. Conclusions

We have proposed the STS approach for modelling and reasoning over security requirements for socio-technical systems. The distinguishing features of STS are: (i) analysing the problem domain in terms of both social and technical aspects, and (ii) relating security requirements to the interactions among the actors in the socio-technical system. Our approach is based on a revised version of STS-ml [15], the security requirements modelling language for socio-technical systems.

STS-ml models employ a high-level representation of assets, modelling actors' goals, information, and documents. STS-ml clearly distinguishes information from its representation (in terms of documents), and keeps track of information structure and permissions granted or prohibitions specified over information. This allows STS-ml to treat information as a first-class citizen (goals and documents are considered relevant from a security point of view because of the information they use and represent, respectively) when dealing with information security, and at the same time support a rich set of security requirements types.

This high expressiveness is ensured by clearly distinguishing actors' intentional (goals) and informational assets (documents), keeping track of information ownership and the restrictions actors impose over their information (via authorisations), apart from considering security issues over actors' interactions.

STS-ml comes with a well-defined semantics, as presented by the formal framework, which supports automated analysis.

The modelling and analysis activities of requirements engineers and analysts are supported by the STS-Tool. The STS-Tool provides a graphical interface for creating STS-ml models through three different views: social, information, and authorisation view. Multi-view modelling leads to the construction of three corresponding (sub)models of the said STS-ml model, namely social model, information model, and authorisation model. The tool supports inter-model consistency by allowing for well-formedness checks on the fly, ensuring that the models comply with the syntax of the modelling language.

STS-Tool supports automated reasoning by integrating the formal framework of STS-ml. Security analysis is implemented in Disjunctive Datalog and the DLV Engine is integrated in the STS-Tool to support this analysis.

We have shown how to detect two types of conflicts: (1) among security requirements (mainly on authorisations); 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.

Limitations. The proposed approach suffers from the following limitations: (i) design-time analysis—the proposed formal framework covers the verification of a subset of STS-ml security requirements types that can be verified at design-time. An extension of the framework is needed to support run-time analysis; (ii) goal-modelling limitations—cognitive scalability due to the size of the output models, and a steep learning curve. Being a goal-oriented modelling language, STS-ml inherits these problems from the family of languages it pertains; (iii) standardisability—the proposed taxonomy is inspired by standards, but not a standard one. The security requirements supported by STS-ml are refined starting from standard security principles, to comply to the terminology used in traditional security for software systems. The STS-ml taxonomy relies on six security principles that are assembled from mainstream taxonomies and security standards, while their concrete representation and specification is naturally linked to STS-ml concepts and relationships. As such, efforts are required to standardize the STS-ml taxonomy.

Ongoing and future work. While the STS approach (comprising STS-ml, the formal framework, and STS-Tool) is quite mature, several future directions remain open: (i) *Assessing and analysing privacy requirements*: STS supports privacy as confidentiality, by expressing security requirements concerning the access and manipulation of information. Although expressive in terms of security requirements it supports, our approach does not support other types of privacy requirements, such as *privacy as control* or *privacy as practice* [39,40]; (ii) *Informing later phases*: explore how STS-ml can inform later phases for the design of secure socio-technical systems, such as, for instance, the definition of access control policies. This will require mapping the security requirements specification derived from STS, to specification policy languages, e.g., XACML; (iii) *Optimising reasoning techniques*: devise further analysis techniques for more sophisticated reasoning. In particular, efforts will be dedicated to further optimising reasoning techniques by: (i) checking whether there is a variant in which there is no conflict, and checking whether there is one variant in which there is a conflict for a given security requirement, in order to avoid generating all possible variants. The use of modelling checking should further improve the scalability of the reasoning techniques; and (ii) setting priorities among security requirements according to their severity in order to eliminate some of the possible conflicts [41,42]. Prioritization is a first step towards the resolution of conflicts. However, when prioritization is not applicable, we will explore alternative techniques for conflict resolution [30,29];

Acknowledgements

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). The authors thank Alexander Borgida for his useful suggestions.

References

[1] F.E. Emery, Characteristics of socio-technical systems, Tech. Rep., 527, Tavistock Institute, London, 1959.

[2] I. Sommerville, D. Cliff, R. Calinescu, J. Keen, T. Kelly, M. Kwiatkowska, J. Mcdermid, R. Paige, Large-scale complex IT systems, Commun. ACM 55 (7) (2012) 71–77.

[3] F. Dalpiaz, P. Giorgini, J. Mylopoulos, Adaptive socio-technical systems: a requirements-driven approach, Requir. Eng. 18 (4) (2013) 1–24.

[4] L. Liu, E. Yu, J. Mylopoulos, Security and privacy requirements analysis within a social setting, Proceedings of the 11th IEEE International Requirements Engineering Conference 2003, pp. 151–161.

[5] P. Giorgini, F. Massacci, J. Mylopoulos, Requirement engineering meets security: a case study on modelling secure electronic transactions by VISA and Mastercard, Proceedings of the 22nd International Conference on Conceptual Modeling, LNCS, vol. 2813 2003, pp. 263–276.

[6] D. Mellado, C. Blanco, L.E. Sánchez, E. Fernández-Medina, A systematic review of security requirements engineering, Comput. Stand. Interfaces 32 (4) (2010) 153–165.

[7] D.G. Firesmith, Security use cases, J. Object Technol. 2 (3) (2003) 53–64.

[8] N.R. Mead, E.D. Hough, T.R. Stehney II, Security quality requirements engineering (SQUARE) methodology, Tech. Rep. CMU/SEI-2005-009, 2005.

[9] J. McDermott, C. Fox, Using abuse case models for security requirements analysis, Proceedings of the 15th Annual Computer Security Applications Conference 1999, pp. 55–64.

[10] G. Sindre, A.L. Opdahl, Eliciting security requirements with misuse cases, Requir. Eng. 10 (1) (2005) 34–44.

[11] A. van Lamsweerde, Elaborating security requirements by construction of intentional anti-models, Proceedings of the 26th International Conference on Software Engineering 2004, pp. 148–157.

[12] C.B. Haley, R.R. Laney, J.D. Moffett, B. Nuseibeh, Security requirements engineering: a framework for representation and analysis, IEEE Trans. Softw. Eng. 34 (1) (2008) 133–153.

[13] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, Modeling security requirements through ownership, permission and delegation, Proceedings of the 13th IEEE International Conference on Requirements Engineering 2005, pp. 167–176.

[14] H. Mouratidis, P. Giorgini, Secure Tropos: a security-oriented extension of the Tropos methodology, Int. J. Softw. Eng. Knowl. Eng. 17 (2) (2007) 285–309.

[15] F. Dalpiaz, E. Paja, P. Giorgini, Security requirements engineering via commitments, Proceedings of the First Workshop on Socio-technical Aspects in Security and Trust 2011, pp. 1–8.

[16] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, B. Nuseibeh, Inconsistency handling in multiperspective specifications, IEEE TSE 20 (8) (1994) 569–578.

[17] S. Trösterer, E. Beck, F. Dalpiaz, E. Paja, P. Giorgini, M. Tscheligi, Formative user-centered evaluation of security modeling: results from a case study, Int. J. Secure Softw. Eng. 3 (1) (2012) 1–19.

[18] E. Paja, F. Dalpiaz, M. Poggianella, P. Roberti, P. Giorgini, Modelling security requirements in socio-technical systems with STS-Tool, Proceedings of the Forum at the 24th International Conference on Advanced Information Systems Engineering, CEUR-WS, vol. 855 2012, pp. 155–162.

[19] E. Paja, F. Dalpiaz, M. Poggianella, P. Roberti, P. Giorgini, STS-Tool: socio-technical security requirements through social commitments, Proceedings of the 20th IEEE International Conference on Requirements Engineering 2012, pp. 331–332.

[20] E. Paja, F. Dalpiaz, P. Giorgini, Managing security requirements conflicts in socio-technical systems, Proceedings of the 32nd International Conference on Conceptual Modeling, LNCS, vol. 8217 2013, pp. 270–283.

[21] P. Shvaiko, L. Mion, F. Dalpiaz, G. Angelini, The TasLab portal for collaborative innovation, Proceedings of the 16th International Conference on Concurrent Enterprising, 2010.

[22] F. Dalpiaz, A.K. Chopra, P. Giorgini, J. Mylopoulos, Adaptation in open systems: giving interaction its rightful place, Proceedings of the 29th International Conference on Conceptual Modeling, LNCS, vol. 6412 2010, pp. 31–45.

[23] D. Gollmann, Computer Security, 3rd edition John Wiley & Sons, 2011.

[24] C.P. Pfleeger, S.L. Pfleeger, Analyzing Computer Security: A Threat/Vulnerability/Countermeasure Approach, Prentice Hall, 2012.

[25] R. Kissel, Glossary of key information security terms, Tech. Rep. IR 7298 Rev 1, NIST, 2011.

[26] BS ISO/IEC 27005:2011, Tech. rep. ISO/IEC, 2011.

[27] E. Paja, F. Dalpiaz, M. Poggianella, P. Roberti, P. Giorgini, STS-Tool: socio-technical security requirements through social commitments, Proc. of RE'12 2012, pp. 331–332.

[28] E. Paja, F. Dalpiaz, P. Giorgin, Identifying conflicts in security requirements with STS-ml, Tech. Rep. DISI-12-041, University of Trento, 2012 (http://disi.unitn.it/paja/tr-identifying-sec-conflicts.pdf).

[29] G. Elahi, E. Yu, A goal oriented approach for modeling and analyzing security trade-offs, Proceedings of the 26th International Conference on Conceptual Modeling, LNCS, vol. 4801 2007, pp. 375–390.

[30] A. van Lamsweerde, R. Darimont, E. Letier, Managing conflicts in goal-driven requirements engineering, IEEE Trans. Softw. Eng. 24 (11) (1998) 908–926.

[31] A. Fuxman, M. Pistore, J. Mylopoulos, P. Traverso, Model checking early requirements specifications in Tropos, Proceedings of the 5th IEEE International Symposium on Requirements Engineering 2001, pp. 174–181.

[32] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, R. Sebastiani, Reasoning with goal models, Proceedings of the 21st International Conference on Conceptual Modeling, LNCS, vol. 2503 2002, pp. 167–181.

[33] P. Giorgini, J. Mylopoulos, R. Sebastiani, Goal-oriented requirements analysis and reasoning in the Tropos methodology, Eng. Appl. Artif. Intell. 18 (2) (2005) 159–171.

[34] J. Horkoff, E. Yu, Evaluating goal achievement in enterprise modeling—an interactive procedure and experiences, The Practice of Enterprise Modeling, Springer 2009, pp. 145–160.

[35] J. Horkoff, E. Yu, Finding solutions in goal models: an interactive backward reasoning approach, Proceedings of the 29th International Conference on Conceptual Modeling, LNCS, vol. 6412 2010, pp. 59–75.

[36] E. Yu, Modelling Strategic Relationships for Process Reengineering(Ph.D. thesis) University of Toronto, Canada, 1996.

[37] R. De Landtsheer, A. Van Lamsweerde, Reasoning about confidentiality at requirements engineering time, Proceedings of the 13th ACM SIGSOFT International Symposium on the Foundations of Software Engineering 2005, pp. 41–49.

[38] M. Jackson, Problem Frames: Analysing and Structuring Software Development Problems, Addison-Wesley, 2001.

[39] S. Gürses, Multilateral Privacy Requirements Analysis in Online Social Networks(Ph.D. thesis) HMDB, Department of Computer Science, KU Leuven, Belgium, May 2010.

[40] B. Berendt, More than modelling and hiding: towards a comprehensive view of Web mining and privacy, Data Min. Knowl. Disc. 24 (3) (2012) 697–737.

[41] S. Liaskos, S.A. McIlraith, S. Sohrabi, J. Mylopoulos, Integrating preferences into goal models for requirements engineering, 18th IEEE International Requirements Engineering Conference, IEEE 2010, pp. 135–144.

[42] S. Liaskos, S.A. McIlraith, S. Sohrabi, J. Mylopoulos, Representing and reasoning about preferences in requirements engineering, Requir. Eng. 16 (3) (2011) 227–249.