STS-Tool: Security Requirements Engineering for Socio-Technical Systems

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

[1] University of Trento, Italy – {elda.paja, paolo.giorgini}@unitn.it
[2] Utrecht University, The Netherlands – f.dalpiaz@uu.nl

Abstract. We present the latest version of STS-Tool, the modelling and analysis support tool for STS-ml, an actor- and goal-oriented security requirements modelling language for socio-technical systems. STS-Tool allows designers to model a socio-technical system in terms of high-level primitives such as actor, goal, and delegation; to express security constraints over the interactions between the actors; and to derive security requirements once the modelling is done. The tool features a set of automated reasoning techniques for (i) checking if a given STS-ml model is well-formed, and (ii) determining if the specification of security requirements is consistent, that is, there are no conflicts among security requirements, and (iii) calculating the threat trace of events threatening actors' assets.

1 Introduction

Socio-Technical Systems are an interplay of social actors (human and organizations) and technical subsystems, which interact with one another to reach their objectives [2]. Each participant acts, guided by its objectives, according to its business policies, and the socio-technical system comes into existence when the participants interact to get things done. In e-commerce, buyers and sellers interact with one another (social actor - social actor interaction) making use of the web application (social actor - technical actor interaction), which relies on secure channels (technical actor - technical actor interaction) to ensure the transactions among the buyer and the seller are secure. The diversity and autonomy of participants makes the design of a secure socio-technical system a challenging task, for which the study of technical aspects alone is not enough, instead social aspects need to be investigated too.

STS-ml [7,3] (Socio-Technical Security modelling language), is an actor (to represent the various stakeholders) and goal-oriented (capturing their main objectives) security requirements modelling language based upon these principles. It allows tackling security issues already in the early phases of socio-technical system design. STS-ml is based on the idea of relating security requirements to *interaction*. The language allows stakeholders to express *security needs* over their interactions to constrain the way interaction is to take place. For example, if a buyer sends its personal data to a seller, the buyer may require the data not to be disclosed to third parties, only the seller should have access to them. STS-ml specifies security requirements in terms of *social commitments* [8], promises with contractual validity made by an actor to another. One actor commits (responsible) to another (requester) that, while delivering some service, it will

comply with the required security needs. In the example above, a security requirement is that the seller commits not to disclose buyer's personal data to other parties.

Differently from other goal-oriented approaches to security requirements engineering [4,5,6], STS-ml offers a more expressive ontology. It supports expressing fine-grained and contradictory authorisations over information, which allow to effectively represent real-world information security requirements [1,9].

We show how STS-Tool, the case tool for STS-ml, supports requirements analyst and security engineers in designing secure socio-technical systems. The tool is the result of an iterative development process, following several evaluation activities, in the scope of the FP7 European Project Aniketos [3]. It has been used in modelling and reasoning over models of a large size from different domains, such as eGovernment, Telecommunications, and Air Traffic Management Control. We first introduce STS-ml in Sect. 2 as the baseline for using STS-Tool, briefly presenting its modelling features (Sect. 2.1) and then providing the users with a modelling process to design socio-technical systems with STS-ml and STS-Tool (Sect. 2.2). Sect. 3 describes the main features of STS-Tool, while Sect. 4 demonstrates the use of the tool in modelling a scenario from eGovernment, guiding the user step by step through a series of exercises that follow the phases of the modelling method. Finally, Sect. 5 concludes with a discussion of related work and future directions.

2 Baseline: STS-ml

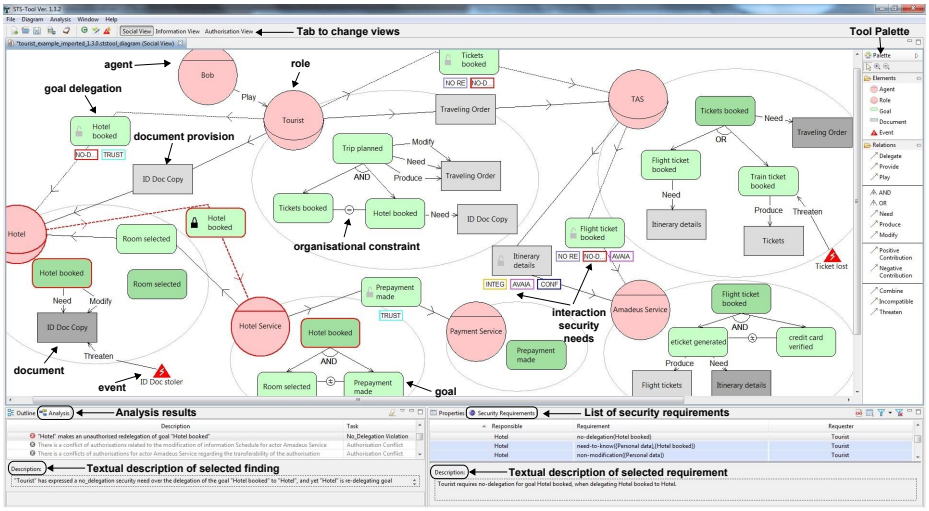
STS-ml is an actor- and goal-oriented security requirements engineering modelling language. It includes high-level organisational concepts such as actor, goal, delegation, etc.

STS-ml is a diagrammatical language, i.e., graphical concepts and relations are used to create the models. A particular feature of STS-ml is that it allows modelling socio-technical systems by focusing on different perspectives (views) at a time. This feature is known as: *multiview modelling*. Below, we introduce STS-ml constructs and show how they are used to model socio-technical systems. Additionally, we present an iterative process to model socio-technical system with STS-ml, as well as to analyze the models built, to then derive security requirements for the system to be.

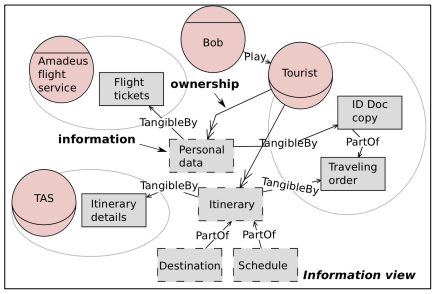
2.1 Multiview modelling with STS-ml

STS-ml modelling consists of three complementary views, *social*, *information*, and *authorisation* view. The three views together form the overall model for the system-at-hand (STS-ml model). To facilitate the description of STS-ml, we will use a small running example.

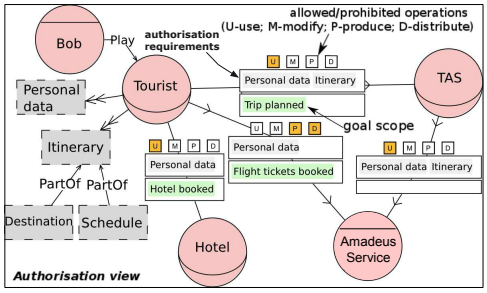
Example 1 (Travel Planning). A tourist wants to organise a trip using a Travel Agency Service (*TAS*). *TAS* allows end users to read about various destinations, book flights and hotels, hire a car, etc., and uses the *Amadeus flight service* to book flight tickets. To book hotels, the *Tourist* has chosen to directly contact the *Hotel* himself, without interacting with *TAS*.



(a) Social view



(b) Information view



(c) Authorisation view

Fig. 1: Multi-view modelling for the travel planning scenario

The *social view* (Fig. 1a) represents actors as intentional and social entities. Actors are intentional as they aim to attain their objectives (*goals*), and they are social, for they interact with others to fulfill their objectives (by *delegating goals*) and obtain information (by *exchanging documents*). STS-ml supports two types of actors: *agents*—concrete participants, and *roles*—abstract actors, used when the actual participant is unknown. In our example, we represent the *TAS* as a role, and the *Amadeus Service* as an agent, as it refers to a specific flight service. Actors may *possess* documents, they may *use*, *modify*, or *produce* documents while achieving their goals. For instance, *Tourist* wants to achieve the goal *Trip planned*, for which it needs to both have *Tickets booked* and *Hotel booked*. To book the hotel it needs document *ID Doc copy*.

The *information view* (Fig. 1b) shows how information and documents are interconnected to identify which information actors manipulate, when they *use*, *modify*, *pro-*

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

duce, or *distribute* documents to achieve their goals in the social view. Additionally, it gives a structured representation of actors' information and documents. Information can be represented by one or more documents (through Tangible By), and on the other hand one or more information entities can be part of some document. For instance, information *Personal data* is represented by both *ID Doc copy* and *Flight tickets* documents.

The *authorisation view* (Fig. 1c) shows the authorisations actors grant to others over information, specifying which operations they are allowed (prohibited) to do, for which goals (scope), and whether authorisation can be further transferred or not. For instance, *Tourist* authorises *TAS* to use (U selected) information *Personal data* and *Itinerary* in the scope of the goal *Tickets booked* granting a transferrable authorisation (authorisation's arrow line is continuous).

Through its three views, STS-ml supports different types of security needs:

- *Interaction (security) needs* are security-related constraints on *goal delegations* and *document provisions*, e.g., non-repudiation, integrity of transmission, etc.;
- *Authorisation needs* determine which information can be used, how, for which purpose, and by whom, e.g. non-disclosure, need-to-know;
- *Organisational constraints* constrain the adoption of roles and the uptake of responsibilities, e.g. separation or binding of duties, conflicting or combinable goals.

Together, these needs constitute the security needs of STS-ml, from which the security requirements can be derived. In STS-ml, security requirements are social relationships where an actor (*responsible*) commits to another actor (*requester*) to comply with a requested security need. That is, for each security need, a security requirement to satifsy the need is generated. For more details, see Section 2.2, Phase 5.

How can we construct the presented views, and have the model presented in Fig. 1? How can we express actors' security needs to then derive security requirements? How can we verify validity of the model and verify compliance with security requirements?

In the following section we will describe in detail how to built the various views, how to capture security requirements, while introducing and expressing the security needs supported by STS-ml, and how to analyze the created models.

2.2 The STS method

We provide an iterative process, which supports modelling and analysing socio-technical systems, namely the STS method (see Fig. 2), to facilitate the work of the requirements analysts. Note, however, that the provided steps are a guideline for them, not necessarily mandatory to be followed in the indicated order.

Phase 1. Modelling the Social view

Step 1.1. Identify stakeholders. As described above, stakeholders in STS-ml are represented via agents and roles [4]. Role is an abstract characterization of the behavior of an active entity within some context. Most participants are unknown at design time, e.g., Tourist, Travel Agency Service (TAS), Hotel, etc. Agents play (adopt) roles at runtime, and they can change the roles they play, e.g., Bob, John, CheapTravels Inc. Some agents

[4] We use the general term *actor* whenever relationships are applicable to both agents and roles

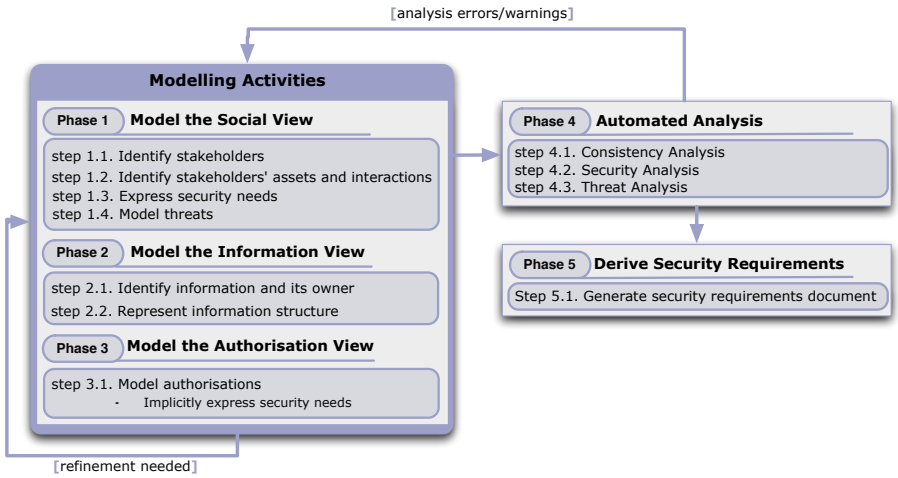


Fig. 2: The STS method

are known already at design time when we build STS-ml models, e.g., Amadeus Service. Fig. 3 shows how roles and agents are represented graphically in STS-ml. See the identified roles and agents for the tourist example in Fig. 1.

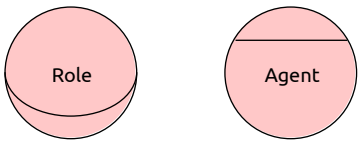
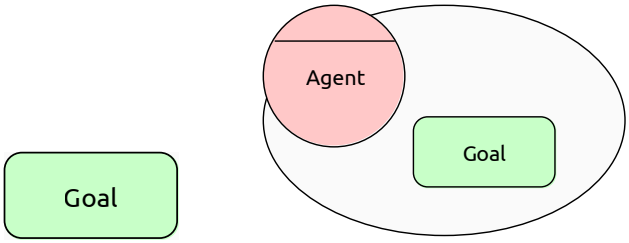


Fig. 3: Graphical representation of roles and agents

Step 1.2. Identify assets and interactions. When talking about security, stakeholders of a system wish to protect their important assets. Stakeholders in the socio-technical system participate in order to achieve their desired objectives—modelled in STS-ml through the concept of *goal*. A goal is a *state of affairs that an actor intends to achieve*, e.g., trip planned, flight tickets booked. They are used to capture motivations and responsibilities of actors. Thus, we consider goals as assets for an actor, and refer to them as *intentional assets*. In addition, their owned *information* entities are another important asset for stakeholders. We refer to them as *informational assets*. The graphical representation of goals and actors' intention to fulfil them is shown in Fig. 4.

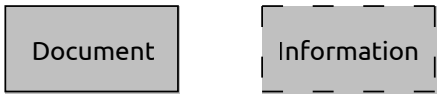
Information as is, cannot be exchanged among actors, nor can it be manipulated by them. We use the concept of *documents*—representing information—to allow stakeholders to make use of and exchange information. Information and documents are graphically represented as shown in Fig. 5.

In Fig 1, *Amadeus Service* has the goal *Flight tickets booked*, for which it is responsible. Goals can be refined into subgoals through *and/or-decompositions*: and-decomposition (Fig. 6a) represents the process of achieving a goal, while or-decomposition



(a) Goal (b) Intention

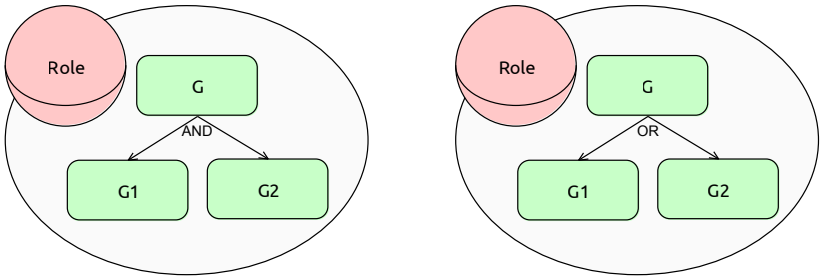
Fig. 4: Graphical representation of goals and intentions



(a) Document (b) Information

Fig. 5: Graphical representation of informational assets

(Fig. 6b) represents alternative ways for achieving a goal. In Fig. 1a, *TAS* or-decomposes goal *Tickets booked* into goals *Flight ticket booked* and *Train ticket booked*.



(a) And-decomposition (b) Or-decomposition

Fig. 6: Graphical representation of goal and/or decompositions

The goal model of an actor ties together goals and documents, in various ways: an actor *possesses* a set of documents; an actor *needs* one or more documents to fulfil a goal; an actor *produces* documents while fulfilling a goal; an actor *modifies* a document while fulfilling a goal, see Fig. 7. The relation *possesses* indicates that actors have a specific document, i.e., they can furnish or provide it to other roles. Graphically, this is represented by including a document in the actor's scope, see Fig. 8a. Note that this is different from ownership (Fig. 8b), an actor might have a document without necessarily being the owner of the information it contains. For instance, *TAS* possesses document

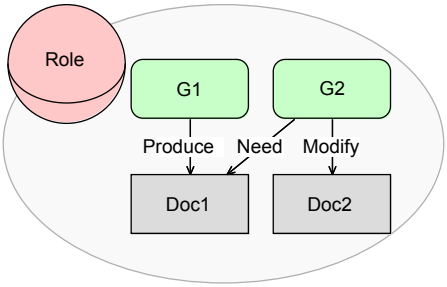
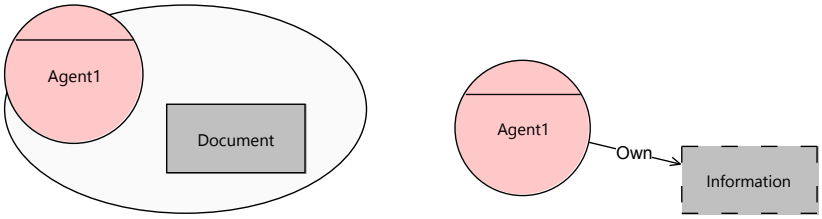


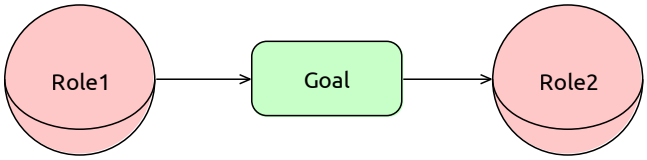
Fig. 7: Graphical representation of goal-document relationships

Traveling order, but it is not the owner of information *Personal data*, the *Tourist* is, see Fig. 1a and 1b.

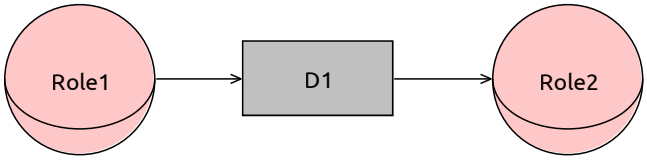


(a) Possession (b) Ownership

Fig. 8: Graphical representation of document possession and information ownership



(a) Goal delegation



(b) Document provisions

Fig. 9: Graphical representation of goal delegation and document provision

Within the same step, we need to identify interactions among actors as well. *Goal delegations* capture the transfer of responsibility for the fulfillment of the goal from one actor to another, i.e., a delegator actor delegates the fulfillment of a goal (delega-

tum) to another actor (delegatee), see Fig. 9a. Note that in STS-ml, only leaf goals can be delegated, in Figure 1a *Tourist* delegates to *TAS* the fulfillment of *Tickets booked*. *Document provision*, on the other hand, specifies the exchange of information between actors, a sender actor provides a document to a receiver actor, see Fig. 9b. Providing the document refers strictly to the actual supply or delivery of the document. Information as is (e.g. ideas) cannot be transferred if not explicitly made concrete by a document (e.g. a paper, an e-mail). In STS-ml, a document can be provided only by an actor that possesses it, see in Fig. 1a how *Tourist* provides document *Traveling order* to *TAS*.

Step 1.3. Express security needs. STS-ml allows actors to express security needs over their interactions, in this step we analyze these interactions (goal delegations and document provisions) actors participate in to elicit their needs with regard to security. To specify security needs over goal delegations and document provisions, these relationships are annotated via security needs the interacting parties (being them agents or roles) want each other to comply with.

STS-ml supports the following interaction security needs:

1. *Over goal delegations:*
   (a) *No-redelegation*—the re-delegation of the fulfilment of a goal is forbidden; In Fig. 10, *Tourist* requires *Hotel* not to redelegate goal *Hotel booked*.
   (b) *Non-repudiation*—the delegator cannot repudiate he delegated (non-repudiation of delegation); and the delegatee cannot repudiate he accepted the delegation (non-repudiation of acceptance); for instance, *TAS* requires *Amadeus Service* non-repudiation of the delegation of goal *Flight ticket booked*, see Fig. 10.

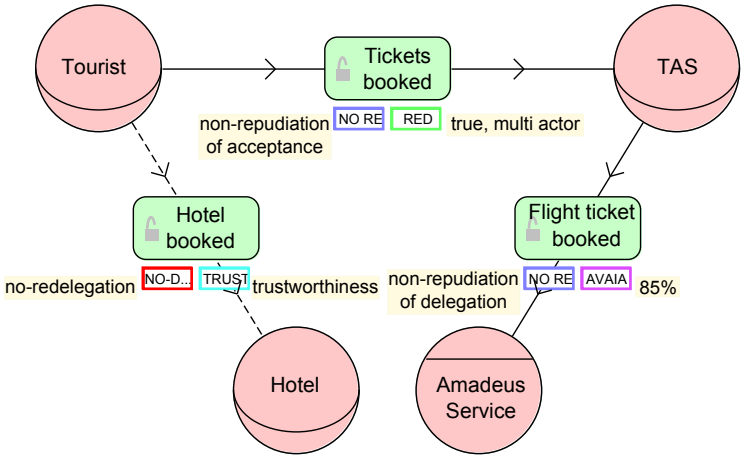


Fig. 10: Security needs over goal delegations

   (c) *Redundancy*—the delegatee has to employ alternative ways of achieving a goal; We consider two types of redundancy: *True* and *Fallback*. *True redundancy*: at least two or more different strategies are considered to fulfil the goal, and they are executed simultaneously to ensure goal fulfillment. *Fallback redundancy*: a primary strategy is selected to fulfill the goal, and at the same time a number of other strategies is considered and maintained as backup to fulfill the goal.

None of the backup strategies is used as long as the first strategy successfully fulfils the goal. Within these two categories of redundancy, two sub-cases exist: (i) only one actor employs different strategies to ensure redundancy: single actor redundancy; and (ii) multiple actors employ different strategies to ensure redundancy: multi actor redundancy. In total, we can distinguish four types of redundancy, which are all mutually exclusive, so we can consider them as four different security needs, namely, (i) fallback redundancy single, (ii) fallback redundancy multi, (iii) true redundancy single, and (iv) true redundancy multi. In Fig. 10, *Tourist* requires *TAS* true redundancy multi for goal *Tickets booked*.

(d) *Trustworthiness*—the delegation of the goal will take place only if the delegatee is trustworthy; for instance, the delegation of goal *Hotel booked* from *Tourist* to *Hotel* will take place only to trustworthy hotels, see Fig. 10.

(e) *Availability*—the delegatee should ensure a minim availability level for the delegated goal; for instance, *TAS* requires *Amadeus Service* 85% availability for goal *Flight ticket booked*, see Fig. 10.

2. *Over document provisions:*
   (a) *Integrity of transmission*—the sender should ensure that the document shall not be altered while providing it;
   (b) *Confidentiality of transmission*—the sender should ensure the confidentiality of transmission for the provided document;
   (c) *Availability*—the sender should ensure a minimal availability level (in percentage) for the provided document. In Fig. 11, *TAS* should ensure integrity and confidentiality of transmission, as well as an availability level of 98% when providing the document *Itinerary details* to *Amadeus flight service*.

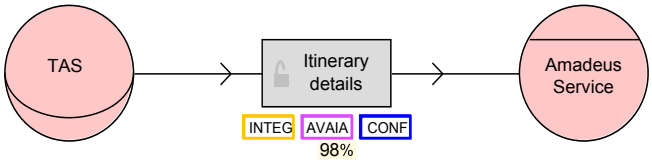


Fig. 11: Security needs over document provisions

3. *From organisational constraints* [5]*:*
   (a) *Separation of duties* (SoD)—defines incompatible roles and incompatible goals, so we define two types: role-based SoD—two roles are incompatible, i.e., cannot be played by the same agent, and goal-based SoD—two goals shall be achieved by different actors; for instance, the goals *eticket generated* and *credit card verified* are defined as incompatible (unequals sign, see Fig. 12).
   (b) *Combination of duties* (CoD)—defines combinable roles and combinable goals, so we distinguish between role-based CoD—two roles are combinable, i.e., shall be played by the same agent; and goal-based CoD—two goals shall be achieved by the same actor. Note that these security needs from organisational constraints are translated to a set of relationships, *incompatible* (represented

[5] Organisational constraints are imposed either by the rules and regulations of the organisation, or by law.

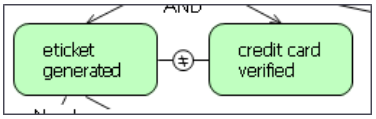


Fig. 12: Security needs from organisational constraints

as a circle with the unequal sign within) and *combines* (represented as a circle with the equals sign within) respectively. This is related to the fact that they are not directly expressed over a social relationship, but constrain the uptake of responsibilities of stakeholders. Both relationships are symmetric, therefore there are no arrows pointing to the concepts they relate.

Step 1.4. Modelling threats. In STS-ml we represent events threatening stakeholders' assets. STS-ml proposes the concept *event* and the relationship *threaten* relating the event to the asset it threatens. As introduced earlier, we consider two types of assets, *intentional assets* and *informational assets* respectively. However, in the social view stakeholders exchange and manipulate information via documents, so in this step we model the events that threaten actors' *goals* and *documents* respectively. For instance, the event *ID Doc Copy lost* threatens document *ID Doc Copy*, see Fig. 1. Broadly, an event threatening a goal means that the goal cannot be reached, while an event threatening a document means that the document becomes unavailable.

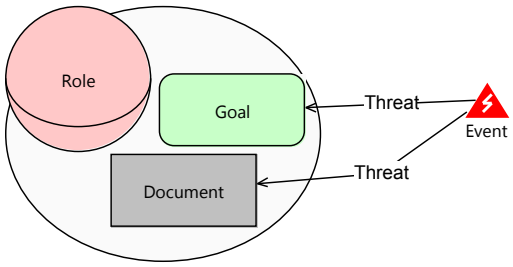


Fig. 13: Graphical representation of events threatening actors' assets in STS-ml

Phase 1. Summary. Each of the above steps is repeated until all stakeholders have been modelled, all their assets and interactions have been represented, their desired security needs have been expressed, and events threatening actors' assets have been modelled. The result of this iterative modelling process supported by **Phase 1** is Fig. 1a.

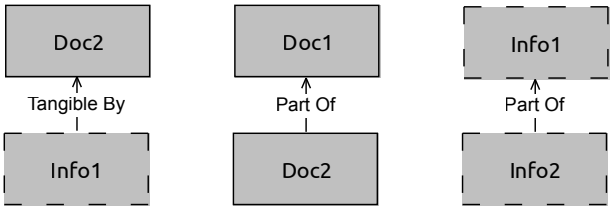
Phase 2. Modelling the Information view

To protect information, it is important to first identify information, its representation (documents), and to know who information owners are, for they are the ones concerned with what happens to their information.

Step 2.1. Identify information and its owner. Documents represent information, so when modelling the information view we first identify which are the informational entities represented by each document in the social view. For each identified information, we identify who are the owners of this information. For instance, *Tourist* is the owner

of his *Personal data*, see Fig. 1b. Note that there can be multiple owners for the same information, to represent shared ownership.

Step 2.2. Represent information structure. Information view gives a structured representation of actors' information and documents, and the way they are interconnected. Information can be represented by one or more documents (through the Tangible By relationship), see Fig. 14a. On the other hand, one or more information pieces can be made tangible by the same document. In Fig. 1b, information *Personal data* is made tangible by document *Eticket* and document *ID Doc Copy*.



(a) Tangible by (b) Part-of (c) Part-of

Fig. 14: Graphical representation of part-of and tangible by

Another feature of the information view is to support composite information (documents). We enable that by means of the part Of relations (see Fig. 14b and 14c), which can be applied between information (documents). For instance, this allows representing that information *Destination* and *Schedule* are part of the information *Itinerary*, while document *ID Doc Copy* is part of document *Traveling Order* (see Fig. 1b).

Phase 2. Summary. These steps are repeated till all important information entities are represented, their owners are identified and they are connected to their corresponding documents or parts of information. The result of the iterative modelling process supported by **Phase 2** is Fig. 1b.

Phase 3. Modelling the Authorisation view

An adequate representation of authorisations is necessary to determine if information is exchanged and used in compliance with confidentiality restrictions. Information owners are the unique actors that can legitimately transfer rights to other actors. However, when they transfer full rights to another actor, the latter becomes entitled to transfer the same rights the owner can grant.

Step 3.1. Model authorisations. Authorisations support the transfer of rights between two actors. An actor can grant (receive) an arbitrary number of authorisations about information, specifying:

- *Operations*: refer to actions that an actor can perform on the information. STS-ml supports four basic operations: *use* (U), *modify* (M), *produce* (P), and *distribute* (D). The four supported operations are directly linked to the way information is manipulated within an actor's model in the social view. Usage goes in parallel with the need relation; modification relates to the modify relation, production is reflected by the produce relation, and distribution relates to the document provision relation

between actors. Graphically the allowed operation is highlighted in yellow (see Fig. 15). In Fig. 1c, *Tourist* authorises *TAS* on usage (U is selected).

– *Information*: the transferred rights are granted over at least one information entity. In Fig. 15, authority to *use* and *modify* information *Info1* and *Info2* is granted to *Role2*. In our running example, the authorisation from *Tourist* to *TAS* is granted over information *Personal data* and *Itinerary*, see Fig. 1c.

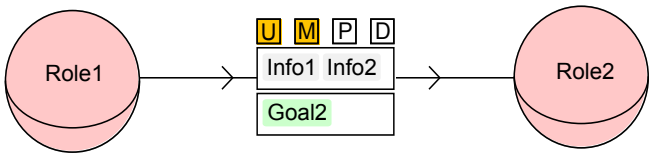


Fig. 15: Graphical representation of authorisations

– *Scope*: authority over information can be limited to their usage in the scope of a certain goal. Our notion of goal scope includes the goal tree rooted by that goal. As a result, authority is given to manipulate information not only for the specified goal, but for its sub-goals as well. For instance, the *Tourist* permits the *TAS* to use his *Personal data* only to book the tickets (i.e., for goal *Tickets booked*), see Fig. 1c.
– *Transferability*: this dimensions allows to specify whether the actor receiving the authorisation can further re-authorise actors, that is, the authorisee not only is granted the permission to perform operations, but also that of further propagating rights over the specified information to other actors. Note that reauthorisation should be compatible with the authority scope the delegator has. Non-transferrability is represented through a dashed authorisation line to show that the authorisation chain should end with the authorisee. For instance, the authorisation from *Tourist* to *TAS* is transferrable, while that from *TAS* to *Amadeus Service* is not, see Fig. 1c.

Implicitly express security needs. Security needs over authorisations are expressed by allowing only certain operations and limiting the scope:

– limiting the scope expresses a security need on *Need-to-know*—requires information is used, modified, produced only for the specified scope; for instance, *Tourist* expresses a need-to-know security need to *Hotel*, which can use *Personal data* only in the scope of *Hotel booked*, see Fig. 16.
– not allowing usage expresses a security need on *Non-usage*— requires the information is not used in an unauthorised way; it implies that the authorisee should not make use of (need) any documents making tangible the specified information. There are no cases of non-usage from our running example.
– not allowing modification expresses a security need on *Non-modification*—requires the information is not modified in an unauthorised way; it implies that the authorisee should not modify any documents making tangible this information. In Fig. 16, *Hotel* cannot modify documents representing *Personal data*.
– not allowing production expresses a security need on *Non-production*—requires the information is not produced in an unauthorised way; it implies that no new document, representing the given information, is produced. In Fig. 16, *TAS* cannot produce documents that represent *Personal data* or *Itinerary*.

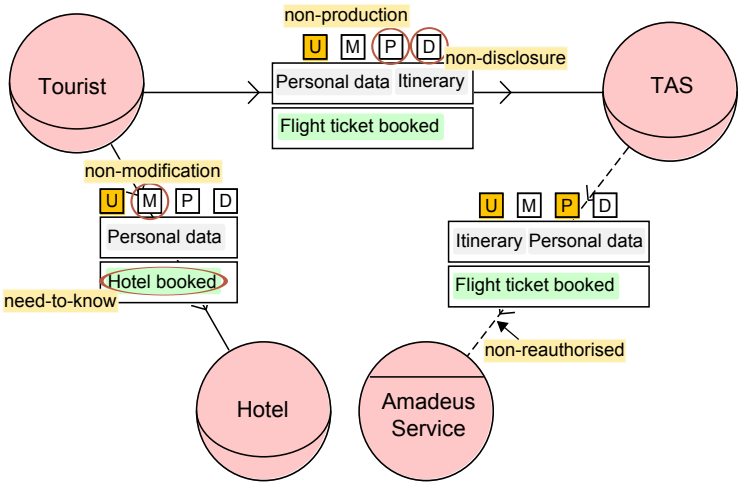


Fig. 16: Security needs over authorisations

- not allowing distribution expresses a security need on *Non-disclosure*—requires the information is not disclosed in an unauthorised way; it implies that no document, representing the given information, is transmitted to other actors. In Fig. 16, *TAS* cannot distribute documents representing *Personal data* or *Itinerary*.
- not allowing transferrability expresses a security need on *Not-reauthorise*—requires the authorisation is not transferrable, i.e., the authorisee does not further transfer rights either for operations not granted to him (implicitly) or when the transferability of the authorisation is set to false (explicitly). This means that any non-usage, non-modification, non-production or non-disclosure security need implies a not-reauthorise security need for the operations that are not allowed. In Fig. 16, *Amadeus Service* cannot further authorise other actors, for the authorisation coming from *TAS* is non-transferable.

Phase 3. Summary. The steps are repeated until all authorisation relationships have been drawn and authorisation needs have been expressed. The result of the iterative modelling process supported by **Phase 3** is Fig. 1c.

Phase 4. Automated Analysis

After the security requirements engineer has performed the modelling activities, the STS method allows him to perform automated analysis over the created STS-ml model. The analyses are supported by the case tool of STS-ml, namely STS-Tool. Details on the tool will follow in Section 3. Currently three types of analysis are supported:

- Consistency Analysis (*step 4.1.*)
- Security Analysis (*step 4.2.*)
- Threat Analysis (*step 4.3.*)

Step 4.1. Consistency Analysis. The purpose of consistency analysis is to verify whether the diagram built by the designer is consistent and valid. It is also referred to as

Offline well-formedness analysis: some well-formedness rules of STS-ml are computationally too expensive for online verification, or their continuous analysis would limit the flexibility of the modelling activities. Thus, some analyses about well-formedness are performed upon explicit user request. Examples of verifications include delegation cycles, part-of cycles, inconsistent or duplicate authorisations, etc.

The results of the analysis are shown in the Analysis tab (under *Diagram consistency*, see Fig. 17) and once selected are visualised graphically over the model (Fig. 17), using red for errors and yellow for warnings. The tabular representation allows filtering and ordering of results depending on the type. Along with the visualisation over the STS-ml model, a textual description is provided for the selected warning or error. In Fig. 17, this analysis has found a warning on a delegation child cycle (highlighted in yellow color), the description better explains it: *There is a delegation cycle created by the delegation of goal "Room selected", which is a subgoal of "Hotel booked", back to "Hotel"*. No errors were found by the consistency analysis for the running scenario. Warnings may be disregarded by the designer, while errors must be solved.

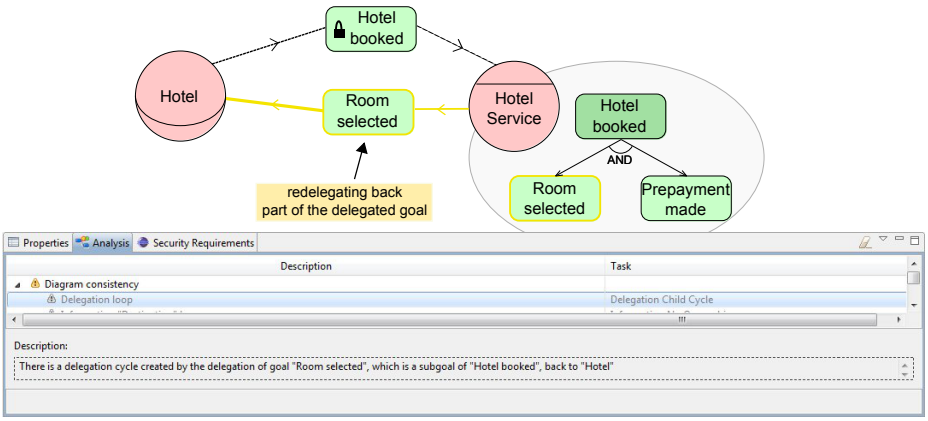


Fig. 17: Results of the consistency analysis

Step 4.2. Security Analysis. For each elicited security need, a security requirement is generated in STS-ml, see **Phase 5.** for more detail. Security analysis is concerned with verifying: (i) if the security requirements specification is consistent—no requirements are potentially conflicting; (ii) if the STS-ml model allows the satisfaction of the specified security requirements. This analysis is implemented in disjunctive Datalog and consists of comparing the possible actor behaviors that the model describes against the behavior mandated by the security requirements. The results are again shown in the Analysis tab visualised on the STS-ml model itself (in red color) when selected, see Fig. 18, which shows a violation of no-redelegation by *Hotel* for goal *Hotel booked*. A textual description provides details on the selected error. In this case, the description states that this is an error because: *"Tourist" has expressed a no-redelegation security need over the delegation of the goal "Hotel booked" to "Hotel", and yet "Hotel" is re-delegating goal "Hotel booked" to "Hotel Service"*.

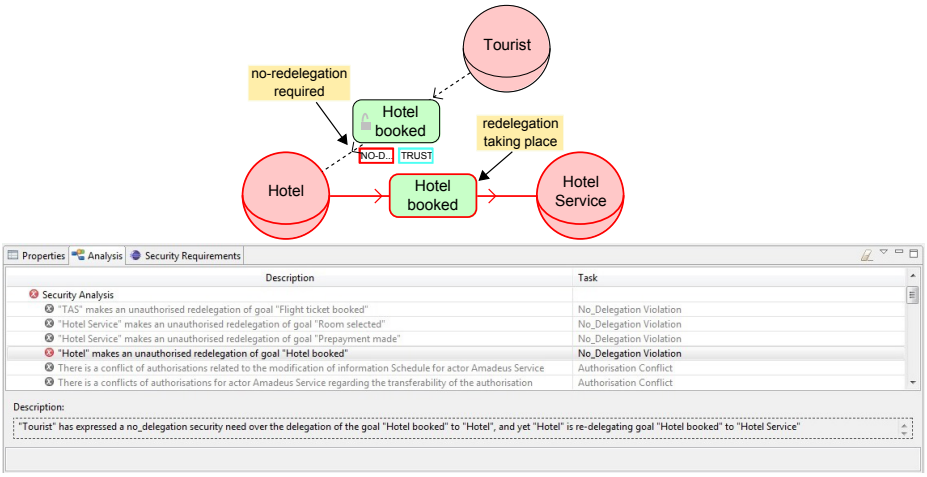


Fig. 18: Results of the security analysis

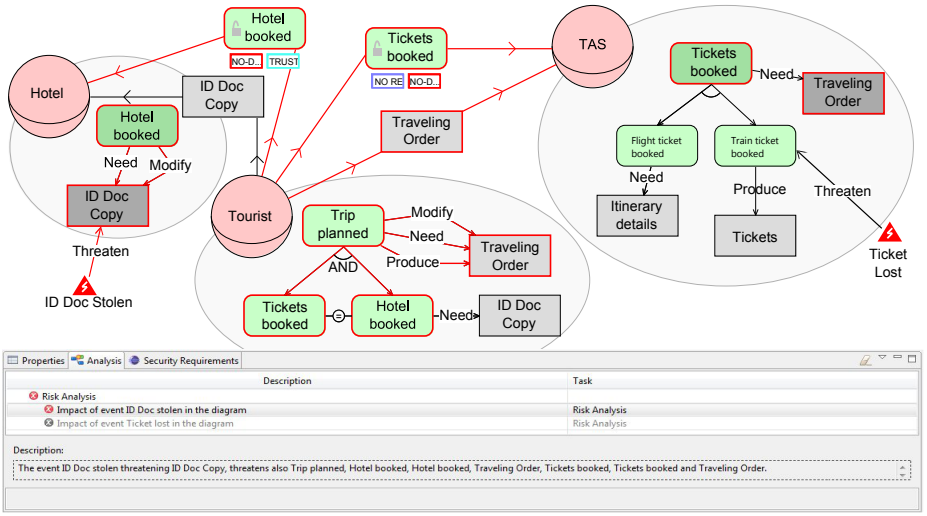


Fig. 19: Results of the threat analysis: threat trace from the event ID Doc Stolen

Step 4.3. Threat Analysis. This analysis calculates the propagation of threatening events over actors' assets. It answers the question: *How does the specification of events threatening actors' assets affect their other assets?'* The results are shown in the Analysis tab, see Fig. 19, which shows the threat trace (highlighted in red color), i.e., the threat propagation for the event *ID Doc Stolen* threatening document *ID Doc Copy* (given this is the one selected). Threatened document affects the goals that need and modify it, as well as the goal of the delegator, should this be a delegated goal. The description for the selected threat propagation states that: *The event "ID Doc Stolen" threatening "ID Doc*

Copy", threatens also "Trip planned", "Hotel booked", "Traveling Order", "Tickets booked", and "Traveling Order".

Phase 5. Derive Security Requirements

Requirements models are useful for communication purposes with the stakeholders. Requirements specifications tell designers what the system has to implement. In STS-ml, security requirements specifications are automatically derived from the constructed requirements model. Security requirements in STS-ml constrain interactions in contractual terms. These contracts are expressed for each required security need, i.e., for each security need expressed from one actor to the other, a security requirement is generated on the opposite direction to express compliance with the required security need.

Responsible	Security Requirement	Requester
TAS	non-repudiation-of-acceptance(delegated (Tourist,TAS,Tickets booked))	Tourist
Tourist	non-repudiation-of-delegation(delegated (Tourist,TAS,Tickets booked))	TAS
TAS	true-redundancy-multiple-actor(Tickets booked)	Tourist
Hotel	no-redelegation(hotel booked)	Tourist
Amadeus Service	integrity-of-transmission(provided(TAS, Amadeus Service, Itinerary details)	TAS
All Agents	not-achieve-both(eticket generated, credit card verified)	Org
Amadeus Service	availability(flight ticket booked, 85%)	TAS
Tourist	delegatedTo(trustworthy(Hotel))	Tourist
TAS	need-to-know(Personal data $\wedge$ Itinerary, Tickets booked)	Tourist
TAS	non-modification(Personal data $\wedge$ Itinerary)	Tourist
TAS	non-production(Personal data $\wedge$ Itinerary)	Tourist
TAS	non-disclosure(Personal data $\wedge$ Itinerary)	Tourist

Table 1: Security Requirements for the running example

The security requirements for the running example are listed in Table 1. For each requirement, we present the *Responsible* actor, the *Security Requirement* itself corresponding to the security need required by the *Requester* actor. We present security requirements showing the responsible actor first, in order to present who are the actors in charge for bringing about or complying with these security requriements. Note that organisational constraints are applicable to "All agents" since they are derived by the organisational rules and regulations, which we denote as *Org*.

Step 5.1. Generate security requirements document. As an output of the fifth phase the method supports the creation of a security requirements document, which contains the list of security requirements derived from the created STS-ml model, as well as information describing the views (information that is customisable by the designers by selecting which concepts or relations they want more information about), and the details of the findings of the automated analyses.

3 STS-Tool: the Case Tool for STS-ml

The STS-Tool is the modelling and analysis support tool for STS-ml. It is an Eclipse Rich Client Platform application written in Java, it is distributed as a compressed archive for multiple platforms (Win 32/64, Mac OS X, Linux). The current version of STS-Tool (v1.3.2) is ready for public use, it is freely available for download from *www.sts-tool.eu*. The website includes extensive documentation including manuals, video tutorials, and lectures. STS-Tool has the following features:

- *Diagrammatic*: the tool enables the creation (drawing) of diagrams. Apart from typical create/modify/save/load operations, the tool also supports:
  - *different views* on a diagram, specifically: *social view*, *information view*, and *authorisation view*. STS-Tool ensures inter-view consistency to facilitate the modelling process.
  - ensuring diagram validity (online): the models are checked for syntactic/ well-formedness validity while being drawn. Examples include enforcing the drawing of relationships over the allowed elements and not others.
  - exporting diagrams to different file formats, such as png, jpg, pdf, svg, etc.
- *Automatic derivation of security requirements*: security requirements are derived from an STS-ml model as relationships between a *requester* (expressing a security need) and a *responsible* actor (in charge of bringing about the security need) for the satisfaction of a *security need*.
- *Automated analysis*: STS-Tool integrates the three automated analyses described in Section 2—consistency, security, and threat analysis. The results of the analyses are enumerated in tabular form below the diagram, and visualised on the diagram itself when selected (see Fig. 1).
- *Generating requirements documents*: the modelling process terminates with the generation of a *security requirements document*: the requirements analyst can customise this document by for instance including only a subset of the actors, concepts or relations, views, etc. The diagrams are described in detail both in textual and tabular form. See [6] for an example.

4 Modelling and reasoning with STS-Tool

We demonstrate the features of STS-Tool by modelling a scenario from a case study on e-Government following the steps of the STS method and using the constructs of the STS-ml language. The demonstration is organised in terms of modelling and analysis exercises to guide the users in building and analysing STS-ml models, and deriving security requirements for the system-to-be.

4.1 Illustrating scenario: Lot searching

The Department of Urban Planning (DoUP) wants to build an application which integrates the existing back-office system with the available commercial services to facilitate the interaction of involved parties when searching for a lot. The *Lot Owner* wants

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

to sell the lot, he defines the lot location and may rely on a Real Estate Agency (*REA*) to sell the lot. *REA* then creates the lot record with all the lot details, and has the responsibility to publish the lot record together with additional legal information arising from the current Legal Framework. *Ministry of Law* publishes the accompanying law on building terms for the lot. The *Interested Party* is searching for a lot and: (i) accesses the DoUP application to invoke services offered by the various REAs; (ii) defines a trustworthiness requirement to allow only trusted REAs to contact him; (iii) sets a criteria to search and select a *Solicitor* and a *Civil Engineer* (CE) to asses the conditions of the lot; (iv) assigns solicitor and CE to act on his behalf so that the lot information is available for evaluation; and (v) populates the lot selection for the chosen CE and Solicitor. *Aggregated REA* defines the list of trusted sources to be used to search candidate lots, it collects candidate lots from trusted sources, and ranks them to visualize to the user. *The Chambers* provide the list of creditable professionals (CE and Solicitors).

4.2 Modelling activities

We present here the steps the designer will follow in order to perform the modelling steps, depicted in see Fig. 2, Section 2.2, while illustrating them building the models for the Lot searching scenario. This activity, apart from guiding the designer through the modelling phases, shows how the tool facilitates and supports the modelling process.

Phase 1. Modelling the Social view

We start the tool (Fig. 20) to begin with the modelling activities, for which we can use the concepts and relations from the Palette.

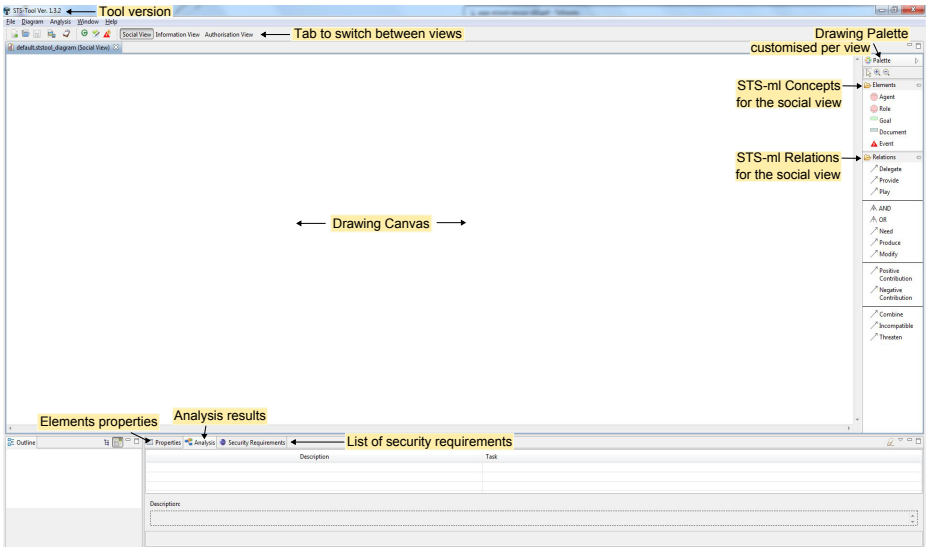


Fig. 20: The STS-Tool

Exercise 1. Identify stakeholders. *Which are the stakeholders we can identify from the Lot Searching scenario? How can they be represented in terms of roles and agents? Explain why. Draw identified roles and agents using STS-Tool. Use properties to better describe them.*

Solution: This corresponds to *step 1.1* of STS method. Make sure you are on the Social View, selecting it as shown in Fig. 21.

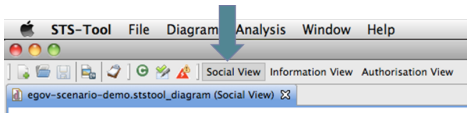


Fig. 21: STS-Tool screenshot: selecting the social view

The identified roles and agents are:

- *Roles*: Lot owner, REA, Map Service Provider, Interested Party, Solicitor, CE Chambers, and Solicitor Chambers
- *Agents*: DoUP Application, Aggregated REA, and Ministry of Law

The reason for this is that *roles* refer to general actors that are instantiated at run time, while *agents* refer to concrete entities already known at design time. So we do not know who Lot owner or Interested Party is going to be, but there is only one Aggregated REA and one Ministry of Law in this scenario, so we know them already at design time.

We draw the identified roles and agents as shown in Fig. 22, and use the properties tab to better describe these roles and agents. In this case, we provide a description for the role *Lot owner*. This feature is helpful because the tool sets a limit of 25 characters on concept names, allowing longer descriptions to be inserted in the properties tab.

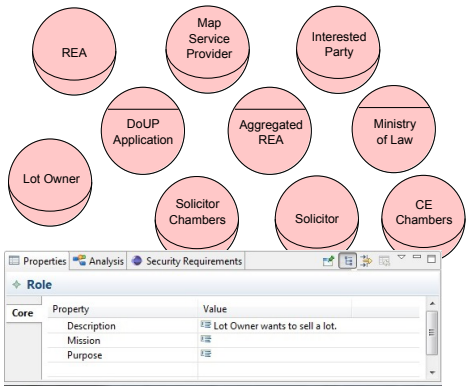


Fig. 22: Stakeholders for the Lot searching scenario

Exercise 2. Building actor models. *For each modelled actor (role and agent) identify its assets. What are the goals they have and how they are achieved? What are the documents actors have and manipulate (use, modify, produce)?*

Solution: This corresponds to the first part of *step 1.2* of STS method, identifying actors' assets. We start with role Lot Owner, whose actor model is shown in Fig. 23.

The Lot Owner wants to sell a lot, therefore his main goal is to have lot sold. He could sell the lot either privately or through an agency. In the Lot searching scenario, the lot owner interacts with a real estate agency (REA), hence we can further refine how this is achieved. To sell the lot through an agency: a lot record should be created, lot information needs to be provided, the lot location needs to be defined and finally the lot price needs to be approved. This is represented through the and-decomposition of goal *lot sold via agency* into the above enumerated goals. In order to create the lot record, the owners personal data are needed (goal *lot record created* needs document *owner personal info*). In order to provide lot info, details about the lot are needed (goal *lot info provided* needs document *lot info*). The tool helps the designer by allowing this relation to be drawn only starting from the goal to the document, not vice-versa.

The same modelling is performed for the other identified actors, we will provide more details for some of them in the following.

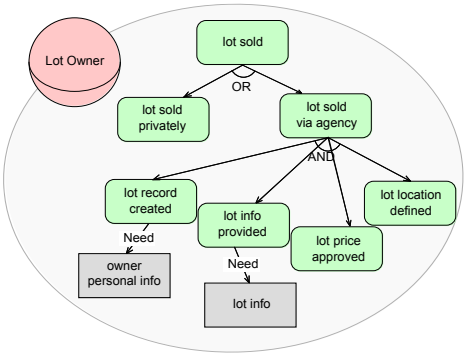


Fig. 23: Actor models: Lot owner

When first created, roles and agents come together with their rationale (open compartment), so that we can specify the goals or documents (assets) they have. The rationales can be hidden or expanded, to give the designer the possibility to focus on some role or agent at a time. We place actor goals within their rationale. STS-Tool facilitates a correct modelling of goal trees, by not allowing goal cycles. Several checks are performed live by the tool for this purpose, such as not permitting the designer to draw a decomposition link from a subgoal to a higher level goal in a goal tree.

Exercise 3. Identifying actors' interactions. *For each actor identify: (i) the goals for which he needs to rely on others (goal delegations); (ii) the documents which he needs to get from (provide to) others (document provisions).*

Solution: This corresponds to the second part of *step 1.2* of STS method, identifying actors' interactions. We start with Lot Owner's interactions. To have the lot record published Lot Owner delegates goal *lot record created* to REA, see Fig. 24.

Note that, when drawing a delegation, the tool makes sure that the actor does have a goal it wants to pursue, before allowing the designer to draw the goal delegation relationship starting from the given actor. It is worth emphasising that STS-ml allows only the delegation of leaf goals, delegation of upper level goals is forbidden, and STS-Tool support this. If a leaf goal is delegated and the designer decides to further decompose

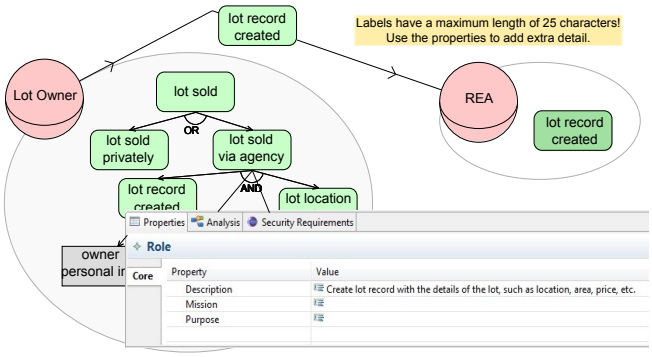


Fig. 24: Actor interactions: lot owner

this goal within the delegator, the tool will prompt him with a message and not allow the further decomposition. Once the goal delegation relationship is drawn, the delegated goal is automatically created within the compartment of the delegatee. This goal is represented in a darker color than the original goal, in order to clearly distinguish for each actor its own goals from the goals delegated to it. Additionally, the tool does not allow the goal to be deleted from the delegatee's compartment unless the delegation is deleted. Importantly, delegation cycles are not permitted by the tool.

STS-ml models are built iteratively, so now we will iteratively build actors' models. For this, the designer should think of: *How can the delegatee achieve the delegated goal?* Answering this question allows one to find out more details on the interacting actors: goal and/or-decompositions, document, goal-document relationships, document provisions and re-delegations if applicable. We continue the solution of ***Exercise 2*** by building the actor model for REA, see Fig. 25.

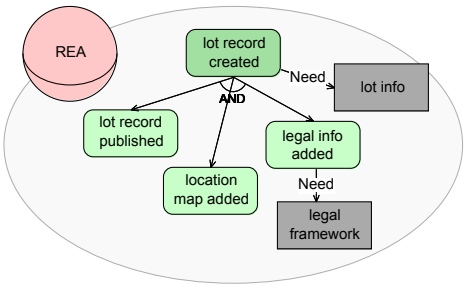


Fig. 25: Actor models: REA

Exercise 4. Expressing security needs. *Analyze goal delegations and identify any applicable security needs.*

Solution: This solution follows *step 1.3.* of the STS method. We have represented two actors so far and the interactions among them, so at this step we consider security needs applicable to this interaction, analysing the drawn goal delegations and document provisions. We focus on the identified goal delegation, and consider which of the

supported security needs (Non-repudiation, Redundancy, No-redelegation, Trustworthiness, Availability) applies to it.

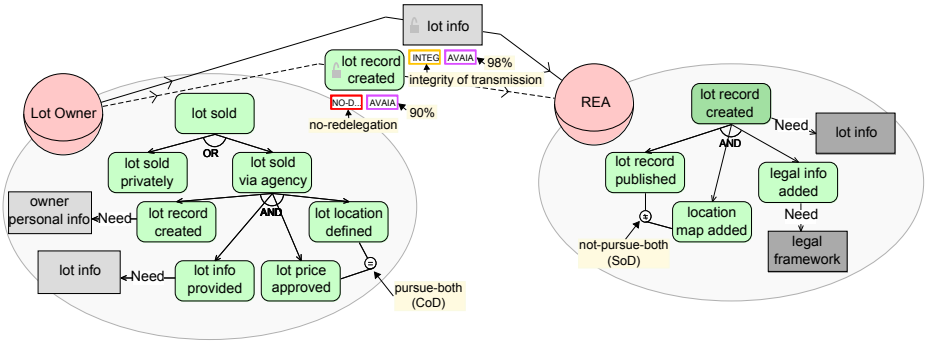


Fig. 26: Expressing security needs: REA

In Fig. 26 *Lot Owner* requires the *Real Estate Agency* no-redelegation of the goal *lot record created*, and an availability level of 85% for the same goal. To specify this using the tool, the designer needs to right-click on the delegated goal, to have a drop down list of security needs and select the desired ones. Graphically security needs can be specified by right-clicking on the goal or document and selecting the desired security needs from a given list. The selection of at least one security need, shows a padlock on the goal or document (see Fig. 26). The selected security need can be shown explicitly by clicking on the padlock, which shows small boxes below the delegated goal or provided document; each box has a different colour and different label (see Fig. 26).

Iteratively building actor models. What about other actors? This is another iteration of *step 1.2.* in identifying actors' assets. We consider now the Interested Party and build its actor model as shown in Fig. 27.

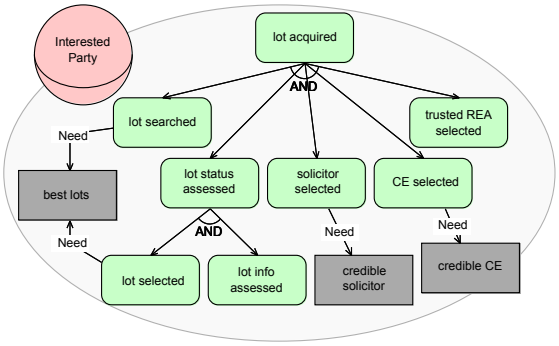


Fig. 27: Actor models: Interested Party

Iteratively identifying actors' interactions. Identify goal delegations and document provisions (iteration of *step 1.2.*) the *Interested Party* relies upon. In addition to goal delegations, we need to consider document provisions. The *Interested Party* needs *best lots* to perform a search over lots and find the best one, the document is provided by *DoUP Application*, see Fig. 28.

Iteratively expressing security needs. Considering *Interested Party* and his interactions, we determine which are the applicable security needs (iteration of *step 1.3.*). See Fig. 28 for the details of the actor model for *Interested Party*, its interactions, and the security needs expressed over them. For instance, *Interested Party* expresses a trustworthiness security need over the delegation of goal *trusted REA selected* to *DoUP Application*. Additionally, a combination of duties is specified between the goals *solicitor selected* and *CE selected*, among others.

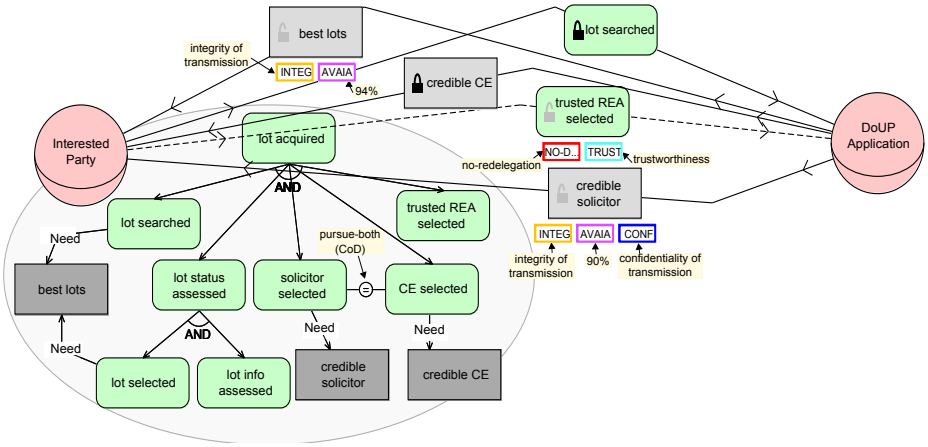


Fig. 28: Actor model and expressing security needs for Interested Party

Exercise 5. Modelling threats. *Which actors' goals and documents are threatened? Represent threats over goals and over documents. Use properties to describe the threats.*

Solution: This corresponds to *step 1.4.* of the STS method. Fig. 29 represents the events identified to threaten actors' assets. For instance, event *file stolen* threatens document *credible CE* of *CE Chambers*.

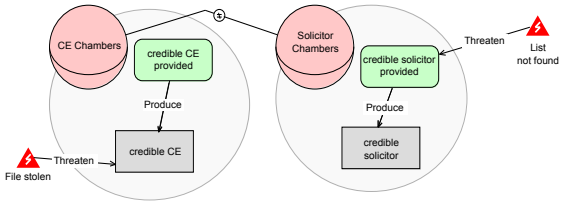


Fig. 29: Modelling threats

Phase 1. Summary. As it can be inferred by the above exercises (1–4), modelling an STS-ml model is an iterative process, *step 1.2.* and *step 1.3.* are repeated till all actor models are built and all security needs are captured. Termination criteria is defined by answering these questions: *Are there any remaining actors? Who are they? Have we captured all their interactions? What about security needs?* If there are still actors to be represented, then answer: *How can they achieve their goals by themselves? What documents do they manipulate? What operations do they need to perform over these*

documents? Do they posses the said documents? Do they need to rely on other actors for some goals? Are there any events threatening their assets?

Exercise 6. Completing the Social View. *Complete the modelling of the social view for the Land searching scenario. As identified in the beginning of this phase, the remaining actors are: DoUP application, Aggregated REA, Ministry of Law, The Chambers (Solicitors' Chambers and CE Chambers), and Solicitor. For the remaining actors: (i) draw their models, (ii) draw their interactions (goal delegations and document provisions), (iii) express security needs over their interactions, and (iv) represent threats over their goals and documents.*

Phase 2. Modelling the Information View

We switch to *Information View* (see Fig. 30) and *represent information* entities relating them together with the *documents* within which they are contained. The tool inherits the roles and agents together with their documents from the social view, so the designer is left with the modelling of the information entities, to then relate them to documents via TangibleBy relationships.

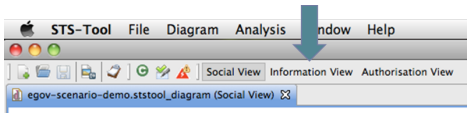


Fig. 30: STS-Tool screenshot: switching to the information view

Exercise 7. Identify information and owners *What is the informational content of the documents represented in the social view? Who are the owners of this information? What is the structure of information? Is there a structure of documents?*

Solution: This corresponds to *steps 2.1* and *2.2* of the STS method. We first need to identify information entities and relate them with documents. We determine who is the owner of the identified information. For instance, *Lot Owner* provides information about the lot, we identify information *lot info details*, which is owned by the *Lot Owner* himself and is represented (made tangible) by document *lot info* (see Fig. 31). STS-Tool allows the relation owns to be drawn starting from the role or agent towards the information it owns, and the relationship Tangible By to be drawn only starting from information to documents.

Then, we model the information hierarchy (relate information with information). In Fig. 31, information *lot geo location* is part of information *lot info details*. Finally, we model the document hierarchy (relate documents with documents). Documents *trusted REA* and *best lots* are part of document *trusted sources*, see Fig. 31.

The tool helps the designer in building this structure by allowing the Part Of relations to be drawn only between information or documents respectively. Additionally, cycles of Part Of are not allowed by the tool.

Iteratively building the Information View. Similarly to the previous identified information, the designer considers for each document what is its informational content. An information could be made tangible by more documents, as well as the same document can make tangible more information pieces. The designer continues modelling until all the relevant information entities have been represented.

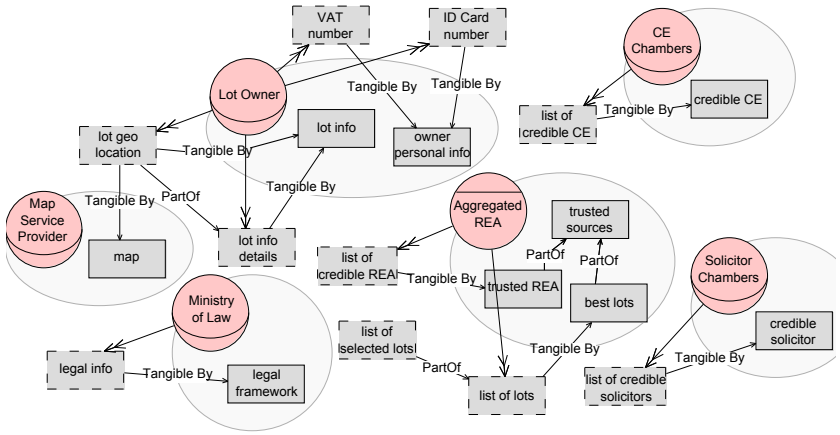


Fig. 31: Part of information view for the Land searching scenario

Phase 2. Summary. Steps 2.1. and 2.2. are repeated till all information entities have been modelled, their owners have been identified, they have been related to their corresponding documents and information or document structure is determined.

Exercise 8. Completing the Information View. *Complete the modelling of the information view for the Land searching scenario.*

Phase 2. Modelling the Authorisation View

We switch to the authorisation view. Starting from information owners, we draw the authorisations they grant to other actors.

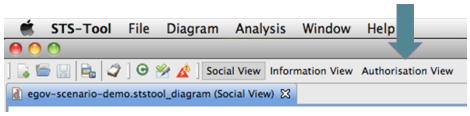


Fig. 32: STS-Tool screenshot: switching to the authorisation view

Exercise 9. Modelling transfer of authorisations. *Are there any authorisations granted from the information owners? Is authority to transfer authorisations granted? Which are the information for which authorisation is granted? Are there any limitations of authority?*

Solution: This corresponds to *step 3.1.* of the STS method. Starting from information owners or authorised parties, we identify authorisation relationships between actors. Fig. 33 depicts the authorisations granted by information owners. For instance, the *Lot Owner* authorises *REA* to use, produce, and distribute information *lot info details* and *lot geo location* in the scope of goal *lot record created*, granting a transferable authorisation.

Fig. 33 shows the authorisations granted by authorised parties. For instance, the *Solicitor* authorises the *DoUP Application* to use and distribute the information *legal info* for goal *citizens helped* granting a transferable authorisation.

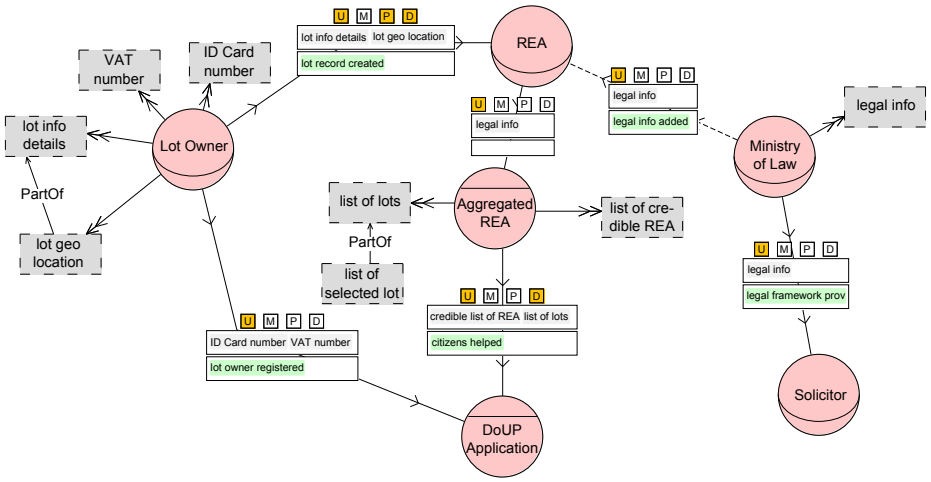


Fig. 33: Authorisation view: authorisations from information owners

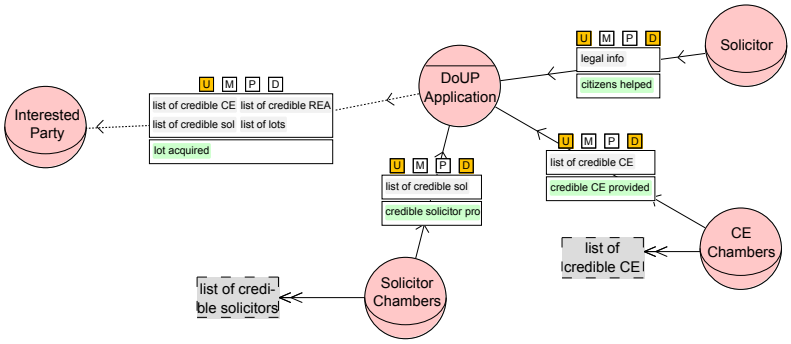


Fig. 34: Authorisation view: authorisations from authorised parties

Implicitly express security needs. Security needs over authorisations are captured implicitly. For instance, from the authorisation relationship from the *Lot Owner* to the *Real Estate Agency* we can derive that a security need for non-modification of information *lot info details* and *lot geo location* is expressed, as well as a need-to-know security need of using and producing this information for goal *lot record created*.

Iteratively building the Authorisation View. For all the represented actors, consider whether there are permissions being granted to them or whether they grant any permissions to the interacting actors.

Phase 3. Summary. *Step 3.1* is repeated till all authorisation relationships have been drawn. Termination criteria is established depending on whether all authorisations have been captured and all the correct authorisation security needs have been expressed.

Exercise 10. Completing the Authorisation View. *Complete the modelling of the authorisation view for the Land searching scenario.*

Modelling Activities Summary. The modelling process (Phases 1—3) is iterative. The views can be further refined, depending on the level of detail that is needed. The

changes in one view have effects on other views. As described above, the different roles or agents are maintained throughout the views, so the addition or deletion of some role or agent would affect the other views. However, even in these cases, the tool provides support by checking that a role or agent is deleted only when it does not have any interactions with other roles (agents). Termination criteria is established by answering these questions: Did I capture all important interconnections? Did I express all the security needs? For a more comprehensive model, use properties to better describe its elements.

4.3 Running Automated Analysis

After constructing the STS-ml model for the Lot searching scenario, we can run the automated analyses to verify its consistency, the satisfaction (possible violation) of security needs, and the threat propagation over actors' assets.

Phase 4. Automated Analysis

Exercise 11. Consistency Analysis. *Is the STS-ml model for the Lot searching scenario well-formed?*

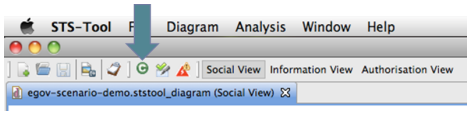


Fig. 35: Executing consistency analysis

Solution: This corresponds to *step 4.1.* of the STS method. The consistency analysis can be executed in the tool by using the Consistency Analysis tab, see Fig. 35. The consistency analysis for the lot searching scenario did not find any warnings or errors.

Exercise 12. Security Analysis. *Is it possible in the model that a security requirement is violated? Identify violations.*

Solution: This corresponds to *step 4.2.* of the STS method. The security analysis can be executed in the tool by using the Security Analysis tab, see Fig. 36. Whenever the designer runs the security analysis the tool automatically call the consistency analysis, in order to ensure that the constructed model is well-formed. Only when the consistency analysis finds no errors, the tool continues with the execution of the security analysis.

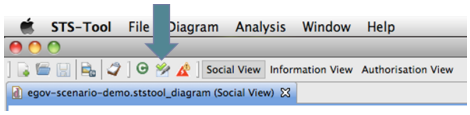


Fig. 36: Executing security analysis

The results of the analysis are shown in the Analysis tab, and once selected are visualised graphically over the model (Fig. 37).

In our example, the security analysis found several violations of the specified security needs (errors), such as for instance the violation of non-production by the Map

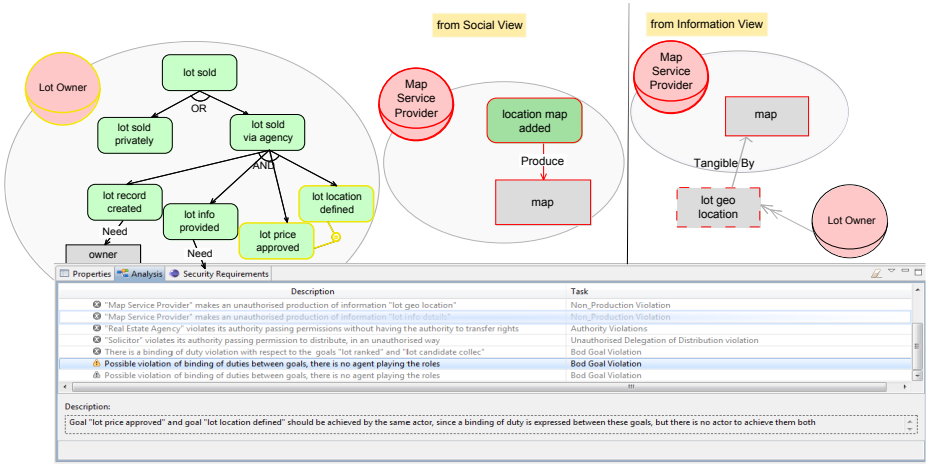


Fig. 37: Executing security analysis: visualisation of results

Service Provider. As it can be seen by the diagram in Fig. 33 and 34 showing authorisation relations, there is no authorisation relationship towards *Map Service Provider*, but *Map Service Provider* can produce *lot geo location* since there is a produce relationship from its goal *location map added* towards document *map* representing (making tangible) information *lot geo location*, information that is owned by *Lot Owner*.

Similarly, there is a possible violation of a combination (binding) of duties between the goals *lot price approved* and *lot location defined* of *Lot Owner*, as there appears to be no single actor achieving both these goals, see Fig. 37 showing this warning.

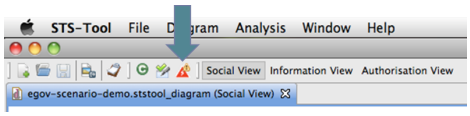


Fig. 38: Executing threat analysis

Exercise 13. Threat Analysis. *What are the effects of the events threatening actors' assets? Identify the threat propagation.*

Solution: This corresponds to *step 4.3.* of the STS method. Go to the Threat Analysis tab, Fig. 39. The results of the threat analysis are shown in the Analysis tab, and once selected are visualised graphically over the model (Fig. 39), which shows the impact of the event *List not found* threatening goal *credible solicitor provided*.

4.4 Deriving security requirements

Requirements specifications define what the system has to implement.

Phase 5. Derive Security Requirements

In STS-ml, security requirements specifications are automatically derived from requirements models.

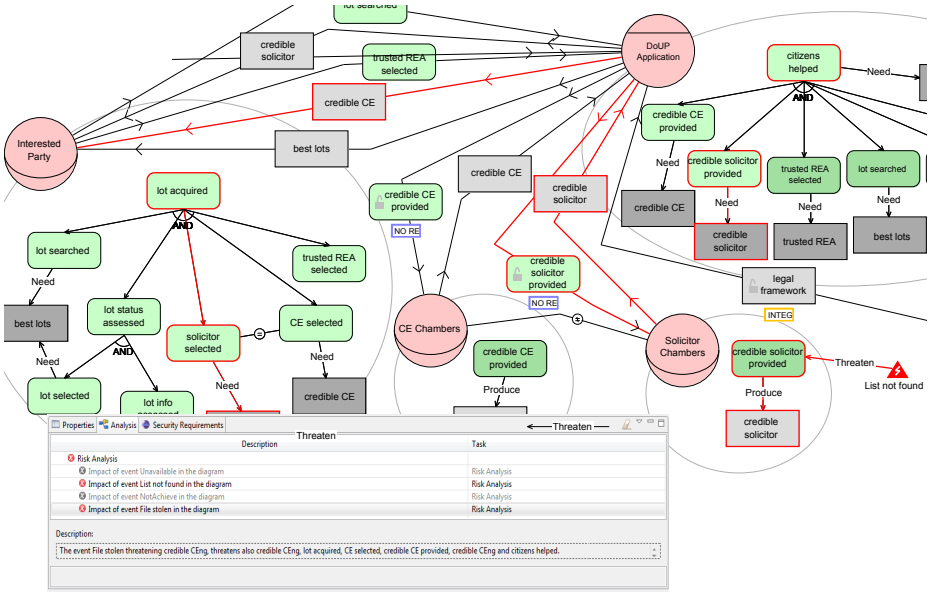


Fig. 39: Executing threat analysis

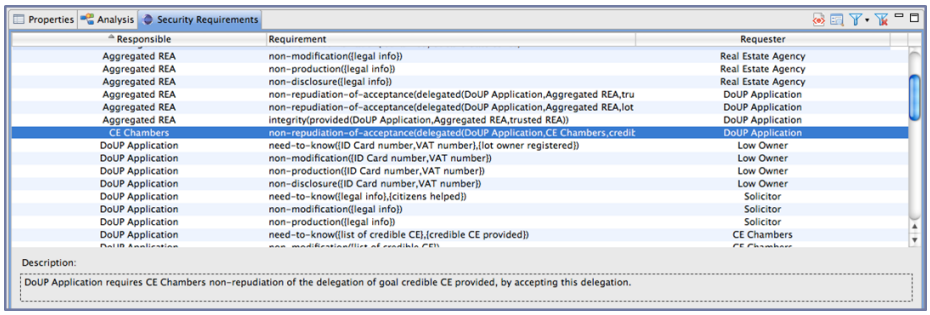


Fig. 40: Security requirements for the Lot searching scenario

Exercise 14. Generate security requirements document. *What are the security requirements for non-modification in the Lot searching scenario? What about security requirements for non-disclosure of legal information?*

Solution: This corresponds to *step 5.1* of the STS method. Security requirements are automatically generated in STS-Tool. To identify security requirements for non-modification, we consider the security requirements for the Lot searching scenario (see Fig. 40) and order them with respect to the requirement, so to group together requirements on non-modification. Similarly, we identify security requirements on non-disclosure, and identify only one on *legal information*, to be satisfied by *Aggregated REA* (no violation was identified by the security analysis).

Exercise 15. Generate security requirements document. *Generate the document for the Lot searching scenario.*

Solution: We generate the security requirements document for the Lot searching scenario by using the tab as shown in Fig. 41.

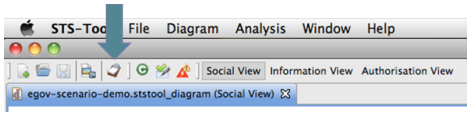
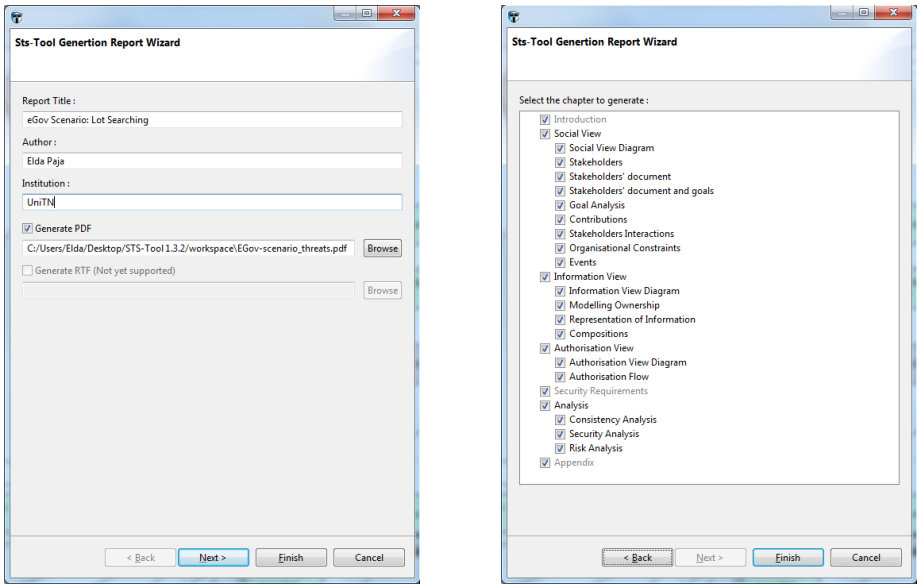


Fig. 41: STS-Tool screenshot: selecting the social view

This opens up a sequence of dialogue windows to determine the title and author for the document (Fig. 42a), and deciding what to include, which views, elements within the views, or analysis results (see Fig. 42b).



(a) (b)

Fig. 42: Customising the security requirements document

5 Conclusions

We have presented how STS-Tool—the case tool for STS-ml—allows an effective security requirements engineering process, following the steps of the STS method. STS-Tool supports modelling and reasoning activities, while aiding the requirements analyst and the security engineer at each step.

We have demonstrated how modelling and reasoning activities can be performed with the help of a case study from eGovernment. STS-ml supports a rich set of security requirements that are derived from security needs expressed over actors' interactions.

Many security requirements engineering frameworks and methods are available in the literature. Secure Tropos [6] models security concerns throughout the whole development process. It expresses security requirements as *security constraints*, considers potential threats and attacks, and provides methodological steps to validate these requirements and overcome vulnerabilities.

Liu et al. [5] extend *i** to deal with security and privacy requirements. Their methodology defines security and privacy-specific analysis mechanisms to identify potential attackers, derive threats and vulnerabilities, thereby suggesting countermeasures. Their solution falls short when considering security issues through the later phases of the development process [6].

SI* [4] is a security requirements engineering framework that builds on *i** [10] by adding security-related concepts, including delegation and trust of execution or permission. SI* uses automated reasoning to check security properties of a model, reasoning on the interplay between execution and permission of trust and delegation relationships. Our framework supports a more expressive ontology (featuring sophisticated authorisations) to represent information security requirements, and clearly decouples business policies (the goals of an individual actor) from security requirements.

Our future work includes improving the usability of STS-Tool for a better user experience, as well as extending its reasoning capabilities for more sophisticated reasoning.

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

References

1. E. Bertino, S. Jajodia, and P. Samarati. A flexible authorization mechanism for relational data management systems. *ACM Transactions on Information Systems*, 17(2):101–140, 1999.
2. F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Adaptive Socio-Technical Systems: a Requirements-driven Approach. *Requirements Engineering*, 18(1):1–24, 2013.
3. F. Dalpiaz, E. Paja, and P. Giorgini. Security requirements engineering via commitments. In *Proceedings of STAST'11*, pages 1–8, 2011.
4. P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling security requirements through ownership, permission and delegation. In *Proc. of RE'05*, pages 167–176, 2005.
5. L. Liu, E. Yu, and J. Mylopoulos. Security and privacy requirements analysis within a social setting. In *Proc. of RE 2003*, pages 151–161, 2003.
6. H. Mouratidis and P. Giorgini. Secure Tropos: A security-oriented extension of the tropos methodology. *IJSEKE*, 17(2):285–309, 2007.
7. Elda Paja, Fabiano Dalpiaz, and Paolo Giorgini. Managing security requirements conflicts in socio-technical systems. In *Proceedings of the 32nd International Conference on Conceptual Modeling*, volume 8217 of *LNCS*, pages 270–283, 2013.
8. M. P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7(1):97–113, 1999.
9. M. E. Whitman and H. J. Mattord. *Principles of Information Security*. Course Technology Press, 4th edition, 2011.
10. E. Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, University of Toronto, Canada, 1996.