The Socio-Technical Security Requirements Modelling Language for Secure Composite Services

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

[1] University of Trento – DISI, Via Sommarive 5, 38123, Povo, Trento, Italy – {elda.paja, paolo.giorgini}@unitn.it
[2] Utrecht University – Department of Information and Computing Sciences, Princetonplein 5, De Uithof, 3584 CC Utrecht, The Netherlands – f.dalpiaz@uu.nl

Abstract. Composite services foster reuse and efficiency in providing consumers with different functionalities (services). However, security aspects are a major concern, considering that both service consumers and providers are autonomous and heterogeneous—thus, loosely controllable entities. When consumers provide information in order to be furnished some service, what happens to that information? Do service consumers trust service providers? In order to tackle the design of secure and trustworthy composite services, we should consider the security requirements such a composition must satisfy. We propose STS-ml, a security requirements modelling language that allows modelling security requirements over participants' (consumers and providers) interactions. These security requirements are expressed in terms of social contracts the various parties shall comply with while interacting (consuming/furnishing some service). Most importantly, STS-ml considers social and organisational threats that might affect the said composite services. In this chapter, we give an overview of STS-ml, introducing its modelling and reasoning capabilities while building models from the Aniketos eGovernment case study and verifying that the composite service complies with the specification, as well as checking whether a recomposition is needed.

1 Introduction

The Future Internet aims at digitalising many aspects of our lives, in particular offering online a great deal of services we are used since ages to have/obtain on the basis of face-to-face communication and interaction. This new system surpasses geographical limitations and confines, for it allows a wide range of organisations and individuals to offer (provide) a plethora of services to a wide variety of users (consumers). Everyone is free to join or leave this system and be in any of the roles, provider or consumer. Thus, the system exists because of the interaction among participants, a black-box interaction among autonomous actors based on service interfaces.

As much as this new environment facilitates interaction and communication, often increasing the quality of services (because of competition—now with all providers of the same types of services), it opens up many new challenges and issues with respect to trust, security and privacy. Typically consumers need to provide or exchange information with service providers to be able to access and use the offered services. But, in many cases consumers have little or no information about the providers, so can they really trust them with their information? Can the consumers trust providers not to misuse their information? Can consumers trust providers not to disclose their data to unauthorised parties? These are some of the questions we need to consider in engineering a system that offers the desired services while respecting users' needs on protecting their information (maintaining user trust and an adequate level of security).

The analysis of security aspects is of utmost importance, since information is disclosed (and tasks are executed) beyond the "safe" boundaries of a single organisation, and due to the autonomy of the participants, users have no control whatsoever over providers (with respect to what might happen to their information). The lack of control makes the design of composite services a challenging task. Such design should not consider technical details alone, but the bigger picture comprising the participants interacting to get and provide services, which stand at the basis of service-oriented applications. We need a way to capture and represent participants' needs when interacting with others, either to consume a service or to exchange information, in order to understand what are their concerns with respect to security over the said interaction. Moreover, in such a dynamic environment, the participants may be subject to security threats affecting their important assets. These threats are not necessarily technical, rather they are *social*, as they originate from the interactions between social actors (humans and organisations).

As in any engineering discipline, early awareness and analysis of potential problems is beneficial to system design, resulting in the development of more robust systems. For this, we have proposed a security requirements engineering modelling language that supports the specification of security requirements for service-oriented applications. The modelling language, STS-ml (Socio-Technical Security modelling language), allows to represent service-oriented settings in terms of goal-oriented actors that interact with one another to obtain (consume) services and exchange information. The key idea is to relate security requirements to *interaction*. This means adding constraints to the way actors exchange information, and to the delegation of goals (services). These constraints help specify the security requirements the actors shall comply with while interacting. STS-ml specifies security requirements as *social commitments* [6], promises with contractual validity made by an actor to another. One actor commits to another that, while delivering some service, it will comply with the required security needs. For example, a service provider might need to respect (commit for) the non-disclosure of the consumer's personal data, which is required as input to the provided service. Similarly, the same service provider may commit not to redelegate its offered services to other actors (providers), which might be not trusted by the consumer. As any approach to security, for a thorough security analysis, STS-ml does not overlook threats, and considers social and organisational threats that might affect the well-functioning of the systems. In our case, threats might affect the services under operation and the threats' impact might require a service recomposition (considering alternative services already known at design time) so that the required functionalities are maintained for the consumers.

Security requirements are integrated within the service interface, so that the provider makes a commitment to prospective service consumers to satisfy the given security needs. In this way, security requirements can be effectively used to specify the service under development. Expressing security requirements within service interfaces ensures that the security needs expressed by the consumers result in actual commitments the provider makes to the consumer to satisfy the imposed constraints (the security needs) while delivering the service. For instance, if consumers are concerned with the disclosure of personal data, the service interface may declare that the data will not be disclosed to other actors. Irrespective of the service implementation, such interface makes the provider committed for non-disclosure.

These specifications guide the design of composite services that satisfy the security requirements. However, in certain cases, the specification may be *inconsistent*, i.e., one or more requirements might be conflicting. If not effectively managed, inconsistencies result in the implementation of a system that violates one or more requirements. We propose to rely on automated reasoning to identify and resolve these conflicts. This choice is justified by our gathered evidence [8] that requirements models are large and that even skilled analysts would be unable to identify all the conflicts in a model.

Analysis results over security requirements are intended to improve the created models, so that the final security requirements specification is consistent and can serve as a basis for the implementation of the considered composite services. An analysis of threats impact, on the other hand, determines whether a service recomposition is required or not, should any of the composing services be threatened (become unavailable).

2 STS-ml: an Overview

STS-ml has been first proposed in [2], here we present the current version of STS-ml. STS-ml includes high-level organisational concepts such as actor, goal, delegation, etc. Security requirements in STS-ml models are mapped to *social commitments* [6]—contracts among actors—that actors (participants) shall comply with at runtime. STS-ml modelling consists of three complementary views of the same model, namely *social*, *information*, and *authorisation view* (see Fig. 1, 3, and 4), so that different interactions among actors can be analysed by concentrating on a specific view at a time. Inter-view consistency is ensured by STS-Tool [3] (see Chapter 7).

We consider a scenario from the eGovernment case study (Chapter 15) as a running example to illustrate STS-ml.

Example 1 (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 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

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

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).

2.1 Multi-view Modelling

STS-ml relies on multiple views of the same model, each representing a specific perspective on the analysed setting. Multi-view modelling promotes modularity and separation of concerns. Currently, STS-ml includes three views: *social view*, *information view*, and *authorisation view*.

Social View. The *social view* (see Fig. 1)—a variant of *i** [9]–based modelling languages, such as SI* [5]—, represents participants of a socio-technical system as *intentional* and *social* actors. These actors are intentional for they enter the system in order to fulfil their objectives (goals), and they are social, for they interact with others to fulfill their objectives (by *delegating goals*) and obtain information (by *exchanging/transmitting documents*). STS-ml supports two types of actors: agents—representing concrete participants, and roles—abstract actors, used when the actual participant is unknown. In our example, (Example 1), the identified roles are *Lot owner*, *REA*, *Map Service Provider*, *Interested Party*, *Solicitor*, *CE Chambers*, and *Solicitor Chambers*, while the represented agents are: *DoUP Application*, *Aggregated REA*, and *Ministry of Law*, see Fig 1. 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. That is, we do not know who *Lot owner* or *Interested Party* is going to be, but we consider that there is only one *Aggregated REA* and one *Ministry of Law* in this scenario, which are known already at design time.

Actors may achieve their goals on their own by decomposing (further refining goals) via: (i) and-decompositions: all subgoals must be achieved for the goal to be achieved; and (ii) or-decompositions: at least one subgoal must be achieved for the goal to be achieved. For instance, in Fig 1, *Lot Owner* has goal *lot sold*. He could sell the lot either privately or through an agency. Therefore, *Lot Owner* or-decomposes *lot sold* into *lot sold privately* and *lot sold via agency*. In the Lot searching scenario, we consider that the *Lot Owner* interacts with a real estate agency (REA), hence we 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. Thus, this is represented through the and-decomposition of goal *lot sold via agency* into goals *lot record created*, *lot info provided*, *lot price approved*, and *lot location defined*.

Actors can delegate goals when they cannot achieve them on their own or it is more convienient to rely on others. Note that delegation is possible if the delegator actor has the said goal [4]. For instance, in Fig 1, *Lot Owner* wants to have the lot sold via agency, for which he delegates goal *lot record created* to the *Real Estate Agency*.

[4] Note that in STS-ml only leaf goals can be delegated!

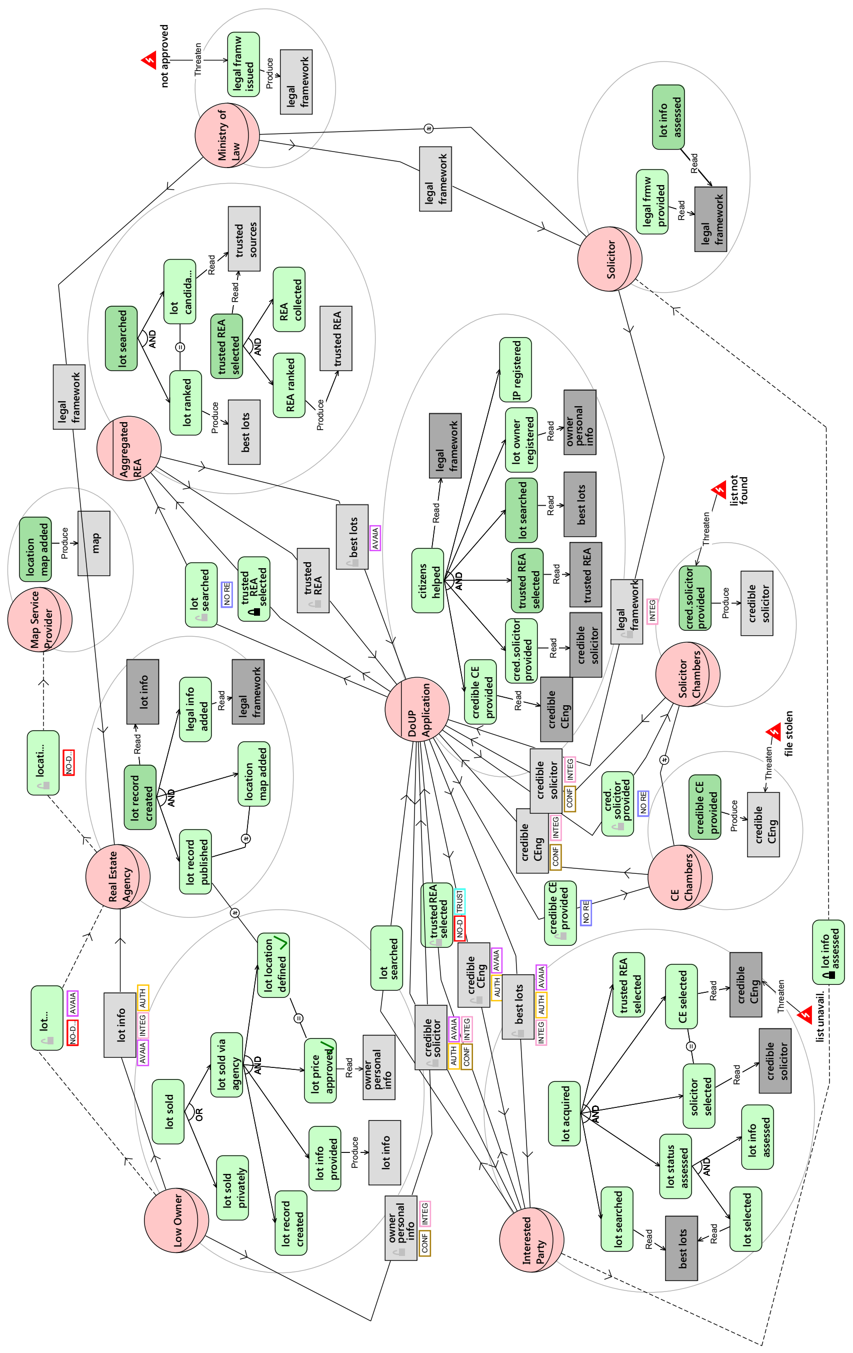


Fig. 1: Lot Searching scenario—Social View

Actors may possess documents (containing information); they may read, modify, or produce documents while achieving their goals. For instance, in Fig 1, *Real Estate Agency* reads *lot info* to achieve goal *lot record created* (the owners personal data are needed to create the *lot record*). This document (*lot info*) is produced by the *Lot Owners* while providing lot information (goal *lot info provided*). Actors can transmit documents to others only if they possess the required document. For instance, in Fig 1, *Lot Owner* is the creator of *lot info* (i.e., possesses the document) and he transmits this document to *Real Estate Agency*.

Modelling threats. In STS-ml we represent events threatening stakeholders' assets—*informational assets* and *intentional assets*. However, given that in the social view stakeholders exchange and manipulate information via documents, we model threats over actors' *documents* and *goals* respectively. STS-ml proposes the concept event and the relationship threaten relating the event to the asset it threatens. For instance, in Fig. 1, we represent the events identified to threaten actors' assets for Example 1. For instance, event *file stolen* threatens document *credible CE* of *CE Chambers* (see Fig. 2 which zooms over Fig. 1).

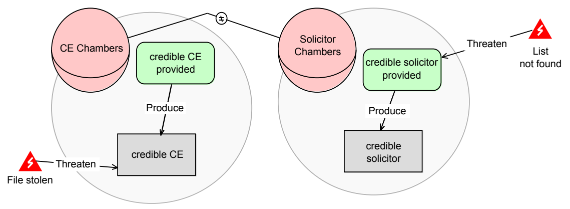


Fig. 2: Modelling threats

Information View. This view (Fig. 3) shows how information and documents are interconnected to identify which information actors manipulate, when they read, modify, produce, or transmit documents to achieve their goals in the social view. Information can be represented by one or more documents (through Tangible By), and on the other hand one or more information entities can be made tangible of the same document.

Importantly, this view relates information to their owners through own relationships. For instance, *Lot Owner* provides information about the lot, and thus 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. 3).

Information view gives a structured representation of actors' information and documents via part of relationships. These relationships can result in information hierarchies (relating information with information) or document hierarchy (relating documents with documents). This means that such relationship holds only between entities of the same type, either information or documents. For instance, in Fig. 3, information *lot geo location* is part of information *lot info details*, while documents *trusted REA* and *best lots* are part of document *trusted sources*.

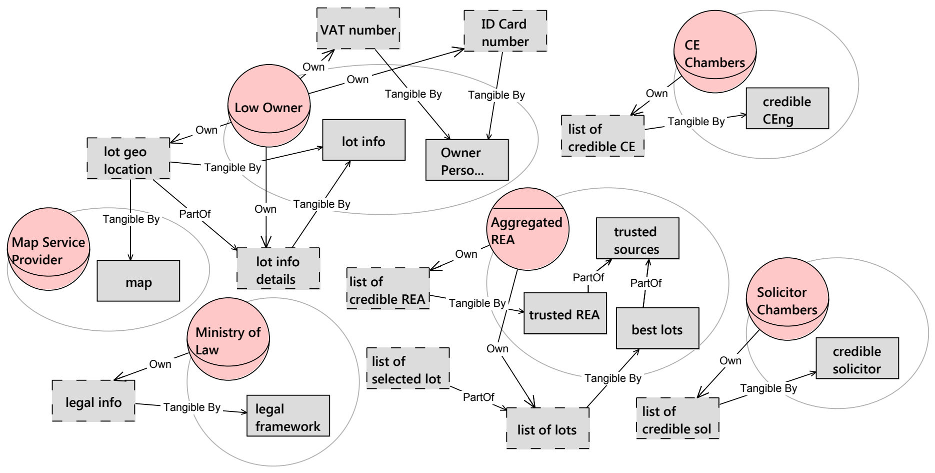


Fig. 3: Lot Searching scenario—Information View

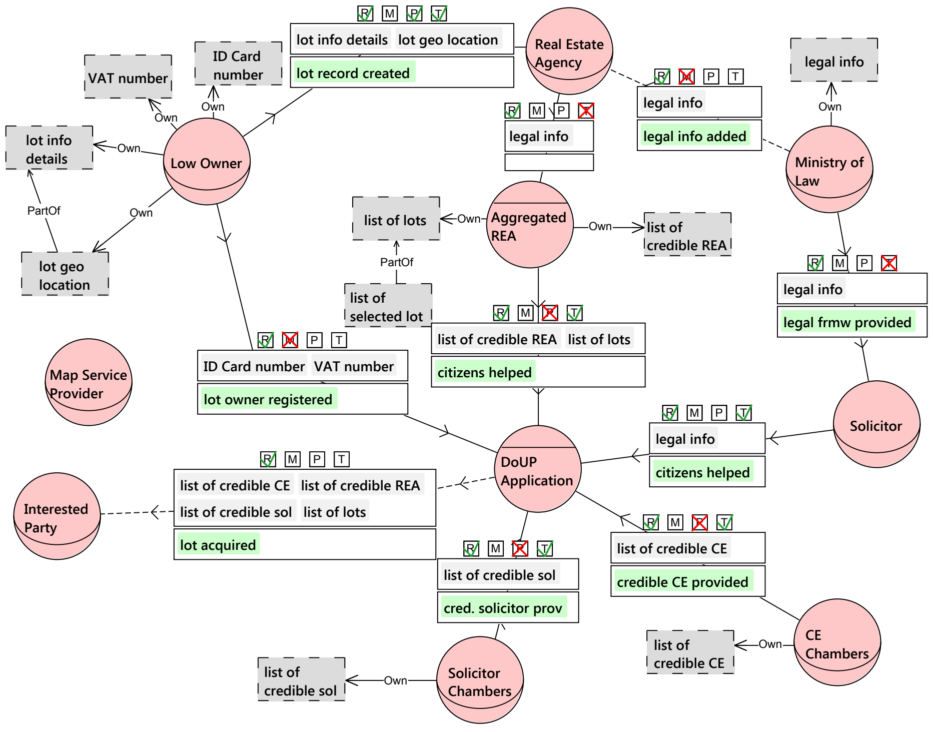


Fig. 4: Lot Searching scenario—Authorisation View

Authorisation View. STS-ml includes the primitive authorisation, see Fig. 4, to capture two key concepts in security, namely *permissions* and *prohibitions*. The main idea behind this view, is that actors (typically information owners) may want to specify what they allow or prohibit others to do over their proprietary information. Following this intuition, the authorisation relationship in STS-ml is specified over four dimensions:

- *Allowed/prohibited operations*: they define whether the actor is permitted (green tick symbol) or prohibited (red cross symbol) to Read (R), Modify (M), Produce (P), and Trasnmit (T) any document that makes tangible the information (operations are graphically represented in four boxes with distinguishable labels, R, M, P, and T respectively). For instance, in Fig. 4, the *Lot Owner* authorises *Real Estate Agency* to read, produce, and transmit information *lot info details* and *lot geo location*. No prohibitions are specified through this authorisation relationship. Instead a prohibition on modifying information *legal info* is expressed from the *Ministry of Law* to *Real Estate Agency*.
- *Information*: authorisation is granted over at least one information entity. Given the structuring of information in terms of part-of relationships, authorising some actor over some information means that the actor is authorised over parts of information as well, because ownership of information propagates top-down through part-of relationships. The information entities over which authorisation is specified is represented right below the allowed/prohibited operations.
- *Scope of authorisation*: authority over information can be limited to the scope of a certain goal. As such, scope of authorisation defines the goals for the fulfillment of which the authorisation is granted. In other words, the authorisation is restricted to a certain purpose, and does not apply to different purposes. Our notion of goal scope adopts the definition in [1], 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. For instance, in Fig. 4, the *Lot Owner* authorises *Real Estate Agency* in the scope of goal *lot record created*, not for every goal of *Real Estate Agency*.
- *Transferability of the permissions*: it specifies whether the actor that receives the authorisation is in turn entitled to transfer the received permissions or specify prohibitions (concerning the received permissions) to other actors. Graphically, transferability of the authorisation is allowed when the authorisation arrow line connecting the two actors is solid, while it is not granted when it is dashed. The authorisation from *Lot Owner* to *Real Estate Agency* is a transferable authorisation (continuous/solid arrow line), while the one from *DoUP Application* to the *Interested Party* granting the authority to read information *list of credible CE*, *list of credible REA*, *list of credible sol* and *list of lots* for goal *lot acquired*, is a non-transferrable authorisation (dashed arrow line).

2.2 Security Requirements in STS-ml

Through its three views, STS-ml supports different types of security requirements. In the social view security requirements are specified over the social relationships in which actors take part, such as goal delegation and document transmission. Moreover, a number of supported security requirements is imposed by the regulatory framework and laws in place, which restrict responsibility uptake and role adoptions. The information view serves as a brigde between the social and authorisation view, for a richer set of security requirements. As such, no security requirements are expressed in the information view. In the authorisation view, security requirements are expressed through the authorisation relationships themselves.

The following is the list of security requirements supported by the social view:

1. *Over goal delegations:*
   (a) *No-redelegation*—the re-delegation of the fulfilment of a goal is forbidden; in Fig. 5 *Lot Owner* requires the *Real Estate Agency* not to redelegate the goal *lot record created*.
   (b) *Non-repudiation*—the delegator cannot repudiate he/she delegated (*non-repudiation of delegation*); and the delegatee cannot repudiate he/she accepted the delegation (*non-repudiation of acceptance*); for instance, in Example 1, *DoUP Application* requires *CE Chambers* the *non-repudiation of the acceptance* of goal *credible CE provided*, see Fig. 1.
   (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 requirements, namely, (i) *fallback redundancy single*, (ii) *fallback redundancy multi*, (iii) *true redundancy single*, and (iv) *true redundancy multi*. In Fig. 6, *Interested Party* imposes on the *DoUP Application* a *true redundancy single* security requirement for goal *trusted REA selected*.
   (d) *Trustworthiness*—the delegation of the goal will take place only if the delegatee is trustworthy; for instance, the delegation of goal *trusted REA selected* from *Interested Party* to *DoUP Application* will take place only to trustworthy application providers, see Fig. 5.
   (e) *Goal Availability*—the delegatee should ensure a minim availability level for the delegated goal; for instance, *Lot Owner* requires *Real Estate Agency* 90% availability for goal *lot record created*, see Fig. 5.
   Note that security requirements over goal delegations are expressed through annotations over these relationships, graphically represented through a padlock symbol, and made explicitly visible under the goal itself, when selected. Different labels and colours are used to distinguish them.
2. *Over document transmissions:*
   (a) *Non-repudiation*—the sender cannot repudiate he/she transmitted (*non-repudiation of transmission*); and the delegatee cannot repudiate he/she received (accepted) the transmission (*non-repudiation of acceptance*);

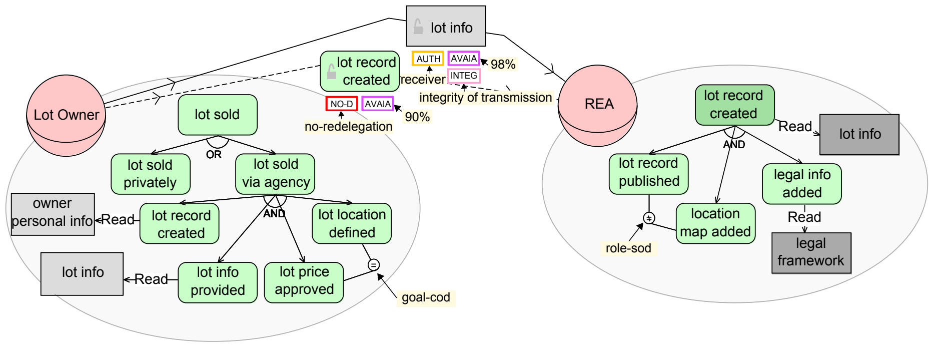


Fig. 5: Capturing security requirements from security needs for REA

(b) *Integrity of transmission*—the sender should ensure that the document shall not be altered while transmitting it (*sender integrity*); the receiver shall ensure the integrity of transmission for the given document is preserved (*receiver integrity*); and the system shall ensure that the integrity of transmission of a document in transit is preserved (*system integrity*). For instance, in Fig. 6, *DoUP Application* shall ensure sender integrity on the transmission of document *best lots* to *Interested Party*.

(c) *Confidentiality of transmission*—the sender should ensure the confidentiality of transmission for the given document (*sender confidentiality*); the receiver shall ensure the confidentiality of transmission for the given document is preserved (*receiver confidentiality*); and the system shall ensure that the confidentiality of transmission of a document in transit is preserved (*system confidentiality*). For instance, in Fig. 6, *DoUP Application* shall ensure sender confidentiality on the transmission of document *credible solicitor* to *Interested Party*.

(d) *Document Availability*—the sender should ensure a minimal availability level (in percentage) for the transmitted document. In Fig. 6, *DoUP Application* should ensure an availability level of 94% for the document *best lots* and an availability level of 90% for the document *credible solicitor*, when transmitting both these documents to *Interested Party*.
Note that security requirements over document transmissions are expressed through annotations over these relationships, graphically represented through a padlock symbol, and made explicitly visible under the document itself, when selected. Different labels and colours are used to distinguish the various supported security requirements over document transmissions.

3. *Over responsibility uptake* [5]*:*
   (a) *Separation of duties* (SoD)—defines incompatible roles and incompatible goals, so we define two types: *role-SoD*—two roles are incompatible, i.e., cannot be played by the same agent, and *goal-SoD*—two goals shall be achieved by different actors; for instance, the goals *lot record published* and *location map added* are defined as incompatible (unequals sign, see Fig. 5). An example of

[5] Imposed either by the rules and regulations of the organisation, or by law.

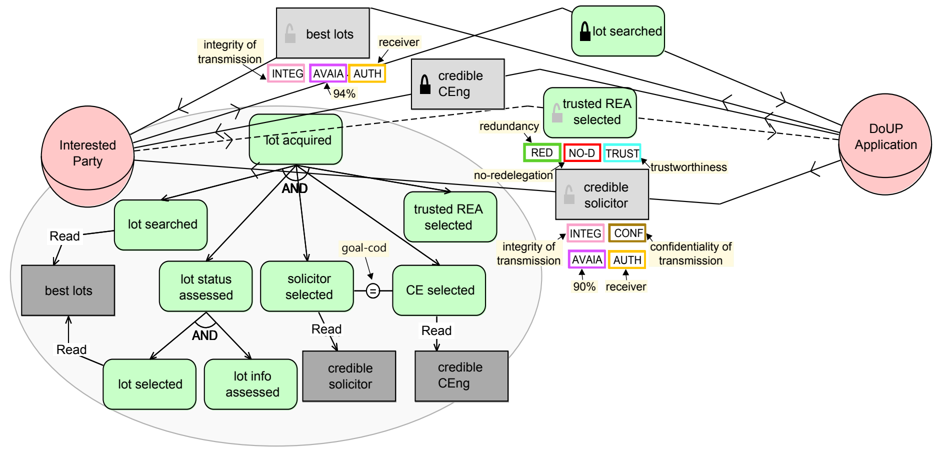


Fig. 6: Capturing security requirements from security needs for Interested Party

role-SoD is shown in Fig. 1 among roles *CE Chambers* and *Solicitor Chambers*.

(b) *Combination of duties* (CoD)—defines combinable roles and combinable goals, so we distinguish between *role-CoD*—two roles are combinable, i.e., shall be played by the same agent; and *goal-CoD*—two goals shall be achieved by the same actor. For instance, in Fig. 5, there is a *goal-CoD* expressed among goals *solicitor selected* and *CE selected* of *Interested Party*. Note that these security requirements from organisational constraints are captured through a set of relationships, namely *incompatible* (represented 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.

Security requirements over authorisations are captured implicitly by prohibiting certain operations and limiting the scope of the authorisation:

– Limiting the scope of the authorisation expresses a *need-to-know* security requirement, which requires that information is read, modified, produced only for the specified scope; for instance, *Lot Owner* authorises *DoUP Application* to read information *ID Card number* and *VAT number* only for the purpose of being registered (goal *lot owner registered*), expressing a need-to-know security requirement to *DoUP Application*, on reading this information only for *lot owner registered*, see Fig. 4.
– Prohibiting the read operation expresses a *non-reading* security requirement, which requires the information is not read in an unauthorised way; it implies that the authorisee should not read any documents making tangible the specified information. There are no examples of the *non-reading* security requiremet in Example 1.
– Prohibiting the modify operation expresses a *non-modification* security requirement, which requires the information is not modified in an unauthorised way; it implies

that the authorisee should not modify any documents making tangible this information. For instance, *DoUP Application* cannot modify documents representing information *ID Card number* and *VAT number*, for the authorisation from *Lot Owner* grants the right to read information *ID Card number* and *VAT number*, but prohibits the right to modify these information entities, see Fig. 4.

– Prohibiting the produce operation expresses a *non-production* security requirement, which requires the information is not produced in an unauthorised way; it implies that no new document, representing the given information, is produced. In Fig. 4, *DoUP Application* cannot produce documents that represent information *list of credible solicitors* or information *list of credible CE*, given that the authorisations from *Solicitor Chambers* and *CE Chambers* prohibit the operation to produce the respective information entities.
– Prohibiting the transmit operation expresses a *non-disclosure* security requirement, which 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. 4, *Solicitor* cannot transmit documents representing information *legal info*.
– Setting the transferrability dimension to false expresses a *non-reauthorisation* security requirement, which requires the authorisation is not transferrable, i.e., the authorisee shall 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 requirement implies a not-reauthorise security requirement for the operations that are not allowed. An example of explicit *non-reauthorisation* in Fig. 4 is expressed by *Ministry of Law* to the *Real Estate Agency*, given that the authorisation coming from the first on information *legal info* is non-transferable (dashed arrow line).

3 Security Requirements Specification for Composite Services with STS-ml

With the help of Example 1, we showed the interactions among the various actors in the eGovernment Lot searching scenario, in particular the interactions with the *DoUP Application*, which is in fact an application that helps citizens making use of a number of services (services that compose *DoUP Application*'s main service), such as: providing the list of credible civil engineers (for which it relies, via a goal delegation, on the *CE Chambers*), providing the list of credible solicitors (for which it relies on the *Solicitor Chambers*), searching for a lot (for which it relies on the *Aggregated REA*), etc. In the same spirit, to offer the best service to citizens, the *DoUP Application* makes use of information such as the *legal framework* (obtained from *Solicitor*, who received it from the *Ministry of Law*).

Notice that the social relationships supported by STS-ml reflect rigorously the service-oriented paradigm, capturing the interactions among a service consumer and a service provider via goal delegations and document transmissions. The interaction between a delegator and a delegatee is similar to that of a service consumer (represented by the delegator) and a service provider (represented by the delegatee) on consuming/furnishing a service (represented by the goal). The same stands for document transmissions too, the sender is the service provider, while the receiver is the service consumer.

Security requirements, on the other hand, reflect the constraints to be integrated and implemented by service interfaces. Think for instance about *non-repudiation*. This security requirements is at the basis of the contracting that occurs among various service providers: a service provider (acting as a consumer in this case) interacts with another provider for a particular service. Non-repudiation is required to ensure that collaborating parties are legally bound when an agreement is reached [7]. The satisfaction of non-repudiation mechanisms such as proof of fulfilment could be employed.

A security requirement for *not-redelegation* imposes limitations to service providers, for they are required not to rely on third parties for offering the required services.

Authentication is typically concerned with who exactly is trying to use the service [7]. This involves confirming a claim that two references to identities are the same, for example, that the sender of a message is the same person. In STS-ml, we extend this to support dual authentication given that any actor could act both as a service consumer and as a service provider.

Notice that *goal availability* (similarly *document 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 between providers and consumers.

Authorisations capture what service consumers are allowed to do. Typically consumers are permitted to use the requested service, however they cannot read the internal policies of the service provider.

Similarly, the rest of the supported set of STS-ml security requirements is to be translated to a service interface specification. But, can these specifications be satisfied by the said services and their respective providers? We aim at providing an answer to this question through automated analysis, to avoid inconsistencies and conflicts before going towards service deployment that might lead to a service not satisfying all security requirements.

4 Automated Analysis

STS-ml supports different automated analyses types. Firstly, given that we are dealing with requirements models that tend to become large and complex, an analysis of the well-formedness is required, to ensure that the created models are syntactically correct, see Section 4.1. Secondly, we verify whether there are any conflicts among the specified security requirements that might lead to a composite service not to be able to satisfy them all at the same time, see Section 4.2. Finally, considering the social and organisational threats affecting either services (represented via goals) or the information they might need to provide the required functionality (be fulfilled), we calculate the impact these threats have on the rest of the system, see Section 4.3.

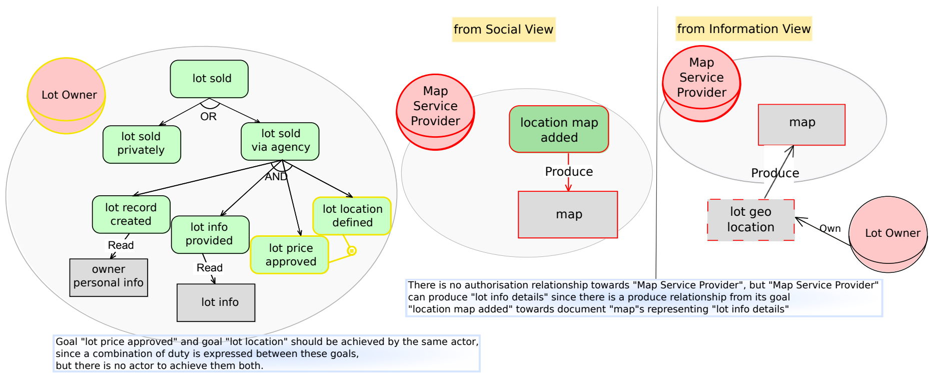
After constructing the STS-ml model for Example 1, we can run the automated analyses to verify its consistency, the satisfaction (or possible violation) of security requirements, and the threat propagation over actors' assets.

4.1 Well-formedness Analysis

The purpose of this analysis is to verify whether the diagram built by the security requirements engineer 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 well-formedness analysis for the scenario of Example 1 did not find any warnings or errors.

4.2 Security Analysis: Reasoning over Security Requirements

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. Under the hood, this analysis is implemented in disjunctive Datalog [3] and consists of comparing the possible actor behaviors that the model describes against the behavior mandated by the security requirements. Principally, requirements define actions that actors must do (or must not do). Conflicts are then identified whenever: (i) actors do actions that security requirements specify they must not do, (ii) actors do not do actions that the security requirements they should comply with mandate doing.



There is no authorisation relationship towards "Map Service Provider", but "Map Service Provider" can produce "lot info details" since there is a produce relationship from its goal "location map added" towards document "map"s representing "lot info details"

Goal "lot price approved" and goal "lot location" should be achieved by the same actor, since a combination of duty is expressed between these goals, but there is no actor to achieve them both.

Fig. 7: Executing security analysis: visualisation of results

The security analysis found several violations (errors) of the specified security requirements, such as for instance the violation of non-production by the *Map Service Provider*. As it can be seen by the diagram in Fig. 4 showing authorisation relations, there is no authorisation relationship towards *Map Service Provider* on information *lot geo location* which, following the semantics of STS-ml, is translated into an authorisation from the owner of this information, namely *Lot Owner*, prohibiting all operations over this information. This means that the *Map Service Provider* is required all security requirements derived from an authorisation relationship over the given information (i.e., *non-reading*, *non-modification*, *non-production*, *non-disclosure*, *not-reauthorisation*). But, from Fig. 1, we see that *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*, owned by the *Lot Owner* who requires non-production of this information. Thus, we identify a conflict that results in the violation of the non-production security requirement.

Similarly, there is a possible violation of a combination of duties between the goals *lot price approved* and *lot location defined* of *Lot Owner*. A combination of duties requires that the same actor pursues both goals, but there is no single actor achieving both these goals, see Fig. 7. However, this could change during runtime, and is to be verified through monitoring techniques. At the design level, we verify throughout the models whether any strategies are undertaken to fulfil the imposed security requirement. Therefore, this conflict is considered a warning, differently from the previous one which is considered an error. Warnings may be skipped, while errors need to be resolved before implementation. Resolution techniques might, however, require negotiation among service consumers and providers, as well as trade-off analysis [4].

4.3 Threat Analysis

Threat analysis is concerned with calculating 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?*"

We consider the threats shown in Fig. 2 and calculate their impact. We present the results of this analysis for the event *list not found* threatening goal *credible solicitor provided* in Fig. 8.

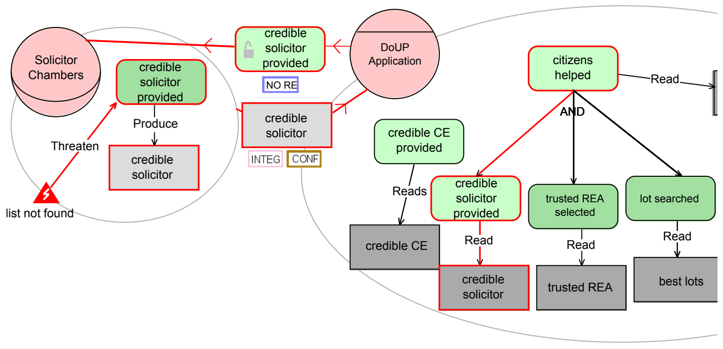


Fig. 8: Executing threat analysis

Considering the results of the threat analysis, we may need to consider a service recomposition, which does not account on *Solicitor Chambers* for offering service *cred-*

ible solicitor provided, since from our information on this provider there are no other alternatives to provide the said service.

5 Conclusions

We discussed the need for early awareness and analysis of security issues (requirements) compositive services should take into account already at design time, before they are implemented.

We presented STS-ml, a security requirements modelling language that relates security to interaction, which makes it suitable to identify security problematics strongly related to the interactions among service consumers and service providers.

We demonstrated how the reasoning techniques offered by STS-ml help designers identify possible conflicts and violations of security requirements for composite services. In particular, social threats putting at risk the well-functioning of the composite service are considered, together with the impact they have on the rest of the system.

References

1. Fabiano Dalpiaz, Amit K Chopra, Paolo Giorgini, and John Mylopoulos. Adaptation in open systems: giving interaction its rightful place. In *Proceedings of the 29th International Conference on Conceptual Modeling*, volume 6412 of *LNCS*, pages 31–45, 2010.
2. Fabiano Dalpiaz, Elda Paja, and Paolo Giorgini. Security requirements engineering via commitments. In *Proceedings of STAST'11*, pages 1–8, 2011.
3. Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive datalog. *ACM Transactions on Database Systems (TODS)*, 22(3):364–418, 1997.
4. Golnaz Elahi and Eric Yu. A Goal Oriented Approach for Modeling and Analyzing Security Trade-offs. In *Proceedings of the 26th International Conference on Conceptual modeling (ER 2007)*, volume 4801 of *LNCS*, pages 375–390, 2007.
5. Paolo Giorgini, Fabio Massacci, John Mylopoulos, and Nicola Zannone. Modeling security requirements through ownership, permission and delegation. In *Proc. of RE'05*, pages 167–176, 2005.
6. Munindar P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7(1):97–113, 1999.
7. Munindar P. Singh and Michael N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons, Chichester, UK, 2005.
8. Sandra Trösterer, Elke Beck, Fabiano Dalpiaz, Elda Paja, Paolo Giorgini, and Manfred Tscheligi. Formative user-centered evaluation of security modeling: Results from a case study. *International Journal of Secure Software Engineering*, 3(1):1–19, 2012.
9. Eric Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, University of Toronto, Canada, 1996.