Toward GDPR-Compliant Socio-Technical Systems: Modeling Language and Reasoning Framework

Marco Robol(✉), Mattia Salnitri, and Paolo Giorgini

University of Trento, Trento, Italy
{marco.robol,mattia.salnitri,paolo.giorgini}@unitn.it

Abstract. Privacy is a key aspect for the European Union (EU), where it is regulated by a specific law, the General Data Protection Regulation (GDPR). Compliance to the GDPR is a problem for organizations, it imposes strict constraints whenever they deal with personal data and, in case of infringement, it specifies severe consequences such as legal and monetary penalties. Such organizations frequently are complex systems, where personal data is processed by humans and technical services. Therefore, it becomes fundamental to consider privacy from the social perspective when designing such system, i.e., when relations between different components are specified. This is, indeed, also specified in the GDPR, which encourages to apply privacy-by-design principles. This paper proposes a method to support the design of GDPR compliant systems, based on a socio-technical approach composed of a modeling language and a reasoning framework.

Keywords: Privacy · Socio-technical systems · Requirement engineering · Modeling languages · Automated reasoning

1 Introduction

Privacy is a central aspect for citizens and it is strongly safeguarded in most countries all over the world. The European Union (EU) has recently developed a privacy law, the General Data Protection Regulation (GDPR) [2], with the aim to equalize privacy safeguard and constraints among all EU member states. All organizations that operate with personal data of EU citizens must be compliant with the GDPR, otherwise, they may occur into legal punishments and monetary penalties. Such organizations are frequently socio-technical systems, i.e., they are composed by people and machines that interact and share personal data of citizens to achieve their objectives. For example, an hospital is considered a socio-technical system since doctors provide medical treatments to patients, they use a database service to store medical record of patients and they use televisit

G. Poels et al. (Eds.): PoEM 2017, LNBIP 305, pp. 236–250, 2017.
https://doi.org/10.1007/978-3-319-70241-4_16

systems to visit patients remotely. Given the complexity of such systems, privacy has to be considered since from design phase, indeed also the GDPR incentives the principle of privacy-by-design, and privacy analysis should includes not only the technical aspects but also the social aspects, i.e., the interactions and the dependencies between people and machines.

The literature offers many approaches to support the principle of privacy-by-design. Design strategies are proposed in [8] as a guide to the main Privacy Enhancing Technologies (PETs) and patterns. In [6], data minimization concept is proposed as the fundamental aspect of privacy-by-design practices. In [11], two different approaches are proposed to privacy: (i) privacy-by-policy, based on the principle of fair information practices; and (ii) privacy-by-architecture, based on the concept of data minimization. There are also attempts to handle security at design-time an that consider the social aspects of organizations. A socio-technical approach to security, that considers a system as a composition of autonomous actors each with its own goals, is adopted by SI* [3], a security requirement engineering framework that extends i^* to handle security concepts. The Socio-Technical Security (STS) [1] adopts a socio-technical approach to support the design of secure systems. It handles only the confidentiality aspect of privacy, globally seen as an aspect of security.

As far as our knowledge goes, no privacy-oriented modeling languages have been proposed that consider privacy aspects during the design of socio-technical systems. In particular, current state-of-art modeling languages and methods lack of specific concepts and relations that can be used to specify requirements and constraints expressed by privacy regulations, such as the GDPR, and verify the compliance of organizations.

In this paper, we propose a modeling framework inspired by the privacy-by-design principle to support the design of GDPR compliant systems. The method is a revision of the STS method [1]: it extends STS-ml, the goal based modeling language provided in STS, with privacy-oriented concepts such as personal data, user consent, legal basis, and the legal figures of data controller and data processor. The extension of STS-ml provided in this paper is supported by a reasoning method to verify model properties expressing privacy policies derived by legal constraints.

The paper is structured as follows. A baseline section explains the STS method, after having introduced a case study. Section 3 presents our proposed method: we define new constructs for the STS-ml language. Section 4 presents the automated reasoning support framework. A related work section presents the main works in the field. The paper ends with some considerations on the obtained results and then general conclusions are drawn.

2 Baseline

The research work described in this paper is based on STS [1] and extends it with privacy-specific concepts, in particular the legal aspects introduced by the GDPR [2]. This section introduces a case study about the medical domain and the STS method.

2.1 Case Study

We consider a case study about the health-care services, focusing on patients data. Particularly, we will focus on the management of patient medical history by the national health-care system and the production of analysis report by private analysis laboratories. Medical data vary from medical analysis results to diagnosis, medical history of the patient, medical prescription, or report. Parties include private organizations, family doctor, the national health-care system, hospitals, and regional health-care services companies.

The case study will focus on the Trentino's health-care services organization, Azienda Provinciale per i Servizi Sanitari (APPS). APSS offers the medical performance through the national health-care system and private medical services organizations. The main hospital in Trento is the S. Chiara, which has a very complex organizational structure with different responsibility figures, such as, director, surgery doctor, nurse, and many others.

2.2 Socio-Technical Security Method

STS [1] is a method for the design of secure complex systems. Complex systems can be modeled as socio-technical system, where a composition of autonomous actors, machines and humans, depend one to another to achieve objectives. STS adopts a socio-technical approach to handle security by considering the system as a whole, and focusing on social aspects among the participants. The method is composed by a goal-based modeling language and a supporting tool with automated reasoning capabilities.

STS-ml is the formally defined language provided by the STS method. It is a goal-based modeling language that relies on the concept of intentional actor and focuses on dependencies among them, in terms of goals delegation and documents transmission. Distinct aspects of the same model are represented in three separate views, with a different diagram in each of them. The rest of the section explain STS-ml in using the case study.

Figure 1 shows an example of STS model. Example of diagrams in the three views of STS are showed in the left part of the Figure. The right part shows a legend of STS graphical notation.

Actors are modeled in the **social view**, each with a set of objectives, called goals, and a set of documents they can use. In the top left part of Fig. 1 is represented the social view diagram of an example of socio-technical system taken from the case study. Goals can be delegated among actors and documents can be transmitted. In the example, patient delegates the goal medical prescription to family doctor, who receives a diagnosis document from private company doctor. Goals can be AND- or OR-decomposed in sub-goals and can depends on a document in terms of reading, editing, or production. As represented in the model, the doctor need to read the document containing the diagnosis, in order to prescribe a medication to the patient.

STS allows to identify information: intangible assets that represent pieces of knowledge relevant for the actors of the system, for example, medical records.

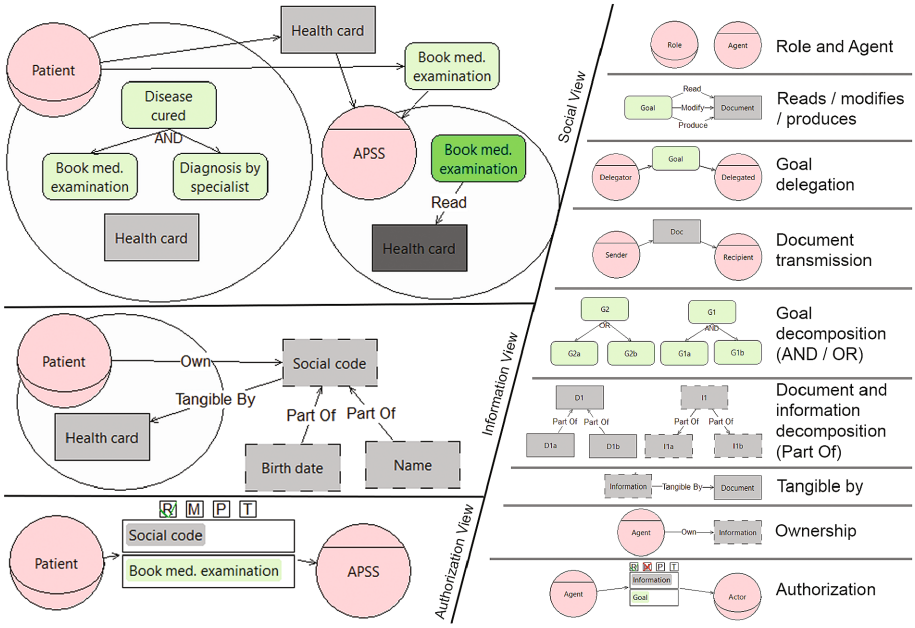

Fig. 1. STS model

Information is modeled in the **information view**, where it is possible to specify the respective owners, the composition of these information with respect to other information, and the documents that make them tangible. An example of information view diagram is showed in the middle left part of Fig. 1. Every document is possessed only by an actor, who can use them to accomplish goals, but can also be transmitted and used by others, as modeled in the social view. For example, Health Card is a document possessed by the patient that makes tangible his social code. The social code information is composed by information about his name and birth.

In order to process a document (read, modify, or produce), actors must be either the owners of the contained information, or be authorized by the owners. Authorizations are represented in the **authorization view** and are identified by: the sender, the addressee, the information, and the goals for which the information can be used. For example, the patient authorizes the APSS to read his social code within the context of booking a medical examination. An example of an authorization view diagram is showed in the bottom left part of Fig. 1.

In STS, actors are distinguished into **roles and agents**. An agent is a participant known to be in the system already at design time, for example the APSS in Fig. 1, while a role represents an actor or a class of actors identifiable by a common behavior, for example the director of the S. Chiara Hospital or the Patient of Fig. 1. Agents can eventually play several roles and roles can be eventually played by several agents.

A supporting tool is provided with **automated reasoning** capabilities, allowed by disjunctive data logic rules that formalizes the modeling language. This allows several properties checking on the model, such as, well-formedness.

3 Revision of STS-ml for Privacy

GDPR [2] imposes strict constraints on organizations whenever dealing with personal data. To support the design of GDPR-compliant systems, we propose a method that, adopting a socio-technical approach, provides a modeling language with an automated reasoning framework to model organizations in terms of intentional actors, goals, documents and information. The modeling language is an extension of STS-ml with privacy concepts added.

In the next sections, we introduce the meta-model of the language, its differences with STS-ml, and we explain how the language supports the modeling of constraints specifically imposed by the GDPR.

3.1 Modeling Language Meta-Model

As in STS-ml, the modeling language proposed in this paper relies on three different views to represent different aspects in separate diagrams. In this section, we present the modeling language proposed in this paper.

Figure 2 shows the meta-model of the social view, where actors are represented with their goals and documents. Relationships between actors are also showed in terms of goal delegation and document transmission. The elements of agent and role, which remains the same as in STS-ml, here are omitted from the meta-models for simplicity.

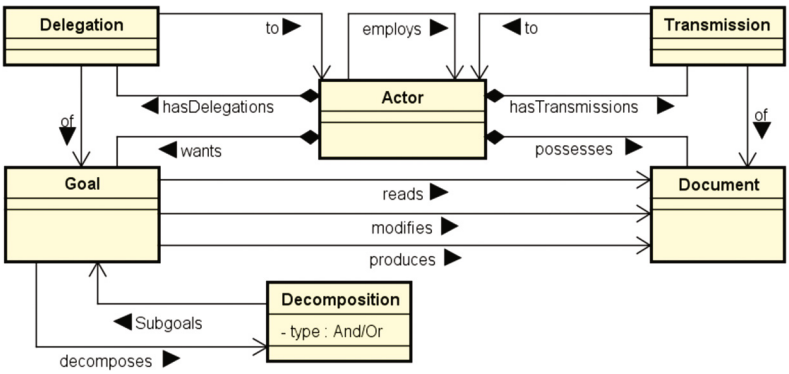

Fig. 2. Meta-model of the social view of the modeling language

Figure 3 shows the meta-model of the information and authorization views. The left part of Fig. 3 is the meta-model of the **information view**, where information are represented with their owners and are associate with the documents

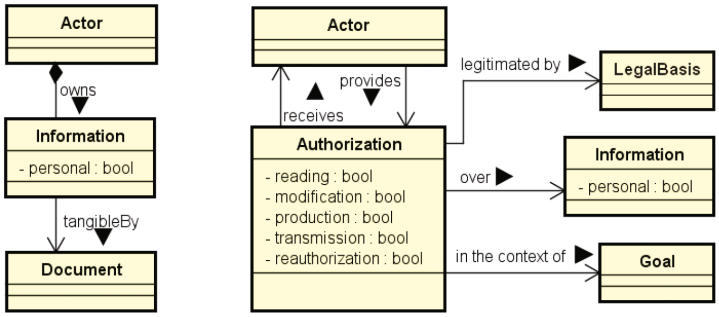

Fig. 3. Meta-model of the information and authorization views

that makes them tangible. The right part of Fig. 3 represents the meta-model of the **authorization view**.

With respect to STS-ml, the modeling language proposed in this paper, focusing on privacy, permits to represent the fundamental concept of **personal data**, as an information. In the meta-model in Fig. 3, a boolean attribute discriminates a personal data from an information. In the revisited language, we also included the **employment** relationship, which is defined from actor to actor. It is used to distinguish third party actors from actors within the same organization. This aspect is fundamental when considering laws and the constraints that they impose on organizations, as in the case of privacy laws. To represent the legal aspects of privacy, we also allow to specify the **legal basis** needed to legitimate the processing of a document revealing personal data. The legitimation relationship is defined between a legal basis and an authorization.

3.2 Representing GDPR Principles

The method proposed in this paper takes into account legal aspects of privacy, to support the representation of social constrains imposed by privacy laws, such as, the GDPR [2]. In this section, we shows how it is possible to represent GDPR constrains in the modeling language.

The modeling language already supports the representation of **personal data**, which in the GDPR are defined as information that is possible to relate to an identified or identifiable natural person, where an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person. Figure 4 shows an example of a diagram where a small letter P in the top right angle of the information identify it as a personal data.

A **natural person** is an individual human being who has its own legal personality. The language does not explicitly support the concept of natural person but only the ones of agent and role, which represent active participants of the

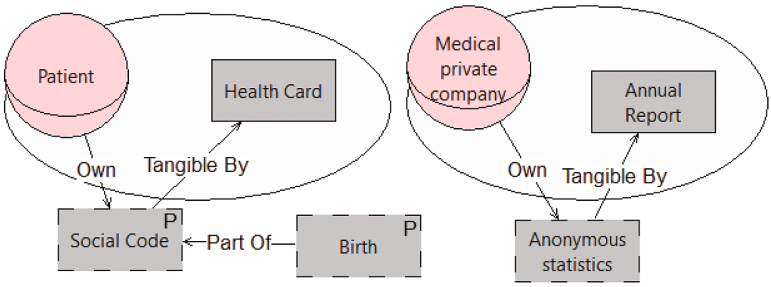

Fig. 4. Information view: personal data

socio-technical system and can be either machines, organizations, or humans, indeed natural persons. For example, APSS is an agent and represents an organization, while patient is a role and represents a class of participants which is a class natural persons.

Data subject is defined as the natural person the data are about. While to data subjects are always granted the right of being in control of their personal data, there are exceptions as in the case of under-aged people, where personal data are under the control of parents or tutors. The modeling language does not explicitly support the concept of data subject but only the one of data owner.

The GDPR identifies two types of responsibilities among the parties (actors) involved in every processing of personal data: controller and processor. GDPR is primarily imposed on **data controller**, who has the primary responsibility for compliance. **data processor**, instead, has limited responsibilities, acting on behalf of the controller by means of a written contract. Employees of data controller or data processors are not to be considered data processors on their own. In the modeling language we identify these two figures by defining direct or indirect authorizations. Data controllers are identified by a direct authorization by the owner, while data processors are identified by an indirect authorization (re-authorization) done by the data controller.

We distinguish among authorizations given to a third party and internal authorizations given to an employee with an apposite relationship defined between actors that identifies employers and employees actors: the **employment relationship**. The employment relationship is modeled in the social view. Figure 5 shows the employment relationship between medical private company and private company doctor.

The GDPR requires a **legal basis** for every lawful processing of personal data. The language allows to model the legal basis needed to legitimate the transmission and the processing of personal data; it also allows to adopt one of them in each authorization involving personal data. The GDPR defines a set of legal basis however, it allows EU member states to define further legal basis. For the above reason, the modeling language described in this paper does not provide a set of predefined legal basis but allows to specify them in a simple and minimalistic way, where all legal basis are represented in a list and each on

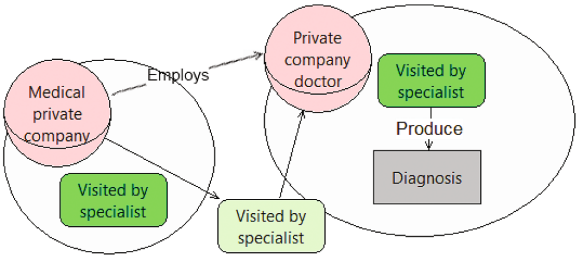

Fig. 5. Social view: employment

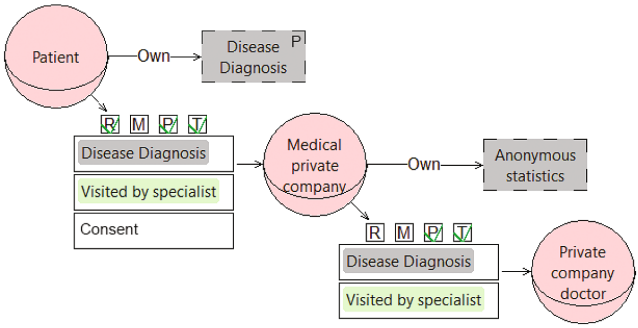

Fig. 6. Authorization view: legitimation

them is provided with a name and a textual description. Figure 6 shows a legal basis provided in the context of the authorization that includes personal data between patient and medical private company.

4 Automated Reasoning Support

Models are an approximation of the world represented in a structured form. Modeling languages should be enough simple to be understandable by non-expert users but, as the models start growing, they could became complex. Automated reasoning can support the designer in the identification of potential inconsistencies in the models, conflicts, and violation of properties.

The formal specification of the language includes constraints on the composition of the model, so that well-formedness can be automatically checked by automated reasoning. This is useful to avoid inconsistencies in the models, such as, cyclical delegations or chains of employment. The formalization allows for a flexible specification of further properties that can be automatically verified, such as, constraints introduced by the GDPR.

In this section, we give a formalization of the language by providing a set of rules that can be automatically verified, then we show how it is possible to formally represent GDPR constraints, so to automate their verification.

4.1 Core Definitions

The formalization of the language is a revision of the one given by Dalpiaz et al. in [1], with, in addition, the formalization of the concepts presented in the previous section. We define atomic variables with strings in italic with a leading capital letter (e.g., G, I); sets are defined with strings in the calligraphic font for mathematical expressions (e.g., $\mathcal{G}$, $\mathcal{I}$); relationship are defined in typewriter style with a leading non-capital letter (e.g., *wants*, *possesses*).

Predicates are provided to represent concepts, general relationships among them, intentional relationships, and social relationships. For those inherited from STS-ml we kept the same definition given by STS-ml authors in [1].

Table 1 lists the predicates used in the social view of the modeling language. The employment relationship defined between two actors with the predicate $\mathit{employs}(\mathsf{A}, \mathsf{A}')$ is used to model the legal contract that exists between an employer and his employees.

Table 1. Social view predicates

Concepts	$\mathit{actor}(\mathsf{A})$, $\mathit{agent}(\mathsf{Ag})$, $\mathit{role}(\mathsf{R})$, $\mathit{goal}(\mathsf{G})$, $\mathit{document}(\mathsf{D})$
Intentional relationships ($\mathcal{IRL}$)	$\mathit{wants}(\mathsf{A}, \mathsf{G})$, $\mathit{possesses}(\mathsf{A}, \mathsf{D})$, $\mathit{decomposes}(\mathsf{A}, \mathsf{G}, \mathcal{S}, \mathsf{DecT})$, $\mathit{reads/modifies/produces}(\mathsf{A}, \mathsf{G}, \mathsf{D}, \mathsf{OpT})$ (where $\mathsf{OpT} \in \{R, M, P\}$ and $\mathsf{DecT} \in \{\mathit{and}, \mathit{or}\}$)
Social relationships ($\mathcal{SR}$)	$\mathit{plays}(\mathsf{Ag}, \mathsf{R})$, $\mathit{delegates}(\mathsf{A}, \mathsf{A}', \mathsf{G})$, $\mathit{transmits}(\mathsf{A}, \mathsf{A}', \mathsf{D})$, $\mathit{employs}(\mathsf{A}, \mathsf{A}')$

Table 2 lists the predicates used in the information view of the modeling language. Personal data are a subclass of information and are represented with the predicate $\mathit{personalData}(\mathsf{I})$.

Table 2. Information view predicates

Concepts	$\mathit{information}(\mathsf{I})$, $\mathit{personalData}(\mathsf{I})$
Relationships	$\mathit{owns}(\mathsf{A}, \mathsf{I})$, $\mathit{makes\text{-}tangible}(\mathsf{I}, \mathsf{D})$, $\mathit{part\text{-}of\text{-}i}(\mathsf{I}_1, \mathsf{I}_2)$, $\mathit{part\text{-}of\text{-}d}(\mathsf{D}_1, \mathsf{D}_2)$

Table 3 lists the predicates used in the authorization view of the modeling language. The predicate $\mathit{legalBasis}(\mathsf{LB})$ models a law, or a law article, that can be used to legitimate something (e.g. explicit consensus legitimates the processing of personal data). The predicate $\mathit{legitimates}(\mathsf{LB}, \mathit{autorises})$ specifies the legal basis used to legitimate an authorization that includes some processing of personal data. For example, in Fig. 6 the consensus is used to legitimate the authorization for the processing of disease diagnosis provided by patient to medical private company.

Table 3. Authorization view predicates

Concepts	*legalBasis*(LB)
Social relationships ($\mathcal{SR}$)	*authorises*($\mathsf{A}_1, \mathsf{A}_2, \mathcal{I}, \mathcal{G}, \mathcal{P}$, TrAuth) (where $\mathcal{P} = (\{R, M, P, T\} \cup \{\overline{R}, \overline{M}, \overline{P}, \overline{T}\})$ and $\mathsf{TrAuth} \in \{and, or\}$)
Relationships	*legitimates*(LB, *authorises*)

Definition: Actor Model. An actor model AM is a tuple $\langle \mathsf{A}, \mathcal{G}, \mathcal{D}, \mathcal{G}_{\mathcal{IRL}}, \mathcal{D}_{\mathcal{IRL}} \rangle$, where A is an actor, $\mathcal{G}$ is a set of goals, $\mathcal{D}$ is a set of documents, $\mathcal{IRL}$ is a set of intentional relationships. We denote the set of actor models as $\mathcal{AM}$.

Actor Model Well-Formedness. An actor model $\mathsf{AM} = \langle \mathsf{A}, \mathcal{G}, \mathcal{D}, \mathcal{G}_{\mathcal{IRL}}, \mathcal{D}_{\mathcal{IRL}} \rangle$ is well-formed if: (i) for each each intentional relationship IRL = *decomposes*(A′, G, $\mathcal{S}$, DecT) in $\mathcal{IRL}$, A′ = A, and both G and all goals in $\mathcal{S}$ are in $\mathcal{G}$; and (ii) for each intentional relationship IRL = *reads/modifies/produces*(A′, G, D, OpT) in $\mathcal{IRL}$, A′ = A, $\mathsf{G} \in \mathcal{G}$, and $\mathsf{D} \in \mathcal{D}$.

Definition: Model. We tie together all the elements in the social, information, and authorization views to define a model. A model M is a tuple $\langle \mathcal{AM}, \mathcal{SR}, \mathsf{IM} \rangle$ where $\mathcal{AM}$ is a set of actor models, $\mathcal{SR}$ is a set of social relationships, and IM is an information view model. Definition 3 lays down the constraints on the well-formedness of a model.

Model Well-Formedness. A model $\mathsf{M} = \langle \mathcal{AM}, \mathcal{SR}, \mathsf{IM} \rangle$ is well-formed if:

1. social relationships are only over actors with models in $\mathcal{AM}$;
2. only leaf goals are delegated; for each *delegates*(A, A′, G) in $\mathcal{SR}$, there is an actor model $\langle \mathsf{A}, \mathcal{G}, \mathcal{D}, \mathcal{G}_{\mathcal{IRL}}, \mathcal{D}_{\mathcal{IRL}} \rangle$ in $\mathcal{AM}$ such that $\mathsf{G} \in \mathcal{G}$ and there is no decomposition of G in $\mathcal{IRL}$;
3. delegated goals appear in the delegatee's actor model: $\forall$*delegates*(A, A′, G) in $\mathcal{SR}$, there is an actor model $\langle \mathsf{A}, \mathcal{G}, \mathcal{D}, \mathcal{G}_{\mathcal{IRL}}, \mathcal{D}_{\mathcal{IRL}} \rangle$ in $\mathcal{AM}$ s.t. $\mathsf{G} \in \mathcal{G}$;
4. the transmitter must possess the document for the document transmission to take place: for each *transmits*(A, A′, D) in $\mathcal{SR}$, there is an actor model $\langle \mathsf{A}, \mathcal{G}, \mathcal{D}, \mathcal{G}_{\mathcal{IRL}}, \mathcal{D}_{\mathcal{IRL}} \rangle \in \mathcal{AM}$ s.t. $\mathsf{D} \in \mathcal{D}$. An actor possesses a document if:
   - it has the document since scratch: *possesses*(A, D) $\in \mathcal{IRL}$, or
   - it creates the document, i.e., $\exists \mathsf{G} \in \mathcal{G}.$*wants*(A, G) $\wedge$ *produces*(A, G, D) $\in \mathcal{IRL}$, or
   - there is an actor A″ such that *transmits*(A″, A, D) $\in \mathcal{SR}$.
5. transmitted documents appear in the receiver's actor model: $\forall$*transmits*(A′, A, D) $\in \mathcal{SR}, \exists \mathsf{AM} = \langle \mathsf{A}, \mathcal{G}, \mathcal{D}, \mathcal{G}_{\mathcal{IRL}}, \mathcal{D}_{\mathcal{IRL}} \rangle \in \mathcal{SR} | \mathsf{D} \in \mathcal{D}$;
6. authorizations are well-formed if: $\forall$*authorizes*(A′, A, $\mathcal{I}$, $\mathcal{G}$, $\mathcal{OP}$, TrAuth) $\in \mathcal{SR}$, there must be at least one information entity specified ($|\ \mathcal{I}\ | \geq 1$), and at least one prohibition or permission is specified;

7. delegations have no cycles: for each $delegates(\mathsf{A}, \mathsf{A}', \mathsf{G})$ in $\mathcal{SR}$, there is no A'' such that $delegates(\mathsf{A}'', \mathsf{A}, \mathsf{G})$ in $\mathcal{SR}$ or $delegates(\mathsf{A}'', \mathsf{A}, \mathsf{Gi})$ in $\mathcal{SR}$, where Gi is a descendant of G in the goal tree of A'';
8. part-of relationships (either over information or documents) have no cycles;

Definition: Authorization Closure. Let $\mathsf{M} = \langle \mathcal{AM}, \mathcal{SR}, \mathsf{IM} \rangle$ be a well-formed model. The authorization closure of $\mathcal{SR}$, denoted as $\triangle_{\mathcal{SR}}$, is a superset of $\mathcal{SR}$ that makes prohibitions explicit, when no authorization is granted by any actor. Formally: $\forall \mathsf{A}, \mathsf{A}'$ with an actor model in $\mathcal{AM}, \forall owns(\mathsf{A}, \mathsf{I}) \in \mathcal{SR}$.

$\neg \exists \mathsf{A}''.authorises(\mathsf{A}'', \mathsf{A}', \mathcal{I}, \mathcal{G}, \mathcal{OP}, \mathsf{TrAuth}) \in \mathsf{SR}$ where $\mathsf{I} \in \mathcal{I} \rightarrow$
$authorises(\mathsf{A}, \mathsf{A}', \mathsf{I}, \mathcal{G}', \overline{\mathcal{OP}}, \mathsf{false}) \in \triangle_{\mathcal{SR}}$ where $\mathcal{G}'$ is the set of goals of A'

Definition: Authorizations Conflict. Two authorizations: $authorises(\mathsf{A}_1, \mathsf{A}_2, \mathcal{I}_1, \mathcal{G}_1, \mathcal{OP}_1, \mathsf{TrAuth}_1)$, and $authorises(\mathsf{A}_3, \mathsf{A}_2, \mathcal{I}_2, \mathcal{G}_2, \mathcal{OP}_2, \mathsf{TrAuth}_2)$ are conflicting if and only if they both regulate the same information ($\mathcal{I}_1 \cap \mathcal{I}_2 \neq \emptyset$), and either:

- $\mathcal{G}_1 \neq \emptyset \wedge \mathcal{G}_2 = \emptyset$, or vice versa; or,
- $\mathcal{G}_1 \cap \mathcal{G}_2 \neq \emptyset$, and either: $\exists \mathsf{OP}.\mathsf{OP} \in \mathcal{OP}_1 \wedge \overline{\mathsf{OP}} \in \mathsf{OP}_2$; or $\mathsf{TrAuth}_1 \neq \mathsf{TrAuth}_2$.

4.2 Verification of Policies Properties

The formal definition of a modeling language allows for automated reasoning over the models, such as, the verification of model properties. In this section, we show how it is possible to formally represent some of the social constraints imposed by the GDPR and automatically verify them. An automated verification of the constraints presented later in this Section is not to be intended sufficient for a system to be compliant with the GDPR, but more as a support in the analysis of the compliance of complex systems with the GDPR.

Employment. Under the GDPR, organizations and employers have different responsibilities with respect to their employees, for what concerns the processing of personal data. To identify employers and employee, the modeling language supports the employment relationship, which can be defined only between actors within the same organization, and, more in particular, always from the employer to all his employees, so to avoid chains of employments and circular relationships.

Formally, employment relationship is defined among two actors, an employer and an employee, where it must be valid that: (i) each employer can have several employees and cannot be employed; and (ii) each employee can have only one employer and no employee: $\forall \mathsf{A}, \mathsf{A}'$, with an actor model in $\mathcal{AM}$, $\forall\ employs(\mathsf{A}, \mathsf{A}') \in \mathcal{SR}, \neg \exists \mathsf{A}'' \neq \mathsf{A}$ s.t. $employs(\mathsf{A}'', \mathsf{A}) \vee employs(\mathsf{A}', \mathsf{A}'') \vee employs(\mathsf{A}'', \mathsf{A}')$.

This policy specify the impossibility to define chains of employment relationships, so to avoid model inconsistencies, with respect to the legislation. For example, if we model the APSS as employer of doctors and nurses, and the APSS director as employer of nurses, we would have an inconsistent model. Automated reasoning can support the designer in the detection of this violation.

Data Controller. Under the GDPR, for every processing of personal data done by other subjects other then the data owner, it is always required to identify a data controller. In the language, the processing of personal data, or more generally of information, is represented by either: (i) the intentional relationship *read/modify/produce(*A,G*)*,D,OpT defined from a goal (in some actor's scope) to a document; or (ii) the social relationship *transmits*$(\mathsf{A}, \mathsf{A}', \mathsf{D})$ defined between two actors. Both of these relationships are valid only in the cases of: (i) an authorization allows for them; or (ii) the actor processing the data (reading, modifying, producing, or transmitting it) is the owner of the information itself. Formally: $\forall \mathsf{I}, \mathit{reads/modifies/produces}(\mathsf{A}, \mathsf{G}, \mathsf{D}, \mathsf{OpT}).\mathit{makes\text{-}tangible}(\mathsf{I}, \mathsf{D}) \implies \mathit{owns}(\mathsf{A}, \mathsf{I}) \vee \exists \mathit{authorises}(\mathsf{A}', \mathsf{A}, \mathcal{I}, \mathcal{G}, \mathcal{OP}, \mathsf{TrAuth}).\mathsf{OpT} \in \mathcal{OP}, \mathsf{G} \in \mathcal{G}, \mathsf{I} \in \mathcal{I}$.

The GDPR requirement of having a data controller for every processing of personal data, can be supported by automated reasoning on the social relationship of authorization. In the language, the data controller is the direct addressee of the authorization provided by the data owner, or its employer (if exists). Since an authorization is always needed in the case of information processed by someone different from the owner himself, if a data controller is missed, an authorization violation can be detected by automated reasoning, and the system designer can be notified. For example, in the model represented in Fig. 5 the goal visited by specialist produces the document diagnosis, which makes tangible personal data. This is a personal data processing and the GDPR requires for it a data controller. Figure 6 shows the authorization given by the data owner to the company for the disease diagnosis information in the context of visited by specialist, this authorizations identify the company as the data controller for this data processing. If this authorization had been missed, a data controller would not have been specified as required by the GDPR.

Legitimation. The GDPR imposes on organizations to have a valid legal basis for every processing of personal data or sharing with a third party (e.g. explicit consensus or contractual necessity). For example, when the Santa Chiara hospital transmits a document containing personal information of a patient to the APSS, the hospital needs to provide a lawful basis, such as, an explicit consensus of the data owner, in order to legitimate the authorization to the APSS.

Formally, for each authorization that includes a personal data, provided to an actor who is not in the same organization of the provider itself (from the employer to an employee, or from an employee to the employer, or among employees within the same organization), it is necessary to specify the legal basis used to legitimate the authorization. Formally: $\forall \mathit{authorises}(\mathsf{A}, \mathsf{A}', \mathcal{I}, \mathcal{G}, \mathcal{OP}, \mathsf{TrAuth}) | \neg \mathit{employs}(\mathsf{A}, \mathsf{A}') \wedge \neg \mathit{employs}(\mathsf{A}', \mathsf{A}) \wedge \neg \exists \mathsf{A}''.\mathit{employs}(\mathsf{A}'', \mathsf{A}) \wedge \mathit{employs}(\mathsf{A}'', \mathsf{A}') : \exists \mathsf{I} \in \mathcal{I}.\mathit{personalData}(\mathsf{I}) \rightarrow \exists \mathsf{LB}.\mathit{legitimates}(\mathsf{LB}, \mathit{authorises})$

In the case of complex scenarios, the number of authorizations can increase a lot and could became difficult to identify the authorizations that includes personal data and that are not between members of the same organization. The automated reasoning feature can support the designer in the identification of this critical subset of authorization, for which it is required to provide a legal

basis. For example, in the medical domain, many information are personal data but not all authorizations are defined between members of different organizations, so we could need help to identify them. For example, Fig. 6 shows two different authorizations, the one on the left, from patient to medical private company, requires a legal basis because it includes the personal data disease diagnosis and is defined between the data owner and a different actor. The authorization on the right part of Fig. 6 is about the same personal data of above, but it is defined between actors within the same organization, so it does not requires any legal basis.

5 Related Work

In this section, we compare with other works about privacy requirements engineering, privacy enhancing technologies and automated reasoning applied to modeling languages.

Hoepman et al. in [8] identify eight privacy design strategies to help IT architects to support privacy by design early in the software development life cycle. The work has been done when data protection regulation in EU was only a proposal, but the ideas of privacy by design and by default were already there. The work reviews the mains available PETs and patterns, but focuses more on the design of a system by providing design strategies. The work does not provide a concrete method for a specific privacy legislation but try to define high level strategies to adopt at the very beginning of system design.

Gurses et al. in [6] provide an overview of privacy-by-design practices, inspired by policy makers but expressed from an engineering perspective. They present two case studies focusing on the aspect of data minimization, trying to demonstrate its importance in the concept of privacy-by-design.

Spiekermann et al. in [11] try to introduce the privacy domain to engineers, by providing a concrete privacy-friendly systems design guidance. The work examines effects on user behavior of three types of system operations, data transfer, storage, and processing. The authors develop guidelines for building privacy-friendly systems, distinguishing two approaches: privacy-by-policy and privacy-by-architecture, where the former relies on the principle of fair information practices and the latter on data minimization concept. In the GDPR these two concepts are both central but there are also other aspects that are not taken into consideration by this work, such as legal responsibilities.

Guarda et al. in [5] present a technological interpretation of legal aspects of privacy for the development of requirement engineering methods. The authors recognize the legal perspective as the leading one important for privacy. They do not propose any method or language but provide an overview of privacy aspects to be taken into consideration by designing methods for privacy-aware systems. The work is antecedent to the GDPR and it is therefore based on the previous EU Directive, the 95/46/EC.

Kalloniatis et al. in [9] present PriS, a security requirement engineering method that, focusing on privacy aspects, incorporates privacy into the system design process. By considering privacy requirements as organizational goals, PriS uses the concept of privacy-process pattern for describing the impact of privacy onto the organizational processes. The work provide an effective method but does not take into consideration any legal aspect.

Qingfeng et al. in [7] present a privacy goal-driven requirements modeling framework to support the design of a Role-Based Access Control (RBAC) system. The work proposes a solution to define low-level access control policies given privacy requirements. The proposed framework addresses a single aspect of privacy, without supporting other privacy-enhancing practices or legal aspects.

Van Lamsweerde et al. in [12] propose automated analysis features to support requirements engineering methods in managing conflicts and applying design principles. Giorgini et al. in [4], introduce automated reasoning feature in order to identify conflicts among goals in the Tropos method.

Giorgini et al. in [3] presents SI*, a security requirement engineering framework that extends i^* [13][1] to handle security concepts. The work does not focus specifically on privacy but handles security aspects by providing a goal modeling language to identify dependencies between actors.

6 Conclusions and Future Work

We have proposed a method for supporting the implementation of systems compliant with the GDPR. More in particular, we have defined a goal-based modeling language, an extension of STS-ml, that allows to model social aspects of the GDPR, such as, the relationship between data subjects, data controllers, data processors, employer, and employees, in the context of personal data processing. We also showed how it is possible to use the modeling language to formally represent and automatically verify privacy policies.

Future work includes the extension of the framework with a business process language, to allow a more detailed specification of the social aspects represented in the language. A further formalization of the language will be needed to specify other constrains imposed by the GDPR. We also plan to develop a tool to support the method, by extending the STS-tool presented in [10]. It will feature automated reasoning capabilities and it will support the user in the identification of inconsistencies and in the verification of other users policies. The tool will also generate, in an automated fashion, documents which report the information and diagrams specified for a system in a format understandable to non STS-ml experts.

[1] i^* is a modeling framework that uses the concepts of actors, goals, tasks, and resources to define dependencies between actors.

References

1. Dalpiaz, F., Paja, E., Giorgini, P.: Security Requirements Engineering: Designing Secure Socio-Technical Systems. MIT Press, Cambridge (2016)
2. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union L119/59, May 2016. http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC
3. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Modeling security requirements through ownership, permission and delegation. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering, pp. 167–176. IEEE (2005)
4. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Goal-oriented requirements analysis and reasoning in the tropos methodology. Eng. Appl. Artif. Intell. **18**(2), 159–171 (2005)
5. Guarda, P., Zannone, N.: Towards the development of privacy-aware systems. Inf. Softw. Technol. **51**(2), 337–350 (2009)
6. Gürses, S., Troncoso, C., Diaz, C.: Engineering privacy by design (2011)
7. He, Q., Antón, A.I., et al.: A framework for modeling privacy requirements in role engineering. In: Proceedings of REFSQ, vol. 3, pp. 137–146 (2003)
8. Hoepman, J.-H.: Privacy design strategies. In: Cuppens-Boulahia, N., Cuppens, F., Jajodia, S., Abou El Kalam, A., Sans, T. (eds.) SEC 2014. IAICT, vol. 428, pp. 446–459. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55415-5_38
9. Kalloniatis, C., Kavakli, E., Gritzalis, S.: Addressing privacy requirements in system design: the PriS method. Requirements Eng. **13**(3), 241–255 (2008)
10. Paja, E., Dalpiaz, F., Poggianella, M., Roberti, P., Giorgini, P.: Specifying and reasoning over socio-technical security requirements with STS-tool. In: Ng, W., Storey, V.C., Trujillo, J.C. (eds.) ER 2013. LNCS, vol. 8217, pp. 504–507. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41924-9_45
11. Spiekermann, S., Cranor, L.F.: Engineering privacy. IEEE Trans. Software Eng. **35**(1), 67–82 (2009)
12. Van Lamsweerde, A., Darimont, R., Letier, E.: Managing conflicts in goal-driven requirements engineering. IEEE Trans. Software Eng. **24**(11), 908–926 (1998)
13. Yu, E.: Modelling Strategic Relationships for Process Reengineering: Social Modeling for Requirements Engineering **11** (2011)