University Of Trento

Department of Information Engineering
and Computer Science (DISI)

ICT International Doctoral School

Consent modeling and verification: privacy regulations compliance from business goals to business processes

PhD Thesis

Candidate: Marco Robol

Supervisor: Paolo Giorgini

Trento, 2020

Abstract

Privacy regulations impose on companies limitations about the collection, use, and disclosure of user data. One of the actions most companies undertake for this, consists in modifying their systems with processes for consent acquisition and management. Unfortunately, where systems are large and with many dependencies, they often also have little documentation, and knowledge on the system is distributed among different domain experts. These circumstances make the re-engineering of systems a tedious and complex, if not impossible, activity. This PhD Thesis proposes a model-based method with a top-down approach, for modeling consent requirements and analyzing compliance with regulations, by refinement of models from organizational structure down to business processes. The method is provided with guidelines in the form of a process and includes modeling languages and reasoning frameworks for the analysis of requirements with respect to a preset of privacy principles on consent. The Thesis includes validations with realistic scenarios and with domain practitioners from the healthcare domain.

Contents

Chapter 1

Introduction

Privacy is about living with discretion without being judged by others for personal choices. Internet and new technologies have facilitated the growth of available personal data, increasing the risk for privacy violations. High speed data connections and pervasive technologies, such as, smartphones, wearable devices, and smart appliances, collect information about our habits and preferences, even if we are often unaware of it. Such data are used by companies to improve the personalization of services and to retain or increase their user base. Feeding such business requires massive quantities of data, that companies obtain by collaborating with third parties by exchanging or trading our data. For example, online advertisers pay site owners to buy spaces on their web pages, but they also collect data from unaware online visitors, that they use to build customer, financial, and health profiles so to offer a more effective service to advertising companies.

Adapted to different scenarios, software systems are in continuous evolution, and while they grow, they are re-organized into smaller and smaller pieces distributed across different elaboration units or devices, managed by different companies with high degrees of autonomy. Also data sets are distributed across databases managed by different companies, where data sources are merged to infer new information from existing data, creating statistics and user profiles. Externalizing or offering data or elaboration services to third parties, companies compromise their autonomy, as well as when employees delegate tasks to colleagues or systems. Conditions for collaboration are negotiated on the basis of trust for specific guarantees, such as, security be it confidentiality, reliability, availability, integrity, and others.

The socio-technical structure of complex systems, describes how humans and machines depend one to another, share data and send messages, while act with a certain degree of autonomy. The flow of personal data that emerges is complex, and because of the autonomy of actors, privacy cannot be protected if not by considering the complete organizational structure.

1.0.1 Privacy regulations

Privacy is regulated by specific laws that allow organizations to process user data with limitations that depend on the purpose of the processing. For example, in the EU, organizations are allowed to process employee data for employment purposes, or customer financial data for payment purposes, or customer data for marketing purposes, but only in the case that the user agrees on this. In the EU, privacy is regulated by a specific law, the newborn General Data Protection Regulation (GDPR), while different is in the US, where sector-specific laws regulate privacy in their specific domain, such as, HIPAA which regulates privacy in health-care domain, COPPA which regulates online collection of data of children under 13 years of age, or DPPA which regulates the disclosure of personal data by the Department of Motor Vehicles.

If it is not the case that data can be used because of specific reasons considered by the law, regulations usually allow the use of personal data when the user actively agrees or does not oppose. Differences on this exist between regulations: in the US, where organization can process data without user consent, users have the right to oppose and stop it; different is in the EU, where the GDPR requires user consent to be explicit, informed, and specific, meaning that the user must be informed and must agree before any of his data is collected and processed.

1.1 Motivating scenarios

This Thesis presents and discusses privacy in a scenario offered by the Trentino health-care provider, the "Azienda Provinciale per i Servizi Sanitari" (APSS). The health-care domain, being complex and particularly critical to privacy, provides significant examples to discuss the problem and present the solution proposed in this Thesis.

The evolution of healthcare in recent years, included the shift from a paper-based system toward a digital ones, where patient data are put into electronic records. The high flexibility provided by such systems in accessing the data helps improving the overall performances of the system but also creates privacy problems. In Trento, APSS is responsible for the newborn Italian electronic medical database the "Fascicolo Sanitario Elettronico" (FSE). A significant effort in implementing the FSE, in compliance with privacy regulations, consists in the analysis of procedures and processes where data are used in the system, which are designed from scratch in the case of new systems, and are often undocumented in the case of existing ones.

1.2 Research method

The method presented in this Thesis was designed using the design science approach defined by Hevner et al. in [33]. The design science approach is composed of seven key points that we adopted in this Thesis as follows: (i) *Problem relevance* highlights the importance of the problem addressed; in this Thesis the importance of being compliant with GDPR, and especially with requirements on consent defined in the GDPR, is highlighted in the introduction. (ii) *Research rigor* defines the validity of the baseline; in this Thesis STS, DLV, and BPMN 2.0 are well-know frameworks and a de-facto standard for modeling business processes that offer solid research foundations. (iii) *Design and research processes* delineate the process used to create the method proposed; in this Thesis, the design and validation of method followed an iterative process that involved domain practitioners. (iv) *Artifact* focuses on the outcomes provided by the research; in this Thesis the outcome consists in a tool-supported method with guidelines for the final users. (v) *Design evaluation*; in this Thesis the evaluation consists in the application of the method proposed, using a real case study, performed by domain practitioners. (vi) *Research contribution* defines the contribution and the novelty of the proposal; in this Thesis contributions are presented in the introduction and include a tool-supported process, modeling languages, representations and reasoning frameworks, and evaluations with domain practitioners. (vii) *Research communication* in this PhD Thesis adopts a technical style aimed for an academic audience.

1.3 ProPrivacy

Compliance with privacy regulations cannot be achieved when software components are considered independently without considering the whole socio-technical structure where the software, together with other technical and human components, is part of. Where single components may easily appear to comply with privacy laws, problems may exist in the socio-technical system when considered in its wholeness. The engineering of such a complex system and its adaptation to different contexts, while constantly checking and maintaining privacy compliance, can be supported by a systematic method that is easy to repeat.

This Thesis proposes ProPrivacy, a model-based method for the analysis of requirements given by privacy regulations, specifically on user consent.

Figure 1.1 shows the multi-layers architecture of the method, where privacy and consent requirements are (i) analyzed in a socio-technical model and then (ii) refined into business process models, to (iii) produce implementation specifications that are compliant with privacy regulations. Graphical modeling languages and formal frameworks support the engineers in the analysis of requirements.

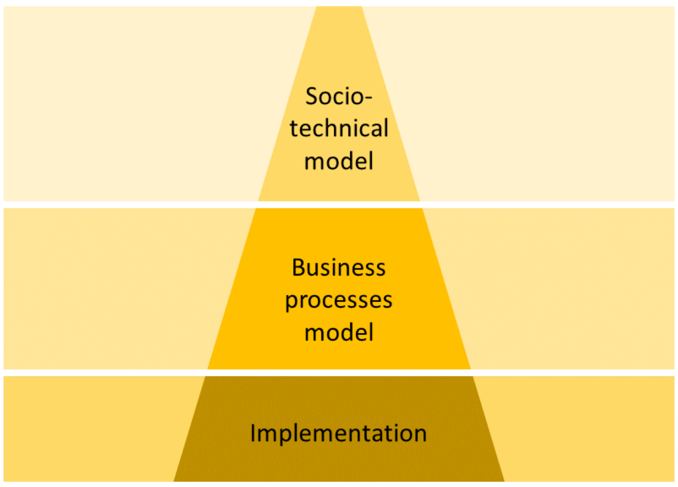


Figure 1.1: Architecture of the ProPrivacy method

1.3.1 Research problem and research questions

The research problem consists in understanding, categorizing, and formalizing the requirements needed to comply with privacy regulations. In the model-based method that we propose, abstractions of the system should include elements to evaluate satisfiability or conflicts of such requirements. This Thesis tries to answer the following research questions:

RQ1 What are the socio-technical aspects of a system that must be considered to comply with privacy regulations?

The socio-technical structure of a system includes aspects about the degree of autonomy of actors in the accomplishment of their goals and their dependency one to another in terms of requirements and resources, including privacy requirements and data resources. Privacy legislation regulates the use and sharing of personal data. To support compliance with privacy regulations, system specifications should include requirements given by regulations, which could be verified only with an analysis of the overall system and not by considering components separately.

RQ2 Given socio-technical aspects regulated by privacy laws, what are the properties of a socio-technical model that verify the compliance with regulations?

Models provide the representation of specific aspects of a system and are used to support their analysis. Modeling languages should be capable of representing aspects of a system that the user may want to analyze. A modeling language that considers privacy regulations can be used to support compliance. To verify the compliance with privacy regulations, the analysis of a socio-technical model should verify specific properties of the system, with respect to what the regulation requires.

RQ3 How to represent, graphically and formally, privacy and consent requirements?

Graphical models ease the analysis of requirements in the design phase, but requirements must be clear and easily understandable as well as the way they conflict one to another and the limits they impose on the system. Reasoning frameworks support users in the analysis of system specifications, which can be hard in the case of complex systems, by providing a formal representation of the model which guarantees specific properties. Automated analysis of models can identify inconsistencies in the system specifications and problems with respect to the properties that the system should verify.

RQ4 Who are target users of the method and what process should they follow?

Modeling languages alone are useless, if no further indications are provided on the use of the models. A process provides users with step-by-step instructions on the use of the modeling languages, models, reasoning frameworks and software supporting tools.

1.3.2 Research contributions

This Thesis proposes a method for the design of socio-technical systems in compliance with privacy regulations. Research contributions include:

- A tool-supported process, to guide analysts in the design of a new system or in its re-engineering; it makes the method systematic and repeatable, so that it can be easily applied on different context.
- Modeling languages, to represent privacy and consent requirements with respect to socio-technical settings and procedures or business processes. Modeling languages are used to represent the system and its requirement on graphical models. The contribution of this Thesis for the modeling languages consists in the extension of STS-ml and SecBPMN2 with elements for the representation of consent.
- A reasoning framework, to automate the analysis of the models, including the verification of built-in GDPR principles. The contribution of the Thesis in the reasoning framework consists in the formalization of the languages and principles of the GDPR, in set theory and an implementation in disjunctive logic.
- An evaluations of the method with domain practitioners in the medical domain, to show how the method can actually supports analysts in their tasks; it is based on an iterative process for the development of the method.

- A framework for consent evolution, that companies and users can use to reason on accessibility of data, including its evaluations in realistic scenarios. The framework consists in a formalization in Description Logics (DLs) that includes consent acquisition and revocation with and without retroactivity, and data collection, use, and sharing; the logical model can be verified for consistency to detect invalid accesses of the data.

1.4 Organization of the Thesis

This Thesis presents the results of the research work done by the PhD candidate, part of which is already published. Each of the works that composes the Thesis provides a component of the privacy framework proposed in the Thesis itself. Chapters match published or still unpublished papers, with small corrections and a slightly revised structure to better integrate with the overall organization of the Thesis.

The Thesis is structured as follows. Chapter 2 presents the state of the art in privacy regulations and consent, trust and certifications, and security engineering methods. Chapter 3 discusses the baseline, BPMN 2.0, STS and SecBPMN2, and DLV, followed by a motivating case study on the Trentino health-care provider. Chapter 4 introduces the method, including the process and the software supporting-tool. Chapter 5 discusses the goal-based modeling and checking of consent, from publication [60]. Chapter 6 presents the refinement of consent on business processes, from publication [59]. Chapter 7 presents how to maintain compliance under evolving consent requirements, from publication [57, 58]. Chapter 8 concludes the Thesis with a discussion on the results obtained and future work.

1.5 Published papers

In 2017, ealy results on this research have been presented at PoEM in Leuven in “Marco Robol, Mattia Salnitri, and Paolo Giorgini. Toward GDPR-compliant socio-technical systems: modeling language and reasoning framework. In IFIP Working Conference on The Practice of Enterprise Modeling, pages 236-250. Springer, 2017” [61]. This work includes early ideas for the socio-technical modeling language and the reasoning framework proposed in this Thesis.

In 2018, a major contribution on goal-oriented consent modeling have been presented at PoEM in Wien in “Marco Robol, Elda Paja, Mattia Salnitri, and Paolo Giorgini. Modeling and reasoning about privacy-consent requirements. In IFIP Working Conference on The Practice of Enterprise Modeling, pages 238-254. Springer, 2018” [60]. This work, with small corrections, composes Chapter 5 of this Thesis, which includes the socio-technical modeling language, the reasoning framework, and validation.

In 2019, we started a work for the verification of consent on business processes, validated with APSS practitioners. Authors of this work are Marco Robol, Mattia Salnitri, Elda Paja, and Paolo Giorgini. We are planning to submit soon this work at Business Process Management Journal (BPM Journal). This work composes Chapter 6 of this Thesis, which includes the representation of consent on business process diagrams, a reasoning framework to analyze processes and consent with respect to privacy regulation principles, and validations of the work.

In 2019, new ideas on consent evolution have been presented at RE in track RE@Next! in South Korea in “Marco Robol, Travis D Breaux, Elda Paja, and Paolo Giorgini. Consent verification under evolving privacy policies. In 2019 IEEE 27th International Requirements Engineering Conference (RE), pages 422427. IEEE, 2019” [57]. A first version of a journal article which includes substantial new contribution composes Chapter 7 of this Thesis, including reasoning framework, examples, and validation with realistic scenarios. We are now close to finish a second version of the journal article, which includes a scripting language for the description of the collection and consent logs.

Chapter 2

State of the art

This section presents state of the art with respect to privacy regulations, including the EU GDPR and US laws that regulate privacy. We discuss about trust and mechanisms to increase it, such as, certifications or insurances. Then, we review security engineering approaches to privacy.

2.1 Privacy regulations

In the Universal Declaration of Human Rights, Article 12 states the principle that "No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks,". Data protection regulations enforce this universally recognized right, to protect users from in-discriminated invasion of privacy by organizations, so to create a more trusty and collaborative environment for internet users, credit card consumers, users in the health-care domain, children, drivers and car owners, and other categories of people.

By applying on all organizations, regulations normalize the expectations of users with respect to privacy protection. However, regulations demand to companies the balancing between no data collection and extensive collection and use of data for service personalization or for selling to third parties. For example, organizations can collect consumer data to suggest products of interest and sell data for targeted advertisements, still complying with privacy regulations, simply by adopting specific procedures or technologies, such as, a pop-up asking the user his consent on the use of cookies on web pages.

For the reasons above, compliance with privacy regulations heavily depends on a correct analysis of organization privacy policies, which also creates a feeling of general diffidence among users, when using technologies that are unknown and through which data can be shared and privacy potentially violated.

2.1.1 Privacy regulations in the European Union

GDPR [26] is the new privacy law of the European Union (EU), enforced since May 2018 it replaces the previous EU regulation [55]. The GDPR is based on general principles which include: **Privacy-by-default** to prevent any invasion of privacy if not agreed by the user; **Transparency** to guarantee that the user is always informed about his privacy or eventual violations; **Minimization** of data and of storage to guarantee that privacy is exposed as low as possible; **Accuracy** and **integrity** of the data to limit the sharing of false or modified information by enforcing adequate security measures.

2.1.2 Privacy regulations in the United States

In the US, privacy is regulated by domain-specific laws, such as, the Fair Credit Reporting Act (FCRA) [80], the Gramm-Leach-Bliley Act (GLBA) [79], the Children Online Privacy Protection Act (COPPA) [78], the Driver's Privacy Protection Act (DPPA) [2], and the Health Information Portability and Accountability Act (HIPAA) [77].

In the US, the federal agency on privacy policy and enforcement has been the Federal Trade Commission (FTC) since the seventies, at the time of the FCRA act. With the quick evolution of internet in the nineties, the Federal Trade Commission (FTC) thought that it was not yet the time to create a new legislation for privacy, with the risk to slow the growth of internet. However, since concerns remained about privacy, the FTC recommended legislation to protect children's privacy and proposed the Fair Information Principles (FIPs). Therefore, voluntary consent, formed the basis of an industry self-regulation approach to privacy policies. At that time, the number of companies providing privacy policies started to increase. Then, the FTC proposed two innovative approaches: online trust certifications, and privacy enhancing technologies (PETs).

2.1.3 Privacy Regulations around the world

Privacy regulations around the world does not limit to the EU and US laws. Studies says that currently, to protect information of internet users, a hundred of countries have introduced privacy laws. This includes countries from any of the six continents, consider, for example, The Personal Information Protection and Electronic Documents Act (PIPEDA) from Canada, National Directorate of Personal Data Protection (PDP) from Argentina, Cyber Security Law from China, Personal Data Protection Bill 2018 from India, The Privacy Act 1988 from Australia, Data Privacy Act of 2012 from Philippines, Protection of Personal Information Act (POPI) from South Africa. Moreover, within the US and EU, single states or countries have their own specific privacy acts, consider for example the California Consumer Privacy Act (CCPA) or the BDSG Germany data protection law.

2.1.4 Consent

Consent is a key element adopted by the majority of privacy regulations, including EU GDPR, to put the user in control of his own personal data.

Adopting the principle of privacy-by-default, the GDPR requires that, in the case of no other specific legitimating reasons, such as, contractual necessity, no processing activities can be done without the consent of the user. Article 6(1) says: *"Processing shall be lawful only if and to the extent that at least one of the following applies: (a) the data subject has given consent to the processing of his or her personal data for one or more specific purposes;..."*.

The GDPR defines **consent** in Article 4(11) as: *"any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement"*.

The definition of consent includes conditions for it to be valid, such as, freely given, specific, informed, and explicit. Further details on those conditions are specified in Article 7 **"Conditions for consent"** and then further detailed in Recitals 32, 33, 42, and 43.

Explicit consent requires companies to demonstrate the acquisition of an unambiguous consent from the user. Article 7(1) defines this condition as: *"Where processing is based on consent, the controller shall be able to demonstrate that the data subject has consented to processing of his or her personal data."*. For example, a signed paper version of the consent with an unambiguous, clear, and comprehensible text. Recital 42 further reformulate this as: *"(...)Where processing is based on the data subject's consent, the controller should be able to demonstrate that the data subject has given consent to the processing operation(...)"*.

Informed consent requires the request for consent to be clear and distinguishable from other matters. Recital 42 defines this condition as: *"(...)For consent to be informed, the data subject should be aware at least of the identity of the controller and the purposes of the processing for which the personal data are intended(...)"*.

Freely given requires that users should not be precluded from the possibility to use a service without consenting on unnecessary uses of personal data. Article 7(2) and 7(4) define this condition as: *"If the data subject's consent is given in the context of a written declaration which also concerns other matters, the request for consent shall be presented in a manner which is clearly distinguishable from the other matters, in an intelligible and easily accessible form, using clear and plain language. Any part of such a declaration which constitutes an infringement of this Regulation shall not be binding."*; *"When assessing whether consent is freely given, utmost account shall be taken of whether, inter alia, the performance of a contract,* *including the provision of a service, is conditional on consent to the processing of personal data that is not necessary for the performance of that contract.".* Recital 42 further reformulate this as: *"(...)Consent should not be regarded as freely given if the data subject has no genuine or free choice or is unable to refuse or withdraw consent without detriment.".*

Specific consent requires a well-defined purpose, clearly stated in the privacy notice, where no general consent is allowed. The purpose of the consent should be clearly distinguishable from other matters, so to not force unnecessary disclosure of information for purposes not relevant to the user. Recital 43 defines this condition as: *"presumed not to be freely given if it does not allow separate consent to be given to different personal data processing operations despite it being appropriate in the individual case, or if the performance of a contract, including the provision of a service, is dependent on the consent despite such consent not being necessary for such performance".*

2.2 Trust and certifications

Trust makes people share personal and intimate information about their lives with others. A climate of trust incentivizes collaboration between people and the sharing of valuable and important information with the society. Trust, however, includes also aspects not related with information and privacy, for example, a climate of trust incentivizes consumers in buying goods and products. Trust is influenced by reliability, security, anonymity and its counterpart accountability, and reputation.

Friedman et al. in [27] explore the features of **trust online**. The authors propose a conceptual framework for understanding and support the engineering of trust, with 10 characteristics of online interaction, distinguishing between electronic commerce and electronic personal interactions. Loss of money is the first way of vulnerability to trust violations in e-commerce, which is due to features of the technology that involve security, anonymity, accountability, and reputation. On the other hand, in e-commerce, we are also vulnerable to loss of privacy, due to the technology, which allows companies to easily collect customers information. In personal online interactions, violations of trust make us vulnerable psychologically. In this case anonymity, in opposition to accountability, can either limit the interactions or create opportunities.

2.2.1 Mechanisms to increase trust

In some context, insurances decrease the level of trust demanded by people to use a service, making them inclined to accept a less trustworthy provider. This is particularly true in e-commerce, where only loss of money is involved. However, when the risk is related with a loss of privacy, the effects of insurances may not be as effective as in other cases, and they may depend on personal privacy attitudes.

Certifications, on the other hands, increase the trustworthiness of a service, which, in turn, is likely to increase the trust people put in the service provider. This is particularly true in the case of new companies and in competitive markets, where a standardized level of expectations should be granted in a sort time. The certification mechanism is based on the trust that people put into the certifying authority, which depends on its reputation among users over time. Certifications easily increase trustworthiness of technical aspects by means of specific technologies. For example, a certification on the use of a pseudonimization privacy-enhancing technology, may increase the trust that people put into the technical system. Moreover, certifications also try to increase trustworthiness of legal practices, including consent and privacy policies. With respect to privacy, certifications do not always guarantee the perfect solution for everyone in every situation. For example, depending from the context and from personal decisions, full accountability may be preferred by some users in place of complete anonymity.

Online "trust" certifications

In online context, certifications have been proposed to increase "trust". Certification authorities audit websites and release certificates that are showed to increase user trust. However, studies criticize such authorities, showing that non-certified website are often more trust-able than certified ones. Edelman in [25] discuss about the adverse selection in online "trust" certifications.

ISO standards

The International Organization for Standardization (ISO) updates his International Standards for privacy information management to ISO/IEC 27701. It includes requirements for establishing, implementing, maintaining and improving a Personal Information Management System (PIMS). In particular the standard provides a solution for addressing article 32 of the GDPR about the security of the information. It includes best-practices that organizations can use to manage the risk of privacy violations.

2.2.2 Economics

To protect privacy, legislation came up with the solution of user consent. However, other solutions have been proposed, such as, the use of economics other than legislation to address privacy risk. Under this perspective, privacy becomes a good, that companies try to buy from users, who may be willing to sell in exchange of services or other goods. In an hypothetical privacy market, where regulations only regulate the market but not privacy by itself, users chose whether or not they want to incur into privacy risks. McDonald et al. in [47]debates about the effectiveness of privacy policies and consent, by estimating the cost of reading privacy policies and comparing it to actual returned value.

2.3 Security Engineering

Different designing frameworks, specific for privacy, have been proposed in the literature. Qingfeng et al. in [32] presents a goal-driven privacy requirements modeling framework to support the design of a Role-Based Access Control (RBAC) system. Kalloniatis et al. in [37] present PriS, a method to support the design of systems starting from privacy requirements. The method elicits privacy goals, analyzes their impact on organizational processes, modeled using privacy-processes patterns, and identify privacy techniques that best support the processes, by using a classification of privacy technologies. Spiekermann et al. in [72] propose an introduction to the privacy domain for engineers, by proposing two privacy design approaches: (i) privacy-by-policy, based on fair information practices; and (ii) privacy-by-architecture, based on data minimization. Gurses et al. in [31], provide an overview of privacy-by-design practices, in particular about data minimization, that are inspired by policy makers but expressed in an engineering perspective. Hoepman et al. in [35] review mains Privacy Enhancing Technologies (PETs) and patterns and propose privacy design strategies to integrate privacy-by-design in the software development life cycle.

2.3.1 Modeling and verification

Methods and frameworks for the engineering of requirements for privacy and security are available in the literature. These methodologies focus on the elicitation of security requirements from business goals. Non Functional Requirements (NFRs) [50] was the first proposal of a comprehensive framework for representing and using non-functional requirements during the development process. KAOS [82] is a method to support the process of deriving requirements, objects and operations assigned to the various agents, starting from high-level goals. It includes a specification language and an elaboration method. $i*$ [43] is a goal-driven modeling language based on actors, goals, tasks, and resources to model dependencies between stakeholders. Tropos [44, 49] is another framework, part of the the i* family, that inherit the concept of softgoal from the NFR framework, to deal with non-functional requirements as a first class concept. GBRAM [7, 8] is a goal-based requirements analysis method that focuses on early elicitation and abstraction of goals from diverse sources.

Automated reasoning

The use of automated reasoning to support the analysis of requirements, was first proposed by Van Lamsweerde et al. in [81]. Giorgini et al. in [30], introduce automated reasoning in the Tropos method to support the identification of requirements conflicts. Giorgini et al. in [29] presents SI*, a security requirement engineering framework that extends *i** [84] to handle security concepts. Dalpiaz et al. in [22] and Paja et al. in [52], propose the STS method and modeling language, a framework for the analysis of security requirements that also includes privacy as confidentiality. Salnitri et al. in [63], [64], and [62], propose SecBPMN2, a method to design secure business processes, starting from requirements analyzed with the STS method and transforming them on business process properties.

Breaux et al. in [15] propose a privacy requirements specification language, called Eddy, with automated reasoning feature. The language provides a set of privacy requirements, on which automated reasoning allows for the detection of conflicts between requirements. Nómos, a software requirement framework, is proposed by Siena et al. in [68, 69] and in its revisited versions [67] and [36], to tackle the problem of compliance of software with the law. It includes a tool-supported (NómosT [85]) modeling language to model norms, as sets of rights and duties, and detect conflicts between requirements. In Nómos 3 [36] the language is extended with legal roles. The framework does not focus on privacy but can be used to model rights and duties of privacy regulations.

2.3.2 Evolution

To the best of our knowledge, this is the first work that tries to understand the effects of policy evolutions on data access in scenarios of multiple consent granting and revocation. We do however discuss related work that study privacy evolution.

Contextual integrity states that information flows conform to norms consisting of a sender, a receiver and who information is about, and was formalized in temporal logics to detect inconsistencies between positive and negative norms [13]. In contrast, data purpose was formalized in DL to verify the purpose limitation and data minimization principles, also called the use and collection limitation principles [17]. In either of these two prior approaches, the issue of granting and revoking consent was not addressed.

Temporality in description logics

Description logics (DLs) are a subset of first order logic, intended as a general-purpose language for knowledge representation, where decidability is valued over expressiveness. The components of description logics are: (i) concepts, (ii) their relations or properties, and (iii) individuals. When using DL to represent an application domain, definitions of concepts and properties compose the TBox, while assertions about individuals and their concepts and properties compose the ABox. Temporal representation is not directly supported by DLs. However, time and temporal concepts can be modeled. The OWL-Time ontology [34] provides concepts related to time representation, however it does not specify how to use these concepts, nor how to reason over such concepts. In general, temporal representation focuses on instants and/or intervals. In a point-based representation, relations between instants are "before", "after", and "equals." In an interval-based representation, relations can get up to the 13 pairwise disjoint Allen's relations [5] showed in Figure 2.1.

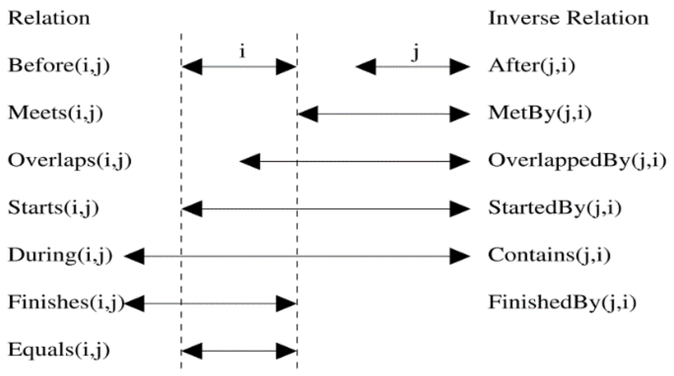


Figure 2.1: Allen's temporal interval relations.

In the case of numerical representation of time, Allen's relations can be easily inferred. For qualitative representation, by assertion of Allen's relations, inferring non-declared relations or checking consistency is an NP-hard problem. Apart from missing representation of time, DL formalisms also miss constructs to represent the evolution of concepts and their properties in time. Because DL only supports binary relations, temporality is not easily encoded as a dimension of an existing relation. For example, a data collection can be expressed as a relation between a data type and a user, however, to include also the time of collection we would need a ternary relation. Many approaches have been proposed to address such problems [9]. Versioning has been discussed in [38], which proposes to create a copy of the ontology at every change. The n-ary relations approach [51] and 4D-fluents [83] are two alternative approaches to represent evolution of concepts. N-ary suggests representing a temporal ternary relation (object, verb, predicate, time) as a concept itself representing the verb, with properties to relate it to the object, the predicate, and the time. The 4D-fluents approach represents temporal relation as a 4-dimensional object, which includes time-specific temporal versions of the object and the predicate, where the original relation is now expressed between the temporal versions of the concepts. With respect to the n-ary approach, 4D-fluents suffers from proliferation of objects (two additional objects for each temporal relation). While n-ary suffers from data redundancy in the case of inverse and symmetric properties (e.g., the inverse of a relation is added explicitly twice). In the representation of consent evolution, collection defines a time instant and consent defines an interval between consent approval and withdrawal. Our approach is based on the representation of non-overlapping time intervals, where data are collected within one of these intervals, while a sequence of intervals defines a consent.

2.3.3 Privacy preferences and user personalization

Recently, companies have pursued advanced personalization of user experiences to strengthen their relationship with users [6, 14, 74]. Personalization is commonly intended as customized or customizable user experiences, based on user's behavior or preferences. Privacy preferences are explicit requests by users about how or when their personal data will be used. For example, users may want to exclude their browser search history from the dataset used by websites to show targeted advertisements.

Ackerman at al. in [3] present the results of a survey on E-commerce privacy concerns. The authors found different attitudes (or preferences) of the respondents with respect to current information practices. They propose privacy clusters for individuals (fundamentalists, pragmatics, and unconcerned) to compare and weight their attitudes (or preferences). Some preferences are shared by all the respondents, such as, automatic transfer and unsolicited communications are disliked, while persistent identifiers are sometimes tolerated. Much of the discomfort with the Web today results from not knowing, or not trusting the information practices of a site [3].

Teltzrow et al. in [76] categorize personalization systems according to the input data they require and then review surveys on privacy concerns, showing that users' privacy concerns have a direct impact on personalization systems. Two approaches are discussed to implement personalization while alleviating the concerns of users, one based on policies and laws, and the other based on anonymity. Spiekermann et al. in [73] discuss about the conflict between the general desire of people to protect their privacy and the companies hope to improve customers retention by service personalization. The contribution consists in an empirical study measures the actual behavior of people versus their stated privacy preferences. The authors categorize approaches to address the privacy issue in three categories: law (the EU model), self-regulation (the US model), and standards. A simple-to-use identity management systems is suggested as an important privacy technology of the future as an alternative to the limitations of anonymity and private credentials.

The P3P [21] is the reference platform for privacy preferences on the web. With P3P, privacy agents are in charge of evaluating website privacy policies with respect to user's privacy preferences. However, preferences are never used to modify websites privacy settings. However, despite the availability of such privacy preferences platforms, Acquisti at al. in [4] shows that people cannot always be counted on to set their own privacy preferences in a self-interested fashion. For the authors the reasons of this are three: (i) people's uncertainty about their privacy behaviors and the consequences of those; (ii) the context-dependency of people's concern about privacy; (iii) and the degree to which privacy concerns are malleable by commercial interests. Because of this, policy approaches that rely exclusively on informing or "empowering" the individual are unlikely to provide adequate privacy protection [4]. In the mobile app world, where something like a privacy platform for preferences is the permissions system offered by the operating system. Lin et al. in [42] investigate the feasibility of identifying a small set of privacy profiles as a way of helping users manage their mobile app privacy preferences.

We reported here a limited selection of works in privacy preferences, most of which are about measuring how people feel and perceive privacy preferences. Together with personalization, these two themes are very important in the research community. Our work is a step forward in understanding how users concerns on privacy, stated preferences, and privacy behaviors may affect the possibility to access user data that is increasingly collected, analyzed, stored, and shared by both online and traditional companies. People do react to opportunities to change consent under different settings, and privacy preferences can give companies more flexibility (e.g., versus withdrawing consent and leaving the service, a user might modify a single preference that only degrades the service quality). Our framework is a first result to help companies understand their opportunities to get more data from the users and increasing their user-base and personalization-quality.

2.3.4 Business processes query languages

Dijkman at al. in [23] review the state of the art in query languages for business processes, that are proposed with the aim of managing repositories containing a large quantity of diagrams. Querying is used for multiple purposes, such as, identifying processes that comply with given standards or features. Existing languages differ because of their expressive power and constructs, which focus either on labels of activities or on control flow aspects.

In opposition, our querying framework for business processes focuses on executions of a given process, and it is not intended for the management of large repositories of business processes. Our language allows to query a process for a specific execution, with the objective to verify control flow constraints. Salnitri et al. in [63] propose SecBPMN-Q, a security-oriented query language for business processes, which, as our allows to express control flow contraints. It is part of the SecBPMN method [64, 62].

Chapter 3

Baseline

The research work described in this thesis is based on the STS method [22], including STS-ml [52] and SecBPMN2.0 [64]. The implementation of the automated reasoning is based on DLV [39, 1] [1], a logical programming language. This section presents BPMN 2.0, the STS method, and the DLV language, then introduces a case study in the medical domain from the biggest health-care provider in Trentino, the APSS.

3.1 BPMN2.0

We introduce here BPMN 2.0 [24], a modeling language for the representation of business processes. BPMN 2.0 provides four type of diagrams, we focus on "collaboration diagrams", which is, by far, the most used part of BPMN 2.0.

A process in BPMN 2.0 is composed by a *start event*, a sequence of *activities*, an *end event*, *data objects* that are read or write by activities, and *gateways* that provide alternatives. In a BPMN 2.0 collaboration diagram, participants collaborate with one another by exchanging messages represented as *message flows*. Participants are categorized into *pool* participants or *swim-lane* participants.

In the graphical representation, activities are white rounded-corners boxes, events are green or red circles, data objects are squared boxes with a flipped corner, gateways are yellow rhombuses, sequence flows are solid arrows, data associations are dashed arrows, and message flows are dashed arrows with a mail envelope.

Figure 3.1 shows an example of a BPMN 2.0 collaboration diagram representing a scenario of the case study, where a doctor visits a patient and sends the referral to a specialist that, in turn, asks an external laboratory to perform blood analysis and creates the diagnosis.

[1]http://www.dlvsystem.com/dlv/

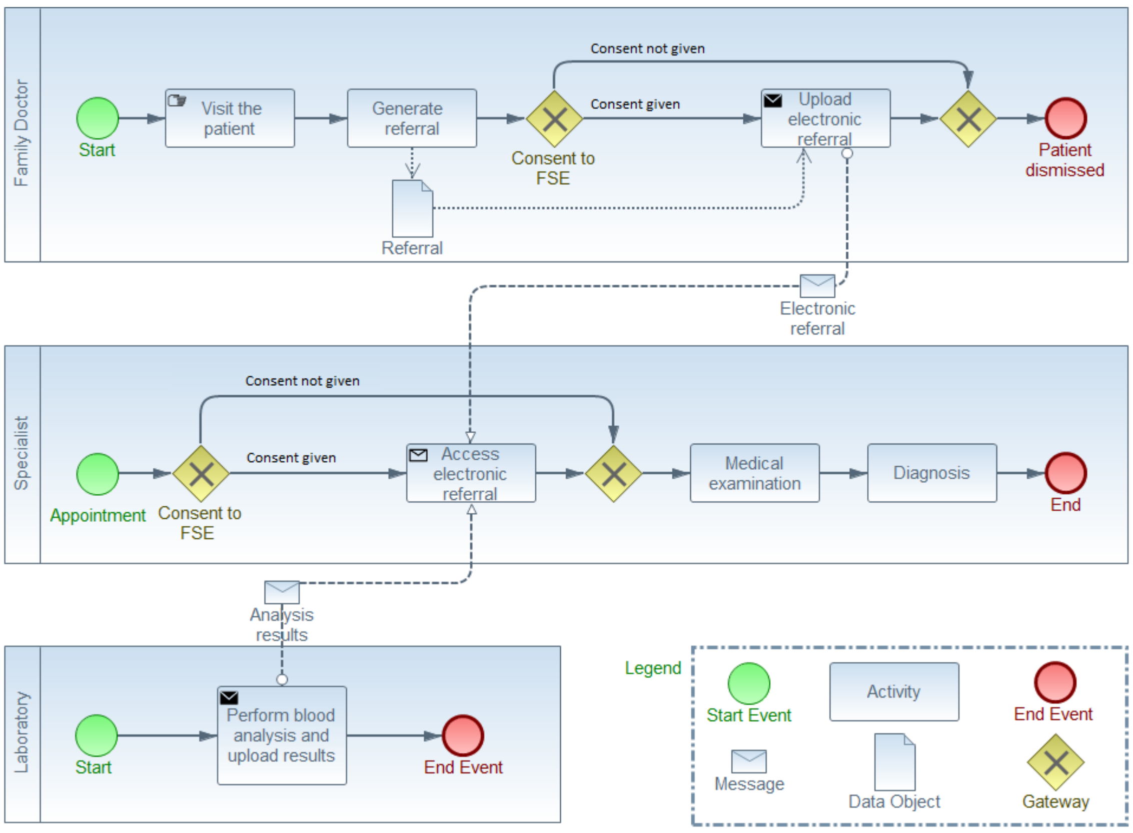
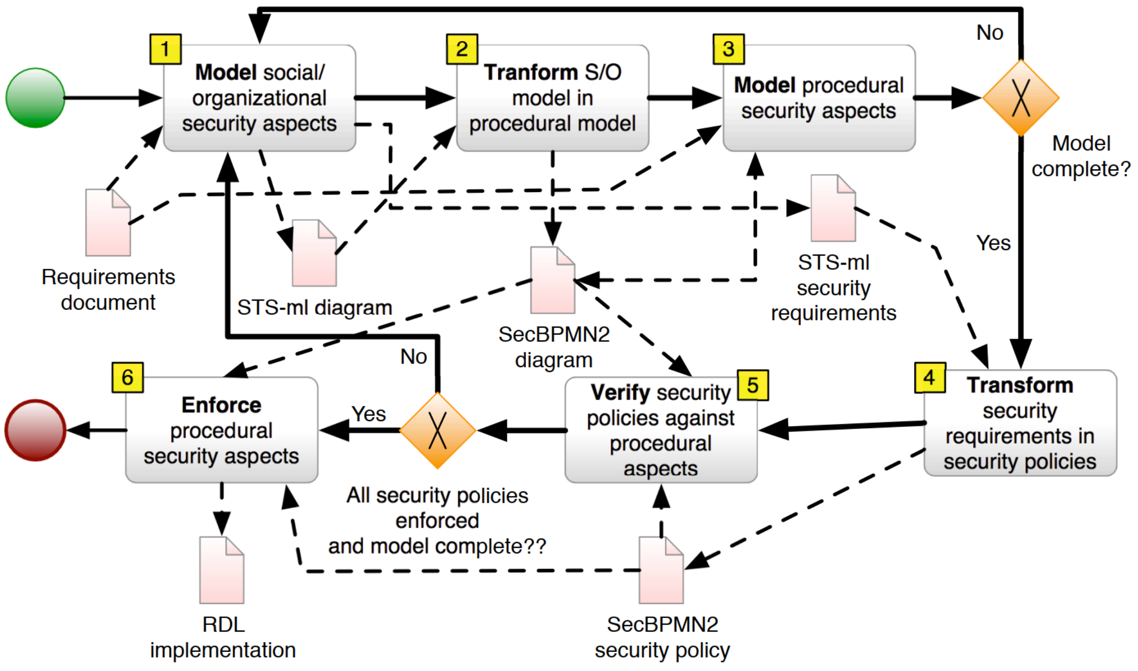


Figure 3.1: Example of a BPMN 2.0 diagram in the medical domain

3.2 STS and SecBPMN2

STS [22] is a security requirement engineering method to support the design of secure systems. STS focuses on social aspects as the main causes of security problems in complex systems. It includes (i) modeling languages to represent the system both at a socio-technical level and at the business process (procedural) level, (ii) an automated reasoning framework, (iii) a supporting tool with automated reasoning capabilities, based on disjunctive logic. Rather than focusing on the single components of a system, STS focuses on the social aspects among all the participants, be they technical machines or human beings.

STS provides a process to support the analysts in the design of a secure system. Figure 3.2 gives an overview of the STS process. In the first phase, (1) analysts collaborate with stakeholders to understand and model the requirements, with the support of automated analyses to check consistency of the model and conflicts between requirements. Then, in a second phase, (2) the STS model is refined in SecBPMN2, with the use of transformation rules to create skeleton processes and policy patterns. (3) Skeleton processes are then modified to include low level details



1
Model social/ organizational security aspects
2
Tranform S/O model in procedural model
3
Model procedural security aspects
No
Model complete?
Yes
Requirements document
STS-ml diagram
SecBPMN2 diagram
STS-ml security requirements
No
6
Enforce procedural security aspects
Yes
All security policies enforced and model complete??
5
Verify security policies against procedural aspects
4
Transform security requirements in security policies
RDL implementation
SecBPMN2 security policy

Figure 3.2: Process of the STS method (including SecBPMN2.0)

not captured in the STS model, and (4) additional patterns are created to express custom policies. (5) Automated reasoning features allow to verify the policies toward the business processes. Iterating on the process allows to constantly checking the model and finally (6) produce secure system specifications. The method can be used either for the design of a new system or for the to re-engineering of an existing one.

3.2.1 STS-ml

STS-ml is the formally defined goal-based modeling language provided by the STS method. It is used to model socio-technical systems as a composition of intentional actors, which represent either a single instance (agent) or a class (role) of either technical components or humans, focusing on actor interactions, i.e. goal delegations and document transmissions. The language is multi-view, so to capture and focus on different aspects of the system separately: **social view** represents actors, goal delegations and document transmission, **information view** represents informative content of documents, while **authorization view** represent security requirements about the reading, modification, production, and transmission of documents.

Figure 3.3 shows an example of an STS model and a legend of STS graphical notation. Examples diagrams of the three views taken from the case study are showed in the left part of the figure.

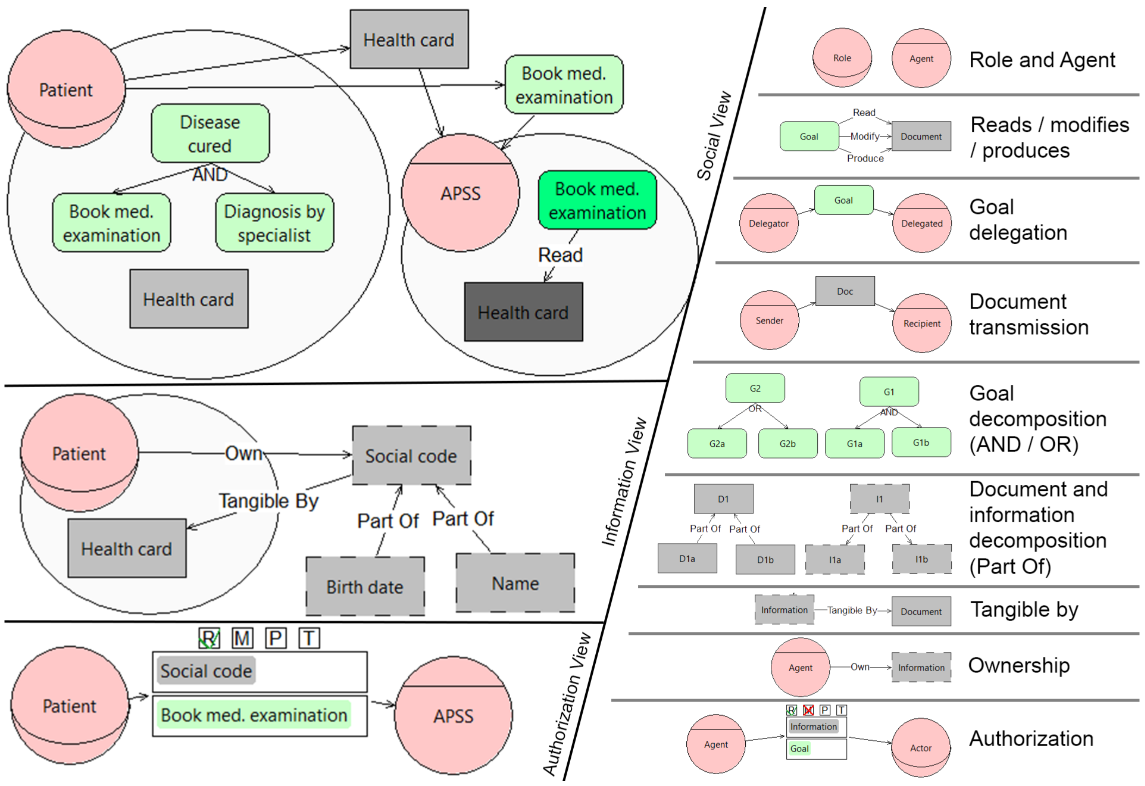


Figure 3.3: STS model

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 Figure 3.3 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. 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 Figure 3.3. 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 Figure 3.3.

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 Figure 3.3, 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 Figure 3.3. Agents can eventually play several roles and roles can be eventually played by several agents.

3.2.2 SecBPMN2

SecBPMN2 [63, 64] is a component of the STS method, based on BPMN2, for the design of secure processes. Starting from an STS model, goals and requirements are refined into business processes with security annotations. With respect to the social modeling phase, where the system is detailed by goals decomposition, in this phase goals are refined into business processes that specify the procedures for the achievement of such goals. Security requirements, that came from the analysis of the social setting, are transformed into business process policies, and verified on the processes with the use of automated reasoning. SecBPMN2 includes two modeling languages: (i) SecBPMN2-ML (Secure Business Process Model and Notation Modeling Language) for the modeling of secure processes, (ii) SecBPMN2-Q for the modeling of process policies. Automated reasoning capabilities allows to verify the policies on the processes.

SecBPMN2-ML

SecBPMN2-ML extends BPMN 2.0 with security annotations.

Figure 3.4 shows an example of a SecBPMN2 model on a payment procedure, which results in a standard BPMN 2.0 diagram if we ignore the three security annotations. The three security requirements that are used in this process are confidentiality, for the transfer request, non-delegation, for the generation of the payment order, and non-repudiation for the reply with the confirmation.

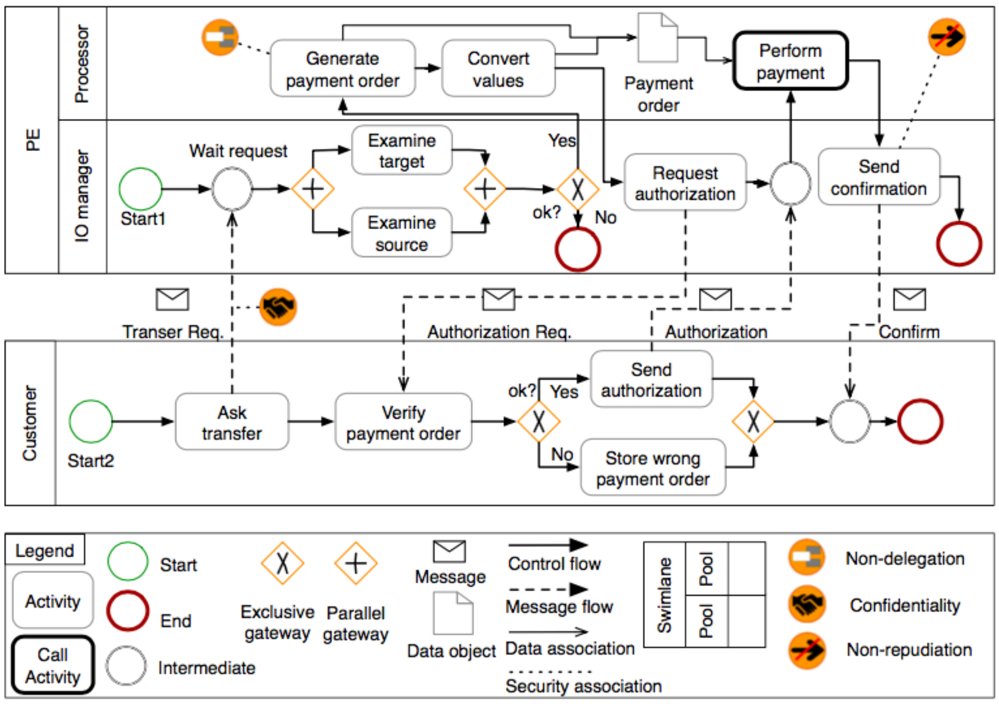


Figure 3.4: Example of a SecBPMN2 model. Figure from [64].

SecBPMN2-Q

SecBPMN2-Q is a query graphical modeling language for business processes. In SecBPMN2 [64], it is proposed to model security policies derived from an STS model with the use of transformation rules, or specified manually by the analyst. Based on the concepts of pattern and anti-pattern, it allows to look for a block of elements in a process that match the ones specified in the query. An automated verification of such policies is based on a formalization of the modeling languages.

Figure 3.5 shows an example of two SecBPMN2 policies, a pattern and an anti-pattern. Elements identified by an @ at the beginning of the name match any element in the queried processes, while elements with a proper name match only the elements with that same name. The pattern policy on the left verify the presence of the confidentiality annotation in a message flow between a ”customer” and a start event named ”wait request” indirectly followed by an activity ”Request authorization”. The anti-pattern policy on the right, verifies that no ”send confirmation” activity indirectly follows a ”convert values” activity.

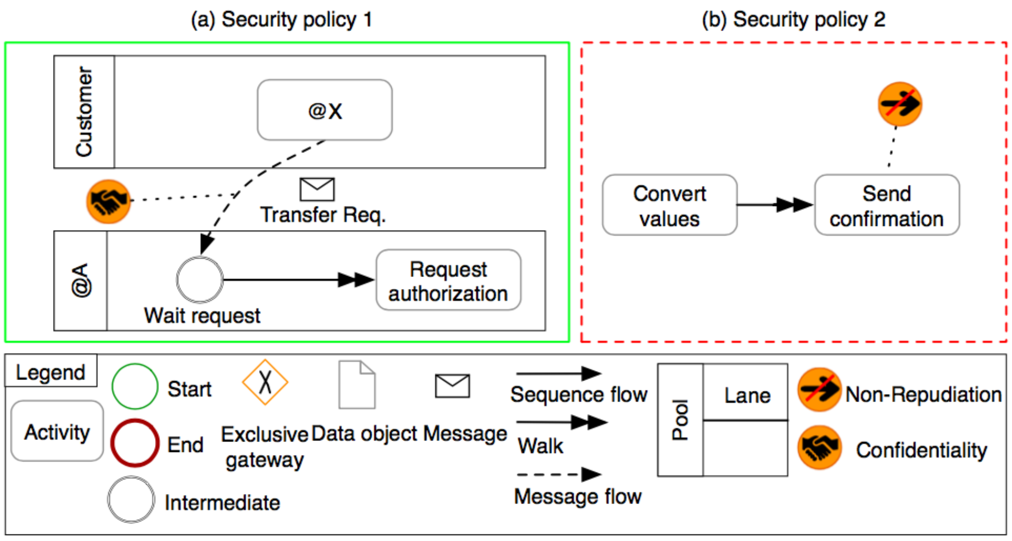


Figure 3.5: Example of a SecBPMN2-Q policy. Figure from [64]

Integration with STS

When SecBPMN is used within STS, mapping rules support the analyst in the refinement of the STS model into SecBPMN2. Actors are mapped to pools participants. Documents are mapped to data objects or messages. Transmissions and delegations are mapped to message flows. Then, security requirements are mapped into policies that are automatically generated by the tool, starting from the requirements in the STS model, with the use of templates.

3.3 DLV

DLV [39] is a disjunctive logic language that allows knowledge representation and reasoning. The DLV engine implements the automated reasoning support for the DLV language. Base elements of the DLV language are simple terms, which can be constants (start with a lower case letter) or variables (start with an upper case letter); predicates express relations and when composed with terms they create atoms. DLV supports explicit negation, written as -, and negation as failure, written as *not*.

Listing 3.1: DLV literals

```
-a(X). not a(X). not ~a(X).
```

A possibly negated atom is a literal. Listing 3.1 shows examples of literals: explicit negation, negation as failure, and negation as failure of an explicit negation.

Listing 3.2: DLV facts

```
a(x1). -a(x1).
```

An explicitly negated literal is a fact. Listing 3.2 shows examples of facts: assertion and negation.

In DLV facts describe the status of the things; facts can be explicitly asserted or inferred by rules. Inference rules are composed by a head which is a disjunction of facts, and a body which is a conjunction of literals separated by the symbol :-; if the body evaluates to true, then the head must evaluate to true as well. Special forms of rules are negation as failure, disjunctive rules, and constraints.

Listing 3.3: DLV rules with negation as failure

```
b :- not a. a :- not b.
```

Negation as failure rules can be used to express alternatives. Listing 3.3 shows an example of such a rule that resolves in either only a being true or only b being true; note that it also admits a solution in which both a and b are true.

Listing 3.4: DLV disjunctive rules

```
a v b :- .
```

Disjunctive rules express alternatives, in an equivalent but more compact form than negation as failure rules. Listing 3.4 shows an example of a disjunctive rule for which either only a is true or only b is true.

Listing 3.5: DLV constraints rules

```
:- rain. :- green(X),red(X).
```

Constraint rules impose conditions for admissible solutions. Listing 3.5 shows examples of constraints for which *rain* is not admissible and nothing can be both green and red.

3.4 Case study: the Trentino health-care provider

When it comes to privacy, the medical domain is one of the most critical. Medical data are highly confidential but at the same time fundamental in performing accurate diagnosis and ultimately saving lives. A large quantity of patient data are produced, recorded and shared between hospitals and other healthcare providers to deliver high quality medical services. Here, a trade-off between privacy and accessibility of data is fundamental. If on one hand there is the need for privacy of patients, on the other, availability of data can be of vital interest.

GDPR is revolutionizing the way of dealing with privacy in several domains, including the medical one. Consequently, organizational processes must be revised and verified, for example, to make sure that consent is asked to the patients before accessing or sharing their medical data.

Healthcare have evolved over years, shifting toward electronic records and digital health, leading to a situation that is dependent to an ever-growing number of complex processes. Unluckily, most of them, from our experience, are usually poorly documented and/or not organized. As a consequence, their analyses require a massive workload that cannot be done in a piecemeal fashion, but rather need to be planned and approached systematically.

User consent, in such a complex scenario, is not solely paper-based but it is mainly electronic-based and integrated in the organization processes. This makes the analysis of requirements related to consent not straightforward, also considering that the health-care system is an evolving complex socio-technical system, where requirements identified at social and organizational level impact on operational processes, such as, accountability of the transmission of medical reports between doctors, impacts on the processes and on technical components.

3.4.1 APSS scenario

We based our case study on the Trentino healthcare service provider, namely *Azienda Provinciale per i Servizi Sanitari* (APSS). APSS is the largest medical organization in the Italian province of Trento [2] and it is integrated in the national healthcare system. It offers direct medical services, in collaboration with external organizations. In particular, we focused on the newly conceived Italian national register of citizens' medical data (aka FSE), which is going to become operative in the near future, and for which, the APPS is working toward implementation in the province of Trento. With the FSE, public and private Italian medical service providers will be all interconnected, giving the possibility to doctors to access patient medical and administrative data from everywhere.

Figure 3.6 shows the social setting of APSS with respect to the management of consent for the FSE in the Santa Chiara Hospital. In orange, the sub-part of the system related with the access of the patient to the health-care system. In red, the sub-system for expressing privacy-preferences and change consent. In green, the sharing of medical documents with APSS professionals. Finally, in blue and purple, technical components of the system, for the management of patient data and consent.

We consider a representative scenario in which a family doctor, an analysis laboratory, and a specialized doctor, share patient's data to provide medical care. Each participant has the patient's consent to access and share his data. These caregiver have each their own system, however they also need to have access to shared systems for sharing medical documents.

We will use this scenario to present the method proposed in this Thesis and evaluate it with domain-experts.

[2] https://www.apss.tn.it/it

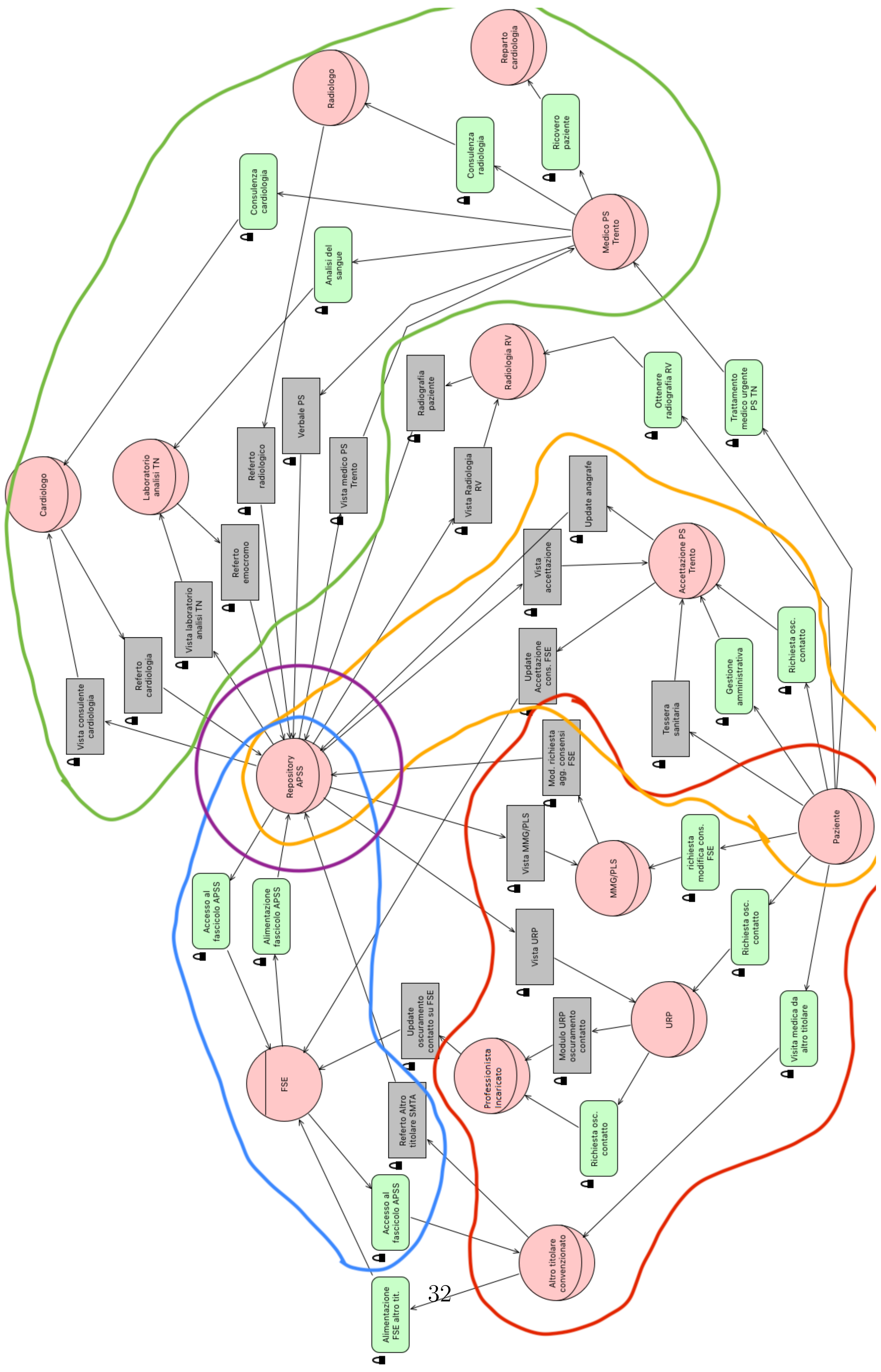


Figure 3.6: Overview of the analysis of APSS social setting, with respect to the management of patient data and consent.

Chapter 4

Consent modeling and verification

This Thesis proposes a method for the analysis of privacy requirements to support the engineering and maintenance of complex systems and their compliance with privacy regulations.

The method adopts a top-down approach, analyzing privacy and consent requirements of a system at different levels of abstractions, focusing first on the socio-technical structure and then moving on procedural aspects. The method, model-based and software supported, supports the analysis of requirements both in the design of new systems or in the maintenance of existing ones.

The method includes a process, two modeling languages, a formal framework and a software supporting tool.

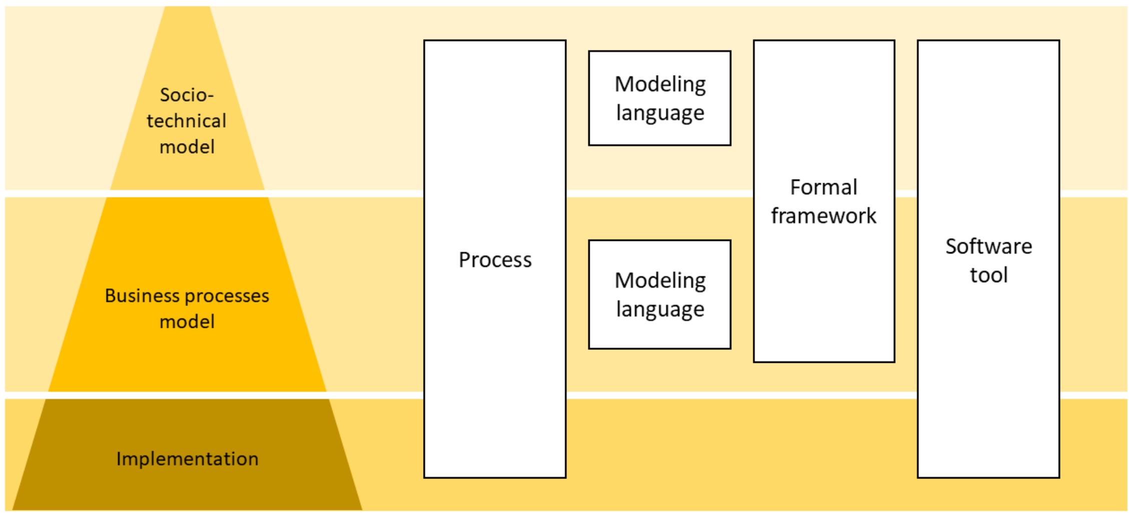


Figure 4.1: Components of the method.

Figure 4.1 gives an overview of the components of the method. The top-down approach adopted by the method is depicted by the pyramidal shape on the left of Figure 4.1. The components of the method are represented on the right part of Figure 4.1.

The process provided by the method span across three layers of abstraction, as in Figure 4.1, from top to bottom: in layer (i) ”socio-technical”, consent is modeled and checked with respect to the organizational structure; then, in layer (ii) ”business process”, consent is modeled and verified with respect to activities and procedures; finally, requirements from the previous steps are used in layer (iii) ”implementation”, to implement the system.

Graphical modeling languages are used to ease the analysis of privacy and consent requirements. The first modeling language is goal-oriented and focuses on the socio-technical structure of an organization. The second one is oriented on business processes and focuses on organizational procedural aspects.

The formal framework includes a formalization of the modeling languages and built-in analyses for the verification of compliance of the requirements analyzed in the graphical models with respect to privacy regulations. Compliance analyses are based on the interpretation of privacy regulations with respect to the abstraction level proposed by the method and represented in the models.

A software supporting-tool guides the analysts in the use of the method. It is provide with automated reasoning capabilities based on the formal framework.

Next sections of this chapter gives more details on the process and on the software supporting-tool.

4.1 Process

The process guides the analysts in the modeling and analysis of privacy requirements.

Figure 4.2 shows the process provided by the method. Rounded boxes, connected by solid arrows, represent the main activities of the process.

The process starts in (1) ”socio-technical modeling”, where organization internal policies are analyzed with respect to the structure of the organization and privacy consent is represented in a socio-technical model; iteratively in (2) ”compliance analysis” the model is analyzed, to check compliance with requirements imposed by privacy regulation; then in (3) requirements of consent are refined into business process models, where the procedures that implement the system are specified; iteratively in (4) ”compliance analysis” business process models are analyzed to verify the compliance with privacy regulations and eventual other requirements from organization policies; the method returns in output requirements compliant with the regulations.

The iteration between the modeling and analysis phases, allow a continuous validation of the models with respect to compliance with regulations.

The method takes in input organization internal policies and privacy regulations, as represented in Figure 4.2 by the squared boxes and dashed arrows. Organization internal policies are used in the modeling of the system, as represented on the left of

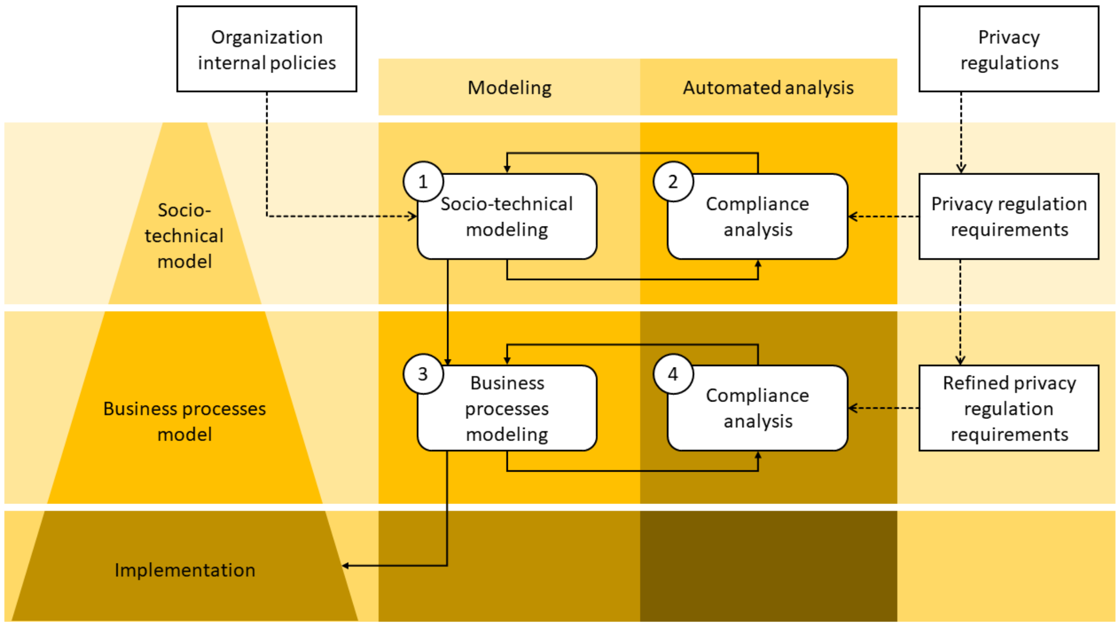


Figure 4.2: ProPrivacy process.

the Figure, where they are analyzed and mapped as authorization requirements and authorization tables. A set of requirements for compliance with privacy regulations have been identified based on our analysis of the regulations. These requirements have been integrated into the formal framework, and are used in the analysis of compliance, as showed on the right of Figure 4.2.

In the case of evolution, when changes are made to the system, for example, for maintenance purposes, the method can be used for its re-engineering. Models of the system are created on the basis of the existing structure and existing models are reused, avoiding excessive overheads of re-modeling. For example, BPMN 2.0 diagrams can be reused by mapping consent requirements on the processes.

4.1.1 Socio-technical modeling

Initially, in order to analyze the requirements of privacy consent, it is necessary to represent in a model the socio-technical structure of the system. The basic elements that characterize such models are the actors, who represent autonomous entities, humans or machines, as also organizations or departments, which share information and delegate goals one another.

We propose a modeling language, based on STS-ml, to analyze the requirements of consent with respect to the organizational structure. It includes the concept of *personal data*, based on the linkability of an information to an actor, that is defined as the *data subject*. Moreover, with respect to STS-ml we have a fourth view, the "consent view", to model *consent* as a set of authorizations to operate on personal data.

Consent authorizations are expressed with respect to *privacy operations*, which are based the taxonomy proposed by Solove in [71]. Specifically, we support authorizations on collection, processing, and dissemination. With respect to the scope of the consent, identified by a set of actors, we speak of: (i) collection in case of data coming from the data subject; (ii) processing in case of data read, modified, or produced, by actors in the scope or transmitted within the scope; and (iii) dissemination in case of data transmitted outside the scope.

Socio-technical compliance analysis

The formal framework in which the language is defined, allows to perform automated analysis on the models. We propose automated analyses of socio-technical models that include well-formedness and compliance with consent requirements from GDPR.

4.1.2 Business processes modeling

Requirements from the socio-technical model are refined onto business processes. This modeling abstraction includes organizational aspects, from the socio-technical level, but adds more details, by focusing on the procedures that implement the system.

The refinement from socio-technical model to business processes is supported by bijective transformation rules. Documents are transformed into data objects, transmissions are transformed into message flows, the use, modification, and production of documents is transformed into dataflows given by data objects taken as input or given as output by activities, and goals are transformed into events.

Consent authorizations are refined on elements of the process into authorization tables, where: data objects are used in place of information; activities are used in place of actors; and data objects or message flows are used in place of document.

Privacy operations of collection, processing, and dissemination are interpreted similarly as in the socio-technical model, more precisely: (i) collection is used in case of message flows from the data subject; (ii) processing in case of data objects in input or output to/from activities or message flows between participants in the consent scope; and (iii) dissemination in case of message flows towards participants outside the scope.

Business processes compliance analysis

The formal framework for automated analysis includes a formal representation of the graphical elements of the business process modeling language. The automated analyses includes the verification of useless authorizations, inter-dependency between separate consents, and the possibility to execute the processes even without providing any authorization. Such analyses are inspired to the GDPR principles of data minimization, freely-given consent, and purpose specificity.

4.2 Supporting software-tool

The method proposed in this Thesis is provided with a supporting software-tool that is based on the STS-Tool [53, 65]. The tool constitutes a fundamental part of the method, guiding experts through all the phases of the process. Specifically, the tool includes modeling editors featuring automated analyses.

Currently, the plugins that extends the tool to support the method proposed in this Thesis are still under development. There are two plugins to extend the socio-technical modeling editor and the business process modeling editor. A version of the tool that fully supports the full process proposed in this Thesis is under development.

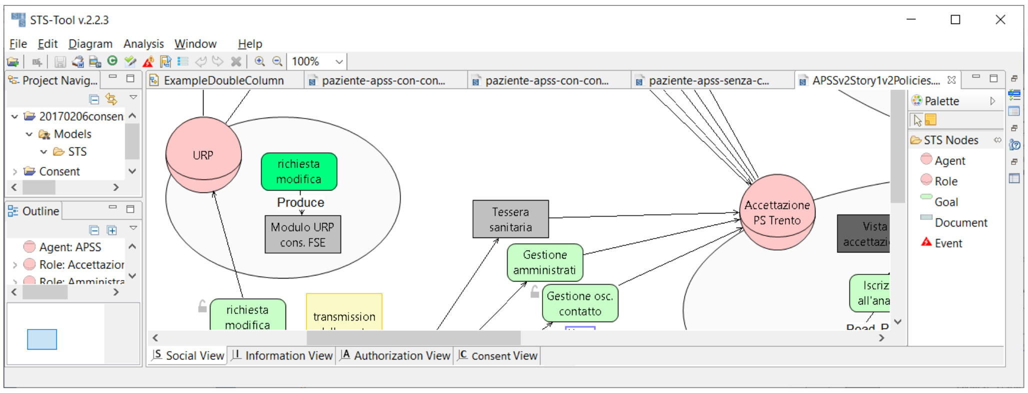


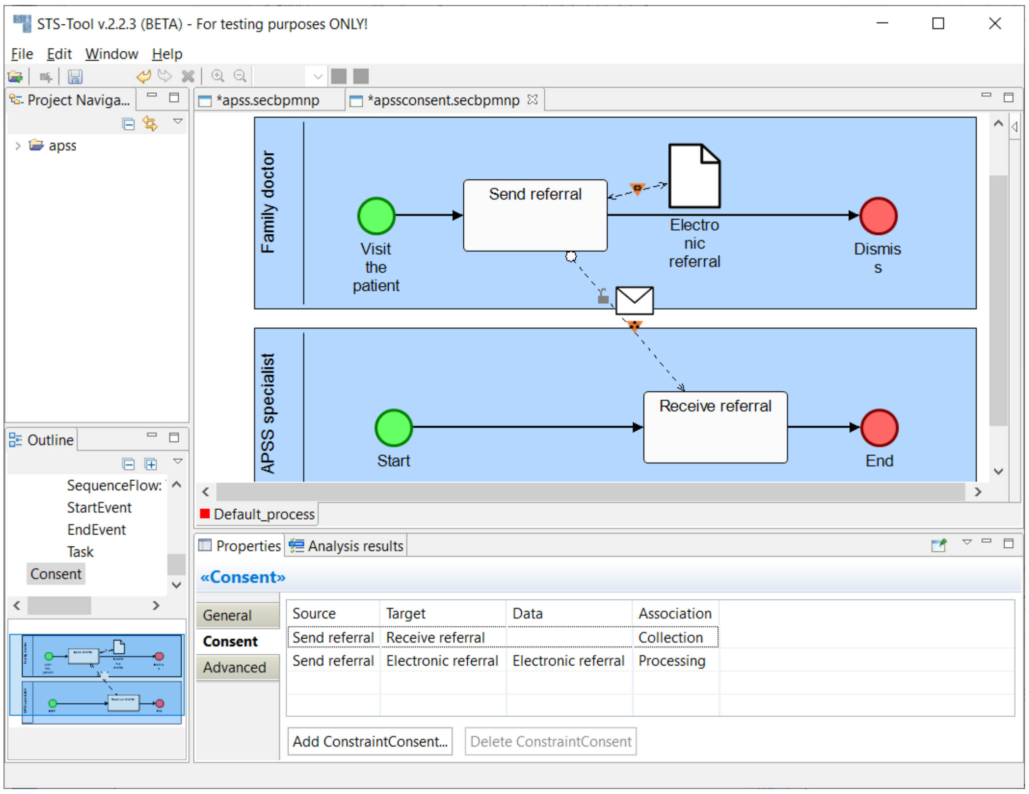
Figure 4.3: Screenshot of the socio-technical modeling editor from the STS-tool

Figure 4.3 shows a screenshot of the socio-technical model editor. The STS editor has been extended to add a fourth view, the consent view, visible in the lower part of the Figure. Currently under development, the editor still misses modeling features for consent and automated analysis.

In the future, the editor will support the modeling of consent authorizations, and the automated analysis proposed in Chapter 5.

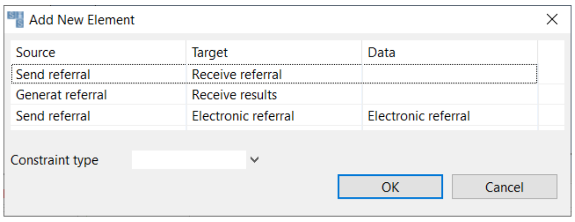
Figure 6.3 shows a screenshot from the business process modeling editor. This editor, based on the SecBPMN2 editor, is extended with consent authorization annotations, as proposed in Chapter 6 by ProPrivacy. The table on the lower part of the Figure represent the consent authorization table, and it is used to automatically annotate the process with consent requirements.

Figure 6.4 shows a screenshot of the window where consent table is modified, by adding or removing authorizations.



Source	Target	Data	Association
Send referral	Receive referral		Collection
Send referral	Electronic referral	Electronic referral	Processing

Figure 4.4: Screenshot from the tool of the ProPrivacy business process editor



Source	Target	Data
Send referral	Receive referral	
Generat referral	Receive results	
Send referral	Electronic referral	Electronic referral

Figure 4.5: Form to modify the consent authorization table from the ProPrivacy tool

Currently, this extended version of the SecBPMN2 editor supports the modification of consent authorization table and the automated annotation of processes on the basis of consent authorizations. In the future, it will also support automated analysis proposed in Chapter 5, for the verification of compliance with respect to GDPR principles.

4.2.1 Architecture of the tool

The tool extends the STS-Tool using its plug-in system, specifically, a plug-in extends the business process modeling editor, while another extends the socio-technical modeling editor with the consent view. The tool is a standalone software written in Java 7, based on Eclipse RCP Framework. It is available for MAC OS, Microsoft Windows, and Linux. The developers that are contributing in this plug-ins are Mauro Poggianella and Daniele Giovanella, following the design specified by Marco Robol (the author of this Thesis), with the help of Elda Paja, Mattia Salnitri and Paolo Giorgini.

The tool offers the following features: (i) a graphical editor for goal oriented and business process oriented modeling languages with privacy requirements; (ii) automated consistency verification of the diagrams; (iii) integration in the STS method that permits to easily shift from goals editor to business process editor; (iv) automated analysis for the verification of compliance with privacy regulations.

4.3 Chapter summary

This chapter presented the method proposed by this Thesis, including an overview of its main components, a description of the process that guides analysts in the adoption of the method, and the supporting software-tool. More details on the components of the methods, for the modeling and analysis of socio-technical setting and business processes, are presented in the next chapters of this Thesis.

Chapter 5

Goal-based consent modeling and checking

The European General Data Protection Regulation (GDPR) [26] has entered into force on May 2018. Compliance is of utmost importance for organizations, in order to avoid monetary penalties which can be up to 20 million Euros. Moreover, public debates on privacy, such as the recent one about personal data being sold away by a well known social network [20], have a strong impact on people, with consequences for organizations that can be even worst than actual fines.

The way how companies do their business is shifting from a traditional closed one, to an approach more open to collaborations with external parties. This way of doing business is supported by *consent*, a key element in privacy regulations that allows the processing and sharing of personal data among organizations, yet it gives users control over their own data. Personal data are stored and shared among organizations by humans through information system. In such complex socio-technical systems, compliance with privacy regulations should be considered starting from social components, down to technical ones.

Compliance with privacy regulations should be handled as soon as possible, considering it as an early requirement, so to avoid unexpected costs of re-engineering [45]. Most importantly, given the relevance of humans in organization activities, the analysis of privacy requirements must carefully represent the social context, the actors involved, their interactions, as well as their expectation and responsibilities in light of privacy regulations.

Methods for the analysis of privacy requirements has been often discussed [32, 37, 15], however, most of these works either does not put social aspects first, or does not take into consideration regulations. Nómos [68, 69] propose a solution for regulatory compliance of software specifications, while other works analyze requirements under a social perspective, such as, $i*$ [84] or Tropos [18], and focus on security as Secure Tropos [29] or STS [22]. In previous work [61], we have presented preliminary results of a framework for privacy requirements, which includes a modeling language based on STS and automated reasoning capabilities.

In this chapter, we propose a method, based on STS, for the analysis of privacy and *consent* requirements, to support compliance with regulations. It is based on (i) the modeling of the domain, (ii) the specification of privacy and *consent* requirements, and (iii) automated reasoning to support compliance with regulations.

The contributions presented in this chapter are the following:

- a modeling language, goal-oriented, focused on privacy and consent;
- A reasoning framework to automate the detection of conflicts between system operations and consent provided by users;
- A validation of the modeling language and the reasoning framework in collaboration with privacy experts in the medical domain, including legal experts, experts in the organization processes, and technical people.

The chapter is structured as follows. Section 5.1 introduces the modeling language for privacy and consent requirements. Section 5.2 discusses the formal framework for automated reasoning. Section 5.3 presents automated reasoning for GDPR requirements checking. Section 5.4 presents the results of the validation, with experts from APSS. Section 5.5 discusses related work, and Section 9 concludes.

5.1 Goal-based modeling language for consent

Graphical models can be used to ease the analysis of requirements both in the design of a new system or in the re-engineering of an existing one. In the analysis of privacy and consent requirements, it is important to model the system so to represent (i) personal data, (ii) data operations, and (iii) privacy consent. We propose a modeling language, based on STS-ml, for the analysis of privacy and consent requirements in complex and evolving socio-technical systems. The new modeling language includes the concepts of (i) *personal data*, based on linkability, (ii) *privacy operations*, based on a taxonomy that includes collection, processing, and disclosure, and (iii) *consent*, based on authorizations. In the rest of the section we go into more details on these aspects, for each of them we present how they are supported by the modeling language.

Our definition of **personal data** is based on the one provided in the GDPR[26]. Here an excerpt from Article 4(1): ”personal data means any information relating to an ... identifiable natural person (‘data subject’); ... one who can be identified, directly or indirectly ...” [26]. The idea behind this definition is about the identifiability of the person in the information. Spiekermann and Cranor, in [72], relate the identifiability of users to **linkability** of data, they talk of personal data in case of linkability and anonymous data in the case of unlinkability. In the model that we propose, information by itself is not directly linkable to a data subject, however, when it is made tangible on a document, this may make it linkable to a data subject. Therefore, we define personal data as an information that is made tangible in a form so that it can be linked to an identifiable user. This design choice was led by the intention of representing the same piece of information under two forms: personal data (when the information is taken from a document linkable to a data subject) and anonymous data (when the information is taken from a document not linkable). We are aware that linkability is actually a very discussed and controversial topic. Deciding and demonstrating the linkability or unlinkability of data is not straightforward and several studies have been done on this, starting from k-anonimity [75] to l-diversity [46] and t-closeness [41]. The process of de-identification of personal information is critical and if not approached correctly, could lead to unwanted and malicious data breaches [28]. We suggest that the lack of linkability, is a property that must be carefully verified. The eventuality that an information could be linkable to the user should be always taken into consideration. In our modeling language, we introduce the *Linkable To* relationship to represent the potential linkability to a *Data Subject* actor, of a *Document*, our tangible form of *Information*.

We investigate on **data operations** that are relevant for privacy, and we propose a classification based on our interpretation of the privacy taxonomy presented in [70]. The taxonomy, in addition to the concept of personal data and data subject [61], is based on the concept of data holder, who is the performer of the following operations: (i) *information collection*, related with the means by which information is gathered from the user by the data holder, (ii) *information processing*, related with the consolidation and use of information and its transfer between information systems by the data holder, (iii) *information dissemination*, related with the disclosure to the public or to another person, (iv) *invasion*, related with intrusion in the private life of the user and interference with his decisions. In our modeling language we include three privacy-relevant operations, namely collection, processing, and dissemination, while we not included invasion since it does not necessary involve information and it is therefore not relevant to information analysis. We speak of *Collection* of personal data in the case of transmission of a document from the data subject, the actor to which the document can be linked to, to another actor. *Processing* of personal data is in the case of any of the operations of reading, modification or production, and also in the case of transmission of documents between actors that are part of the data holder. While *Dissemination* is any transmission made by the data holder, to any other actor that differs from the data holder, and the data subject.

To support **consent** requirements analysis we adopt the definition of consent given in Article 4(11) of the GDPR: ”Consent of the data subject means any freely given, specific, informed and unambiguous indication of the data subject’s wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her". In our interpretation consent in an agreement between two actors, the data subject and the data holder, that consists in the permission for operating on personal data for a specific purpose. For the consent on the processing of personal data, defined in the GDPR, we propose a classification based on [71], that includes consent on the collection, processing (use), and dissemination. In our language we support the modeling of *Consent* as a social relationship between two actors, the *Data Subject* and the *Data Holder*, in terms of authorizations for operating over personal data. The set of actors authorized within a consent defines the *Scope* of the consent, with respect to which we speak of *consent to the collection* in case of authorizations for the transmission of personal data from the data subject to any actor part of the consent scope, *consent to the processing* in case of authorizations to read, modify, produce, or transmit, personal data between actors in the scope, and *consent to the dissemination* in case of authorizations for the transmission of personal data to actors outside consent scope.

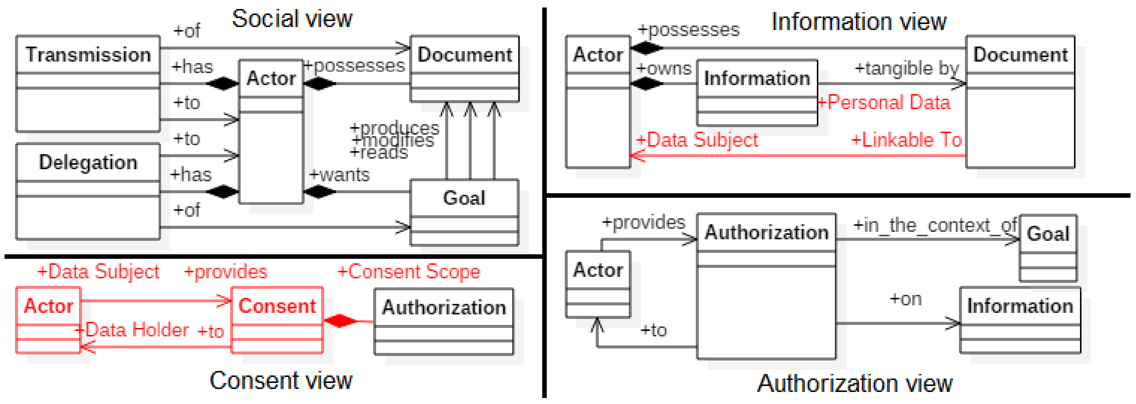


Figure 5.1: Meta-model of the proposed modeling language

Figure 5.1 shows the **meta-model** of the modeling language, in red the elements related to privacy and consent. The meta-model is organized in four diagrams, each representing one of the views proposed in the modeling language. Concepts of the language shared between views are here represented separately in each of the respective meta-model diagrams. The replication of such concepts between the views can be automated by the supporting tool, so to help the modelers in creating consistent diagrams. The modeling language splits across a total of four different views. With respect to STS-ml we added a fourth one, the consent view, to model consent and analyze its requirements. Consent is represented as a relationship between a data subject and a data holder, and consists in a set of authorizations. Permissions specified in this view differs from the ones in the authorization view because: (i) they includes the operations of collection and dissemination and (ii) such permissions are related to a specific consent.

Figure 5.2 shows an excerpt of the **social view**, representing the dependencies

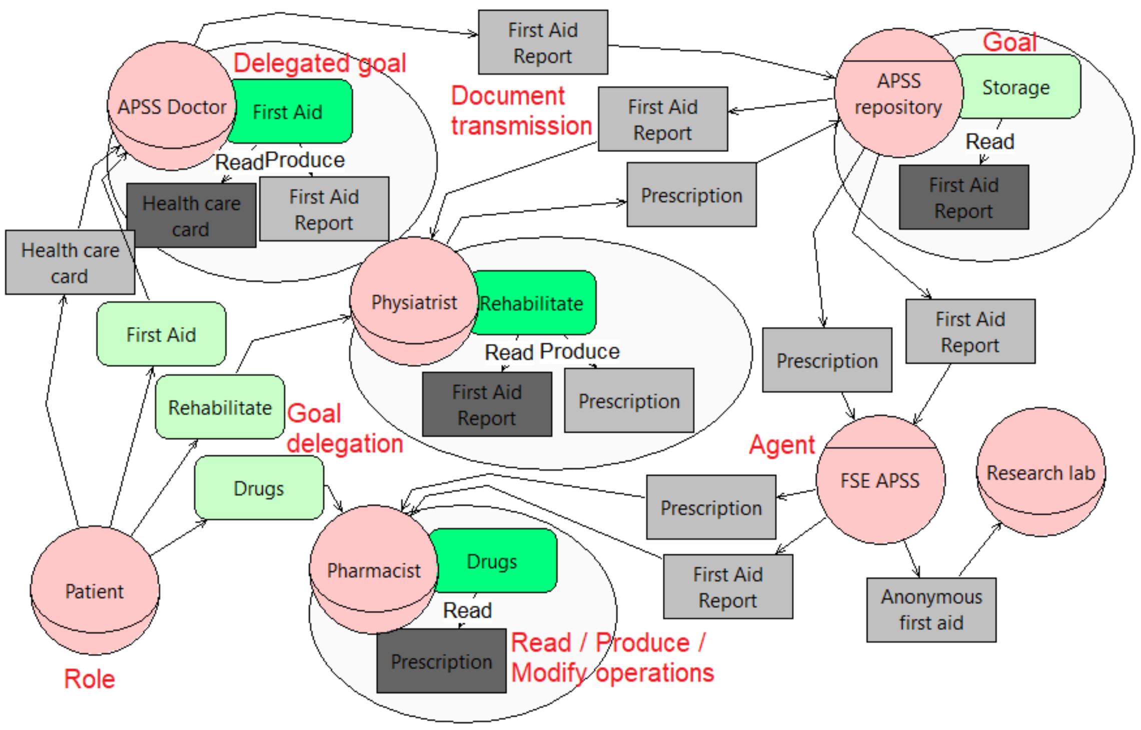


Figure 5.2: APSS social view

between actors, from a model of the case study [1], where a patient interacts with a doctor of the APSS, who provides the first aid, collects his personal data, produces a report, and uploads it to the FSE system, then the patient interacts with a physiatrist for the rehabilitation, who reads the first aid report and produces a prescription, and with a pharmacist, who reads the prescription to provide drugs. A research lab obtains an anonymous version of the first aid report.

Figure 5.3 shows an excerpt of the **information view** from the model of the case study. Patient first aid report is represented as a document, possessed by the actor APSS doctor, such document is *linkable to* the patient himself, meaning that the patient is identifiable. Similarly, the prescription document, that is possessed by the physiatrist, is also linkable to the patient, and the same is for the health care card, also linkable to the patient.

Figure 5.4 shows an excerpt of the **authorization view**, representing the operations permitted by owners of information. In this example, *Pharmacist* is authorized by the *Physiatrist* to read the *Prescription data* in the context of *providing drugs*, while he is not authorized to modify or produce these.

Figure 5.5 shows an excerpt of the consent view from the model of the case study, that includes the details of the three consents provided by the patient to

[1]The complete model can be found at disi.unitn.it/~marco.robol/

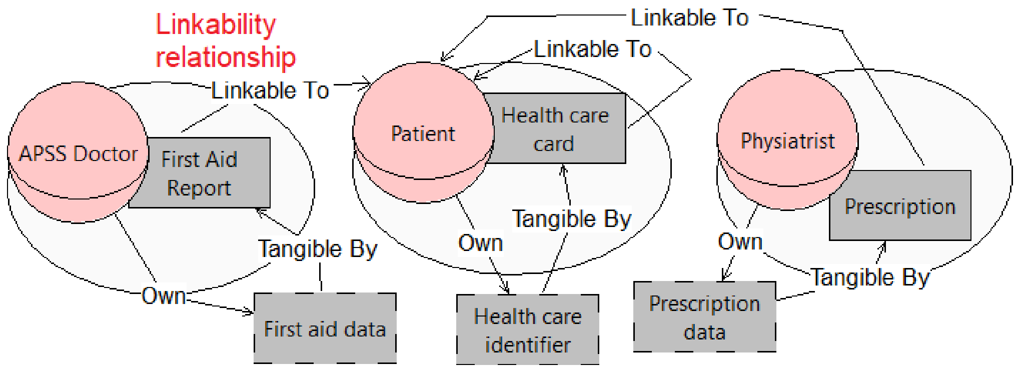


Figure 5.3: APSS information view modified

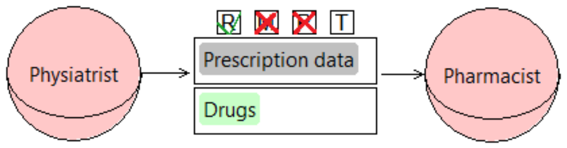


Figure 5.4: APSS authorization view

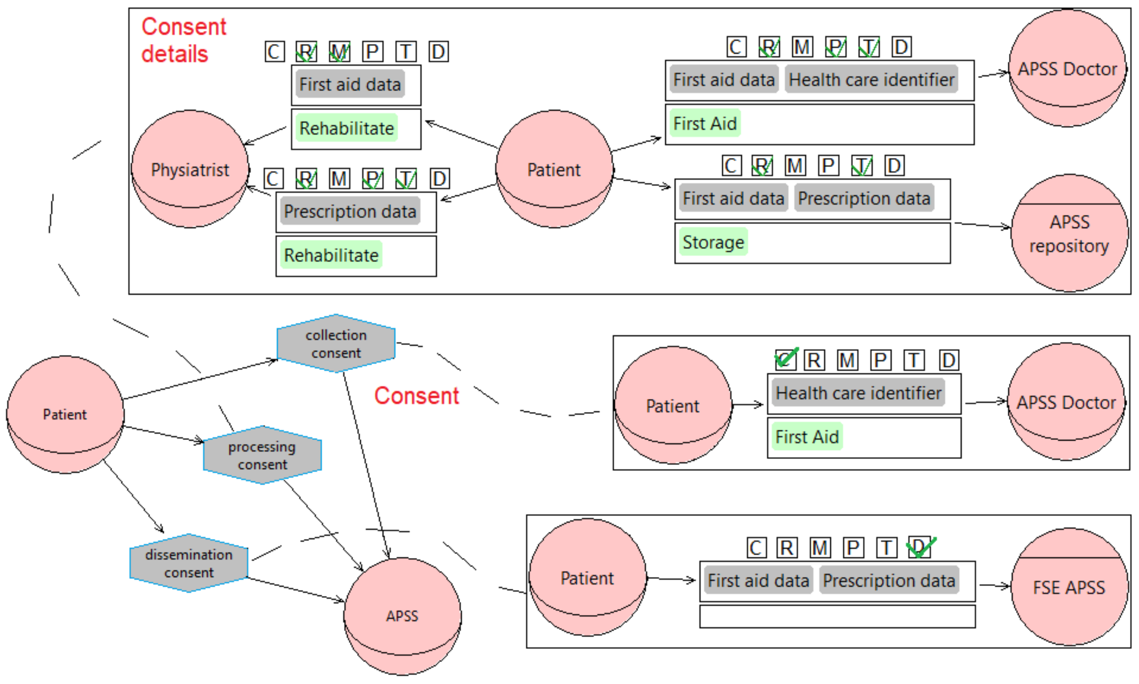


Figure 5.5: APSS consent view

APSS. On the top, details of the processing consent provided to the APSS, which includes in its scope physiatrist, APSS doctor and APSS repository. Letters C, R, M, P, T, and D stands respectively for Collect, Read, Modify, Produce, Transmit, and Disseminate. The APSS doctor is authorized to read, produce and transmit information of the patient within the scope of the consent. Physiatrist is authorized to read and modify first aid data, and read produce and transmit prescription data. On the right, details of the collection consent provided to the APSS, where the APSS doctor is authorized to collect information from the patient. On the bottom, details of the dissemination consent provided to the APSS, which consists in authorizing the FSE in disseminating first aid data and prescription data of the patient to actors outside the scope of this consent.

5.2 Formal framework

Modeling languages should be simple enough to easily identify inconsistencies but, as the models start growing, to adequately represent real world cases, this could became harder [81]. Automated reasoning, based on formal languages, can support users in the identification of potential inconsistencies in the models. The formalization we propose relies on [22, 61], for further supporting consent requirements for privacy.

This section provides a formalization of the language based on the formalization of STS-ml provided in [22]. We use set theory and we define atomic variables with strings in typewriter 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 italics style with a leading non-capital letter (e.g.,*wants*, *possesses*). Table 5.1 lists the predicates used to represent concepts and relationships. We focused on the formalization of concepts related to privacy, such as, consent and personal data. See the information relationship of $\mathit{linkableTo}(\mathsf{D}, \mathsf{A})$ and the social relationship of $\mathit{consents}(\mathsf{A}, \mathsf{A}', \mathsf{S})$ and authorization.

Definition 5.2.1 (Intentional actor)**.** An intentional actor is any agent or role that commits himself in the achievement of a set of goals. An intentional actor model AM is a tuple $\langle \mathsf{A}, \mathcal{G}, \mathcal{D}, \mathcal{IRL} \rangle$, where A is an actor, $\mathcal{G}$ is a set of goals, $\mathcal{D}$ is a set of documents, and $\mathcal{IRL}$ is a set of intentional relationships. An actor model $\mathsf{AM} = \langle \mathsf{A}, \mathcal{G}, \mathcal{D}, \mathcal{IRL} \rangle$ is well-formed if all intentional relationships are defined over actor A, goals in $\mathcal{G}$, and documents in $\mathcal{D}$.

For example, the actor *APSS Doctor* commits himself in the achievement of the goal *First aid provided to the patient*, goal delegated to him by the *Patient*. The model of actor *APSS Doctor* is composed by the goal *First Aid* and the documents *Health Care Card* and *First Aid Report* and the intentional relationships *Read* and *Produce* on the documents.

Definition 5.2.2 (Consent)**.** $\mathit{consents}(\mathsf{A}, \mathsf{A}', \mathcal{A}'', \mathcal{AUTH})$ is a social relationship defined between a data subject A, a data holder A', and a set of actors representing

Table 5.1: Predicates

Concepts: $actor(\mathsf{A})$, $agent(\mathsf{Ag})$, $role(\mathsf{R})$, $goal(\mathsf{G})$, $document(\mathsf{D})$, $information(\mathsf{I})$.
Intentional relationships ($\mathcal{IRL}$): $wants(\mathsf{A}, \mathsf{G})$, $possesses(\mathsf{A}, \mathsf{D})$, $decomposes(\mathsf{A}, \mathsf{G}, \mathcal{S}, \mathsf{DecT})$, where $\mathsf{DecT} \in \{and, or\}$, $reads/modifies/produces(\mathsf{A}, \mathsf{G}, \mathsf{D}, \mathsf{OpT})$, where $\mathsf{OpT} \in \{R, M, P\}$.
Social relationships ($\mathcal{SRL}$): $plays(\mathsf{Ag}, \mathsf{R})$, $delegates(\mathsf{A}, \mathsf{A}', \mathsf{G})$, $transmits(\mathsf{A}, \mathsf{A}', \mathsf{D})$, $authorizes(\mathsf{A}, \mathsf{A}', \mathcal{I}, \mathcal{G}, \mathcal{OP}, \mathsf{Tr})$, $consents(\mathsf{A}, \mathsf{A}', \mathcal{A}'', \mathcal{AUTH})$, where $\mathcal{OP} = (\{C, R, M, P, T, D\} \cup \{\overline{C}, \overline{R}, \overline{M}, \overline{P}, \overline{T}, \overline{D}\})$ and $\mathsf{Tr} \in \{true, false\}$.
Information relationships ($\mathcal{I}_{\mathcal{RL}}$): $owns(\mathsf{A}, \mathsf{I})$, $partOfI(\mathsf{I}_1, \mathsf{I}_2)$, $partOfD(\mathsf{D}_1, \mathsf{D}_2)$, $tangibleBy(\mathsf{I}, \mathsf{D})$, $linkableTo(\mathsf{D}, \mathsf{A})$.

the consent scope $\mathcal{A}''$. Consent consists in a set of authorizations $\mathcal{AUTH}$ provided to actors in the consent scope $\mathcal{A}''$, where authorized privacy operations are expressed with respect to such scope. A $consents(\mathsf{A}, \mathsf{A}', \mathcal{A}'', \mathcal{AUTH})$ is well-formed only if authorizations $\mathcal{AUTH}$ are provided only to actors in the consent scope, represented by the set of actors $\mathcal{A}''$.

For example, considering the Figure 5.5, the *patient*, as data subject, provides the *consent for the collection* of his personal data to the *APSS actor*, the data holder. Such consent consists in permitting the *APSS doctor* to collect from the *patient* the information *health care identifier* in the context of achieving the goal *first aid*.

Definition 5.2.3 (Social model)**.** We bind together actors models and social relationships to compose a social model of the system. A social model SM is a tuple $\langle \mathcal{AM}, \mathcal{SRL}, \mathcal{I}_{\mathcal{RL}} \rangle$ where $\mathcal{AM}$ is a set of intentional actor models, $\mathcal{SRL}$ is a set of social relationships, and $\mathcal{I}_{\mathcal{RL}}$ is a set of information relationships.

Definition 5.2.4 (Authorization closure)**.** We define a closure over authorizations so that if no explicit authorization is provided, any operation on any information is implicitly forbidden. We include here an excerpt of the formal definition as given by Elda et al. in [52]. Let SM be a well-formed social model, the authorization closure over $\mathcal{SRL}$ in SM, denoted as $\triangle_{\mathcal{SRL}}$, is a super-set of $\mathcal{SRL}$ that makes prohibitions explicit, when no authorization is granted by any actor.

Definition 5.2.5 (Consent closure)**.** We define a closure over consents so that if no consent provides an explicit permission to operate on linkable documents, then operating on linkable documents is implicitly forbidden. The complete formal definition as given by Elda et al. can be found in [52].

5.3 Automated reasoning for GDPR requirements checking

In this section, we present our contribution in the automated reasoning, proposed to support analysis related to privacy and consent.

First example of automated reasoning is related to violated authorizations.

Definition 5.3.1 (Violated authorization)**.** An authorization is violated when, even if it makes prohibition to an actor A to operate on an information I, the actor A actually operates on a document D that makes tangible the information I, also considering the operating context G. Formally, for each provided authorization *authorizes*(A, A', $\mathcal{I}$, $\mathcal{G}$, $\mathcal{OP}$, TrAuth), a violation is detected if exists an operation *reads/modifies/produces*(A', G, D, OpT) s.t. $\mathsf{G} \in \mathcal{G}$, OpT is negated in $\mathcal{OP}$, and exists a *tangibleBy*(I, D) s.t. $\mathsf{I} \in \mathcal{I}$.

In the example of Figure 5.2, considering only the authorizations specified in Figure 5.4, different authorizations are violated. The *APSS Doctor* can not read the *Health care card* of the *Patient*, the Physiatrist can not read the *First air report* of the *APSS Doctor*, and also the the *FSE repository* agent and the *FSE APSS* can not read the *First air report*, and they can not transmit any documents.

Consent requirement specifies the need of assessing user decision on the collection, processing, and disclosure of his personal data. We can automatically detect violation of consent reasoning on operations performed on document, and linkability of documents.

Definition 5.3.2 (Violated collection consent)**.** Violation of collection consent happens in the case of transmission of a document D, from actor A' owner of some information I tangible in D to an actor A, if it is the case that the document itself is linkable to A'.

Definition 5.3.3 (Violated processing consent)**.** Violation of processing consent happens in the case of any processing operation on a document D, executed by an actor A', including the transmission toward an actor A'', whether it is the case that the document is linkable to another actor A, owner of some information I tangible in D, and both actors A' and A'' are part of the consent scope.

Definition 5.3.4 (Violated dissemination consent)**.** Violation of dissemination consent happens in the case of transmission of a document D, from an actor A' to an actor A'', if it is the case that the document is linkable to another actor A, owner of some information I tangible in D, and that the transmission is not authorized within the scope of any consent.

In the example of Figure 5.2, considering the consent view in Figure 5.5, some consent are violated. For example, the transmissions of *First Aid data* and *Prescription data*, from the *APSS repository* to the *FSE APSS*, raise a violation of the

Table 5.2: DLV rules for consent

(i) *violatedCollectionConsent*($\mathsf{A}, \mathsf{I}, \mathsf{D}$) :- not *canCollect*($\mathsf{A}, \mathsf{I}, \mathsf{G}$), *transmits*($\mathsf{A}', \mathsf{A}, \mathsf{D}$), *tangibleBy*($\mathsf{I}, \mathsf{D}$), *linkableTo*($\mathsf{D}, \mathsf{A}'$).
(ii) *violatedProcessingConsent*($\mathsf{A}, \mathsf{I}, \mathsf{G}, \mathsf{D}$) :- not *canProcess*($\mathsf{A}, \mathsf{I}, \mathsf{G}$), *reads/mod./prod./tx.*($\mathsf{A}, \mathsf{G}, \mathsf{D}$), *tangibleBy*($\mathsf{I}, \mathsf{D}$), *linkableTo*($\mathsf{D}, \mathsf{A}'$).
(iii) *violatedDisseminationConsent*($\mathsf{A}, \mathsf{I}, \mathsf{D}$) :- not *canDisseminate*($\mathsf{A}, \mathsf{I}, \mathsf{G}$), *transmits*($\mathsf{A}, \mathsf{A}'', \mathsf{D}$), *tangibleBy*($\mathsf{I}, \mathsf{D}$), *linkableTo*($\mathsf{D}, \mathsf{A}'$).
(iv) *excessiveConsent*($\mathsf{A}, \mathsf{I}, \mathsf{G}, \mathsf{OP}$) :- (*canCollect*($\mathsf{A}, \mathsf{I}, \mathsf{G}$), not *transmits*($\mathsf{A}', \mathsf{A}, \mathsf{D}$)) $\vee$ (*canProcess*($\mathsf{A}, \mathsf{I}, \mathsf{G}$), not *reads/mod./prod./tx.*(A, D)) $\vee$ (*canDisseminate*($\mathsf{A}, \mathsf{I}, \mathsf{G}$), not *transmits*($\mathsf{A}, \mathsf{A}'', \mathsf{D}$)), *tangibleBy*($\mathsf{I}, \mathsf{D}$), *linkableTo*($\mathsf{D}, \mathsf{A}'$).

dissemination consent, because dissemination consent is provided only to the agent *FSE APSS.*

We provide automated reasoning for the minimization of personal data, based on the analysis of information used in the achievement of goals and consent. Excessive permissions are provided by consent in the case no such operations are performed.

Definition 5.3.5 (Excessive consent)**.** A consent is excessive when the provided actor A, in the context of achieving any of the goals G in the authorization, does not perform any of the allowed operations on any document D that makes tangible any of the information in the authorization. Formally, an authorization *authorizes*(A, A', $\mathcal{I}$, $\mathcal{G}$, $\mathcal{OP}$, TrAuth) is excessive if given $\mathsf{I} \in \mathcal{I}$, $\mathsf{G} \in \mathcal{G}$, $\mathsf{OpT} \in \mathcal{OP}$, do not exists any *reads/modifies/produces*(A', G, D, OpT), s.t *tangibleBy*(I, D).

For example, in the *processing consent*, represented in Figure 5.5, *APSS doctor* is authorized to transmit the *Health care identifier*, while this is not necessary in the achievement of any goal, as in Figure 5.2.

5.3.1 Core logic implementation

In STS [22], automated reasoning has been implemented in DLV [1], and integrated in the graphical modeling editor tool. Different types of automated analysis are provided with the tool, such as, well-formedness and security analysis. We present here an excerpt of the core logic implementation, while the integration in the modeling tool is still under development.

In Table 5.2 the rules implementing in DLV the following reasoning: (i), (ii), and (iii) identification of violation of consent for the collection, processing, and disclosure, (iv) excessive consent.

5.4 Domain practitioners evaluation

This section presents the formative evaluation based on an iterative interaction process with domain practitioners, which helped us to iteratively refine and improve the method. Research questions that we wanted to address are about the usability and completeness of the modeling language and the utility of the reasoning framework.

The evaluation has been done in a real case study provided by the Trentino health-care provider (APSS), in the context of a research project consisting in the experimentation of the STS method for the certification of processes with the GDPR. The experiment design consists in an iterative process on the activities of refinement and validation of the modeling language and the reasoning framework. This required several interactions with domain experts. People involved had different backgrounds, different perspectives on privacy, and different levels of expertise in modeling languages. They included legal, business, and technical people from the APSS, such as, privacy and legal experts, organization experts, experts in the processes, and APSS system experts.

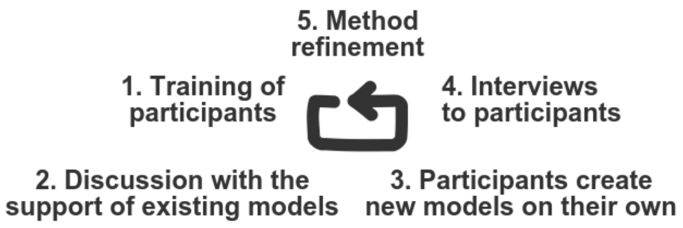

5. Method refinement
1. Training of participants
4. Interviews to participants
2. Discussion with the support of existing models
3. Participants create new models on their own

Figure 5.6: User-Centred Evaluation process

Figure 5.6 shows the steps of each iteration. We first provided a quick introduction on the modeling language, then we discussed with participants with the support of models provided by us, then we let them use the language on their own to produce new models, finally we collected their opinions with respect to the research questions.

The first research question is related to the usability of the language and its capability of being understand by non-experts. Second research question is related to the completeness of the modeling language, and the missing concepts in representing the system and constraints imposed by regulations. Third research question is related to the utility of the automated reasoning framework in identifying privacy criticalities and problems. In the following, we discuss the results of the evaluation.

RQ1 Usability

Opinion of the participants was positive with respect to the usability of the modeling language. They were all able to understand the models, that have been successfully used to support the discussions. Some of the participants were also able to modify existing models and produce new ones. The continuous interactions provided us with new ideas on how to refine and improve the language, on the basis of feedback, comments, and suggestions. For example, the consent relationship between actors, and its detailed view in terms of authorizations, was initially spread among the social and authorization view. After some interaction with experiment participants, we have been able to improved the graphical aspects of the language, by introducing the consent view.

RQ2 Completeness

In the evaluation of completeness of the language, we focused on its ability to represent privacy and consent. In the fist iterations, the lack of specific constructs in the language to represent consent was the major critic by the participants. It was still not clear whether consent should be represented as a goal, as a document, or as an authorization. The language was missing proper constructs to model data collection, data processing, and data dissemination. Iteratively, we modified the language to focus on consent, which we finally defined as an agreement between two actors consisting in a set of authorizations for the operations of collection, processing, and dissemination.

RQ3 Utility of the reasoning framework

We evaluate the utility of the reasoning framework in supporting humans in the analysis of privacy and consent. Models produced in the experimentation are composed by many organizations, departments, and technical systems, with a lot of dependencies, a complex structure of information and documents, and a not straightforward specification of security requirements, such as authorizations. It turned out that models were so complex to be as nearly as impossible for non expert users, and still difficult for experts like us, to analyze and reason on them by hands. For this reason, we improved some automated reasoning, in particular for the identification of violated consent on collection, processing, and dissemination. Opinion of the participants was positive with respect to the utility of the newly integrated automated reasoning on consent.

5.5 Discussion

We have proposed a modeling language and a reasoning framework for the analysis of consent and privacy requirements. Challenges were in the interpretation of the regulation, for example in the formalization of the concepts of personal data and consent, and in the definition of a language to support the analysis of compliance.

We proposed a goal-oriented modeling language, a reasoning framework and a first evaluation based on a real case study in the medical domain. Future work includes (i) a detailed analysis and formalization of the requirements of privacy and consent and their integration in the modeling language and in the reasoning framework, (ii) the development of a supporting tool for the modeling and the automation of the analysis and (iii) the inclusion in the modeling language of other concepts related to privacy.

Chapter 6

Refining and verifying consent on business processes

6.1 Introduction

The General Data Protection Regulation (GDPR) [26] sets a new landscape for companies. Being required to implement privacy-by-design, companies now need their technical and organizational infrastructures to be designed around the means for data processing [Art 25: Data protection by design and by default]. Furthermore, the GDPR mandates companies to certify and document the compliance of their infrastructure, products, processes, and services with the regulation itself [Art 42: Certification].

In certifying their compliance with GDPR, companies *verify* their business processes. However, especially when privacy-by-design principles are not followed, processes are often badly documented, and their upgrade for compliance highly depends on a knowledge that is difficult to recover since it is spread among employees and external parties.

Consent is at the basis of many privacy regulations, providing a lawful basis for the processing of user data. Consent is specified by companies in privacy policies and implemented in their business. The GDPR brings new requirements that companies must meet when managing privacy consent, which include ”minimal”, ”freely given”, ”granular”, ”informed”, and ”unambiguous”. These requirements target business processes of organizations, which must be designed and analyzed with respect to consent, to prove the compliance with GRPR and obtain privacy certifications.

Methods for the engineering of privacy has been often discussed [32, 37, 15], however, most of these research works either does not take into consideration regulations, or does not get down at the level of requirements specification for business processes. Nómos [68, 69] proposes a solution for regulatory compliance of software specifications, while other works such as, Secure Tropos [29] or STS [22], which focus on the analysis of requirements in a social perspective, consider only privacy as confidentiality. Access control technologies are proposed to protect user privacy [11, 10, 48, 54], however, consent and its requirements as imposed by regulations are neither directly considered nor discussed.

We propose a method, ProPrivacy, to automate the verification of consent on business processes, with respect to GDPR principles. The method takes in input organizational business processes, organizational privacy policies, and custom requirements specified by the company. Processes are taken in input as BPMN 2.0 diagrams with no additional modifications, to avoid any analysts' overload of manually remodeling exising diagrams. The second input are organization's legal documents where privacy policies are defined, which are manually analyzed by experts, who interpret and encode consent into authorization tables where specific elements of the processes are authorized. Additional requirements by the company are taken as input and encoded into constraints that can be verified on the processes.

ProPrivacy generates BPMN 2.0 diagrams annotated with privacy requirements, that can be verified to comply with GDPR requirements on consent. The diagrams can be used as specification documents for the development of the system, and then for certification purposes, by including them in the documentation that demonstrate the compliance with the GDPR. The method supports the analysis of processes with an automated reasoning framework that can be used to check the consistency of the processes with consent requirements and user-defined constraints with respect to GDPR principles. We provide an implementation of the verification framework, based on DLV, and evaluate its scalability to prove its potentialities in realistic processes. The contributions of this work are:

- *The ProPrivacy method*, that includes a modeling language that extends BPMN 2.0 with privacy concepts, while also allowing to import existing BPMN 2.0 diagrams;
- *A formal framework* that supports the verification of consent on business processes with respect to the GDPR principles of minimization, freely given consent, and granularity; and with respect to custom organizational requirements;
- *An implementation of the automated reasoning* supported by the formal framework to automatically verify consent on processes;
- *An evaluation* of the method and its reasoning capabilities with domain practitioners through a real scenario;
- *A scalability evaluation* of the implemented reasoning framework performed over realistic scenarios.

This chapter is structured as follows. Section 2 presents the requirements of consent introduced by the GDPR, followed by a motivating case study. Section 3 discusses the baseline, consisting of BPMN 2.0 and DLV. Section 4 introduces the method, including the process and a graphical notation with rules to automate the annotation of processes. Section 5 discusses the formal framework for automated reasoning. Section 6 presents the formalization of GDPR properties, namely, minimization, freely given consent, and granularity. Section 7 presents the evaluation results, concerning both the evaluation with practitioners from the healthcare domain and the results of the scalability tests. Section 8 discusses related work, while Section 9 concludes.

6.2 ProPrivacy, a method to support compliance of business processes with GDPR

This section presents ProPrivacy, a method that supports companies for the compliance verification with GDPR. ProPrivacy is specifically built for existing, large business processes for consent management. Indeed, ProPrivacy allows reusing existing BPMN 2.0 diagrams avoiding the overhead of re-modeling processes from scratch. ProPrivacy features a formal framework that supports automated reasoning capabilities for the analysis and verification of consent on business processes.

ProPrivacy presents an iterative and incremental procedural approach, shown in Figure 6.1, that can be used to ensure compliance of business processes, based on continuous refinements guided by automated analysis that helps identifying privacy issues to be considered. The method takes in input an existing business process to be analysed and privacy policies against which the business process is verified.

The main phases of the method are: (i) analysis of privacy policies ; (ii) automated verification of business process against privacy policies; (iii) refinement of the business processes on the basis of the results of automated analysis.

In the following, we provide the details on target users, the inputs and outputs of the method, and the main components that support ProPrivacy's compliance process.

6.2.1 Target users

The method sees two main roles collaborate to comply with privacy regulations, namely that of privacy expert and business process analyst. The former role is highly skilled on privacy legislation while the latter is more prepared on the design of business processes. The ProPrivacy method requires a close collaboration among such experts.

6.2.2 Inputs

The inputs of ProPrivacy, are (i) business processes, (ii) privacy policies.

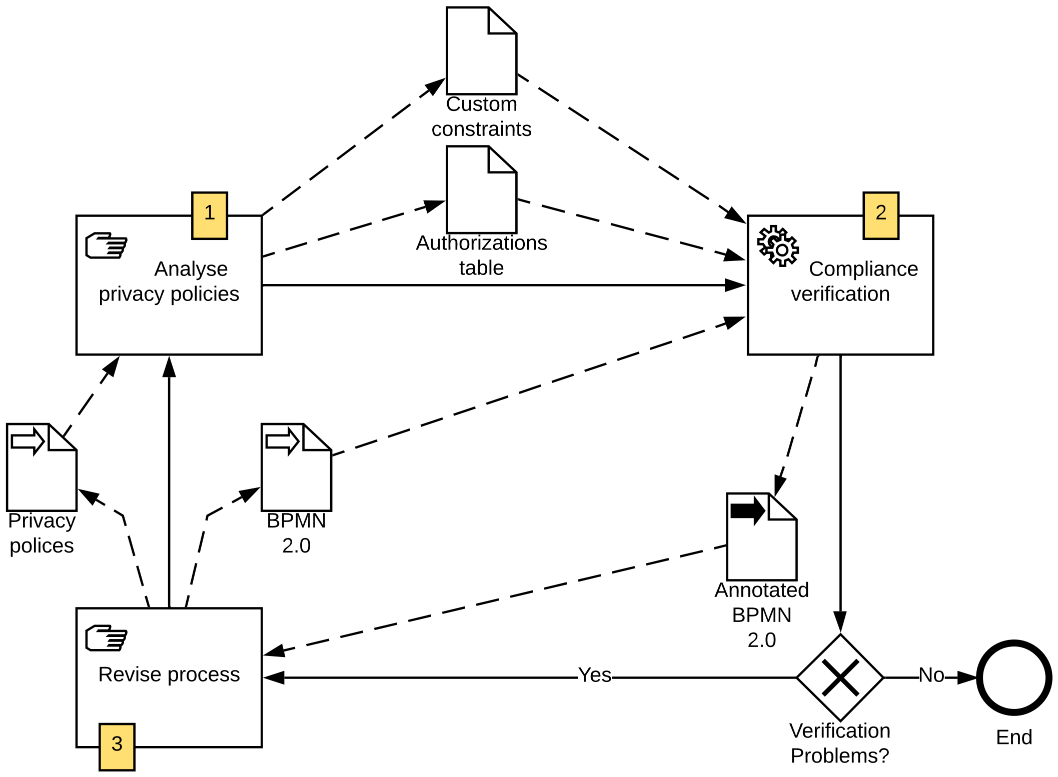


Figure 6.1: The ProPrivacy method

Business processes are taken as input as standard BPMN 2.0 diagrams, without additional annotations. BPMN 2.0 diagrams describe the flowing of data by means of activities that read or produce data objects or that send messages. Additionally, BPMN 2.0 control flow and gateway elements provide alternatives in the execution of processes, therefore describing variants of data flows. The usage of BPMN 2.0 diagrams allows ProPrivacy to perform analysis on data flows with respect to consent and GDPR requirements. Moreover, the usage of a standard modelling language such as BPMN2.0, flatten the learning curve of the method, since it is well-known and widely used.

Privacy policies are documents that describe privacy needs of the users as envisioned and proposed by the company who wants to access on user data. Such documents are elaborated by legal experts of the company and then taken as input by the ProPrivacy method. Its creation, therefore, is out of the scope of this work.

6.2.3 Output

ProPrivacy returns an annotated version of the business process received in input. In this case, we enriched plain BPMN 2.0 with a minimal, self-explanatory, annotations set for the generated diagrams, which are meant to only be read, while experts are not required to use the new modeling language.

In particular, in the case of violations detected by the automated verification, elements that trigger the violation are highlighted. Moreover, consent requirements are graphically represented on business processes in the form of **consent annotations** that are automatically added to the BPMN 2.0 diagrams. This eases the readability of the results of the automated analyses. Three annotations are proposed, one for each authorized operation: collection, processing, and dissemination.

Table 6.1: Annotation rules

Target	Data operation	Icon
`dataobject`	`processing`	▼
`message`	`processing`	▼
`message`	`collection`	▼
`message`	`dissemination`	▼

We define a set of rules, shown in Table 6.1, in order to automate the annotation process on the basis of consent authorizations. The first column in Table 6.1 is the BPMN 2.0 element target of the annotation, the second columns specifies the authorized operation, while the third column is the graphical representation of the annotation.

Figure 6.2 shows a process annotated with authorizations from the user consent in Tables 6.2 and 6.3.

A **supporting software** automatically annotate BPMN 2.0 diagrams starting from privacy requirements, on the basis of the annotation rules.

6.2.4 Phases

We present here the phases of the ProPrivacy iterative approach showed in Figure 6.1, a continuous refinement of business processes based on consent analysis with respect to the GDPR.

In this phase, privacy policy are taken as input and analyzed to extract (i) privacy consents which are reformulated into authorization table, and (ii) custom constraints which are formalized into formal logical rules.

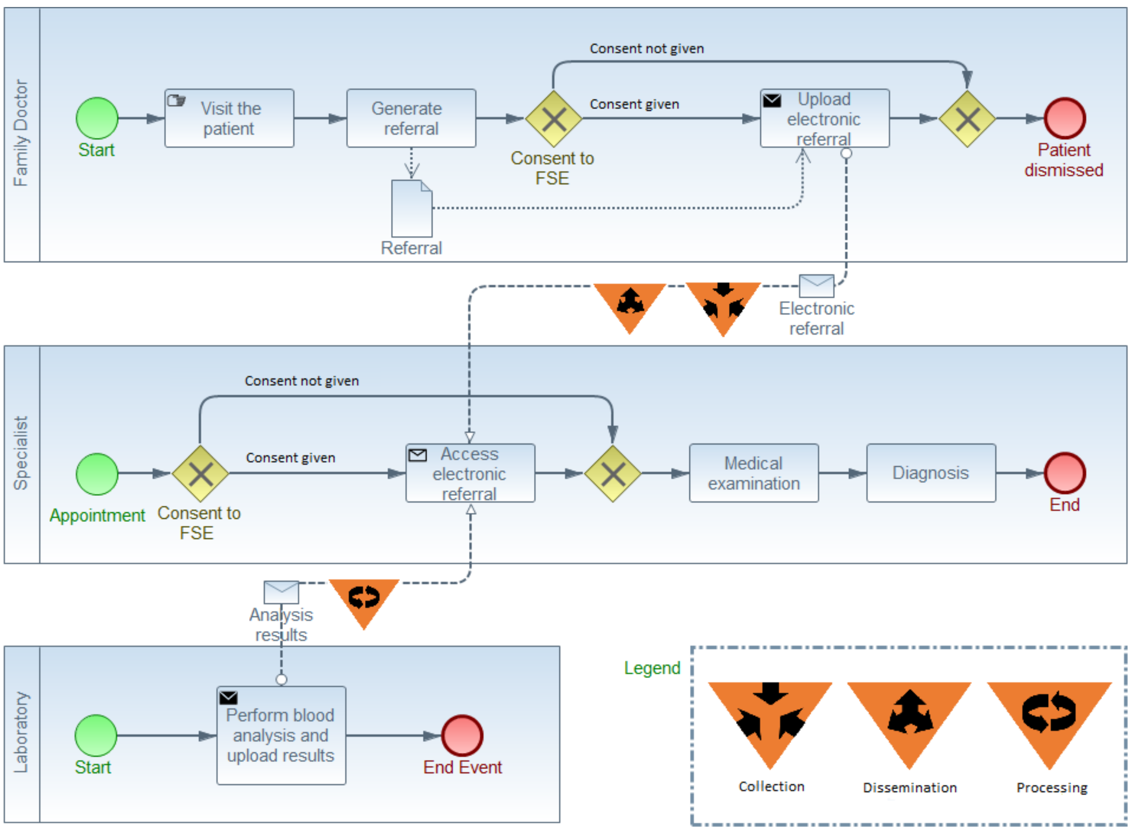


Figure 6.2: BPMN 2.0 with privacy annotations

Phase I: Analyse privacy polices

Privacy consents are sets of authorizations, specified by the users and given to the organization. Such privacy consents are converted into tables of authorizations by privacy experts, where specific elements of the processes are authorized to manipulate sensitive data.

Each authorization is composed of: a data element, an activity element, and a privacy-relevant data operation. We distinguish between three different types of privacy operations, based on the classification from [71]: (i) *collection*, when data are received from the outside of the organization, (ii) *processing*, when data are used and shared within members of the organization, and (iii) *dissemination*, when data are transmitted outside the organization.

Table 6.2: Family Doctor consent example

Participant	Data object	Activity	Operation
Doctor	Electronic Referral	Upload Referral	Dissemination

Table 6.3: Hospital consent example

Participant	Data object	Activity	Operation
Specialist	Electronic Referral	Medical Exam	Collection
Laboratory	Analysis Results	Upload Results	Processing

Tables 6.2 and 6.3 show an example of two user consent represented as authorization tables. In the first one, Doctor is authorized to Disseminate the Electronic Referral in the execution of the activity Upload Referral. In the second one, Specialist is authorized to Collect the Electronic Referral in the execution of the activity Medical Exam, and Laboratory is authorized to Process Analysis Results in the execution of the activity Upload Results.

Custom constraints can be specified by business process analysts, as final users, using a dedicated framework. This is supported in the name of the generalizability of the method to verify other constraints on top of the built-in constraints about consent authorizations. The reasoning framework allows to express custom constraints, such as, traces of execution that may eventually be required or possibly be prohibited, by referring to the elements of BPMN 2.0 diagrams, such as, activities, participants, data objects taken as input by activities or messages flowing between participants.

Phase II: Compliance verification

In this phase, business processes represented into BPMN 2.0 diagrams are taken as input and analyzed with respect to (i) built-in constraints derived from GDPR principles, (ii) artifacts from the previous phase, including authorization table and custom constraints. This activity returns in output an extended version of the BPMN 2.0 received in input, analyzed and annotated with consent requirements.

Built-in constraints are derived by principles of the GDPR. In particular, we focus on constraints derived from GDPR principles of consent *minimization*, *freely given*, and *granularity*. We propose the following interpretation of the above principles, into business process constraints.

- **minimization**: all authorizations of consents gathered by the company, must be used in the process, i.e, the company should ask the minimum amount of authorizations, that is necessary to execute the process.
- **freely given consent**: all business processes should include an alternative execution, in case that no authorizations are given by the user.
- **consent-granularity**: for each authorization that can be granted to a business process, at least one execution trace that needs such authorization (of

that business process), and it does not need authorizations that are part of other consents, must exists.

This phase is supported by a **verification framework**, presented in Section 6.3, that supports the analysis of processes and the verification of built-in GDPR properties, formalized in Section 6.4, and custom constraints, discussed in Section 6.5, respectively.

Phase III: Revise the processes

In order to comply with regulations, processes may need to be reviewed on the basis of automated analysis results. Business process diagrams are modified following the new specifications. In some cases, privacy experts or lawyers may need to modify privacy policies, in the case of changes made to critical parts of the processes.

At this point, the process goes back to phase 1, where policies may need to be reanalyzed if affected by modifications, or the authorization table may need to be updated with reference to the changed business process diagrams.

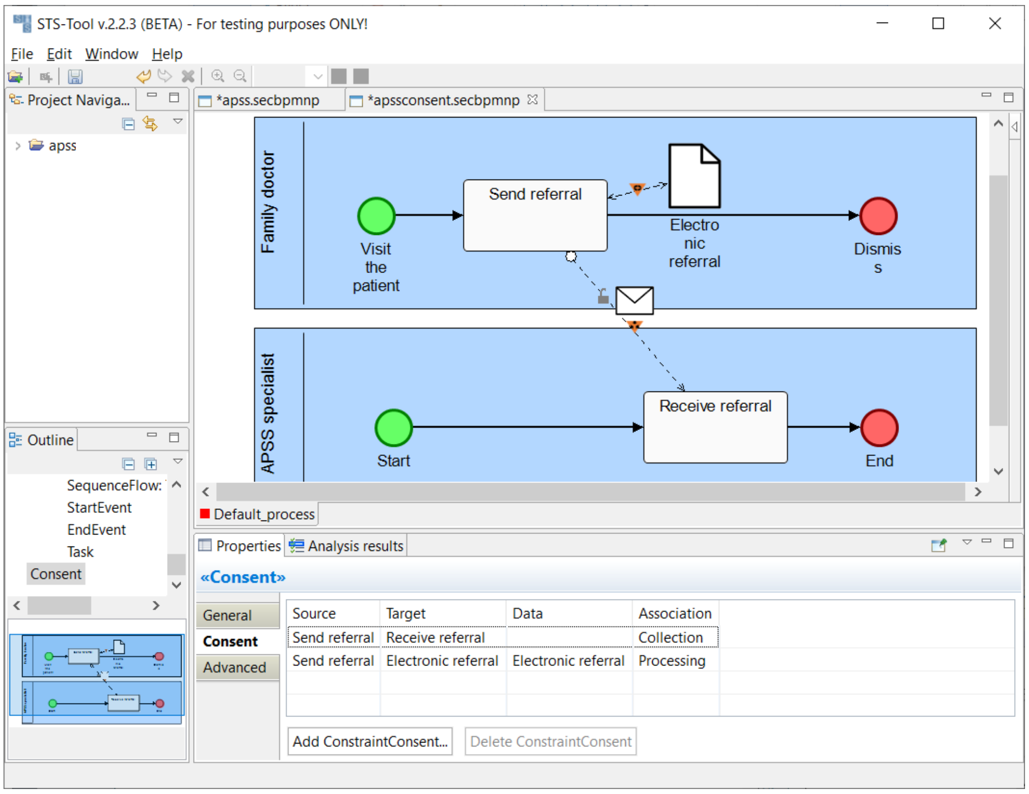
6.2.5 Tool-supported method

ProPrivacy is provided with a supporting software-tool, based on the STS-Tool [53, 65]. The tool constitutes a fundamental part of ProPrivacy, guiding experts through all the phases of the process. Currently, the tool supports (i) the editing of consent into authorization tables and (ii) the automated annotation of processes on the basis of consent. Under development is the integration of automated analysis proposed in the reasoning framework, for the verification of compliance of processes with respect to GDPR principles.

Figure 6.3 shows a screenshot from the business process diagrams editor. This editor, based on the SecBPMN2 editor, is extended with consent requirements annotations. The table on the lower part of the Figure represent the authorization table, used to automatically annotate the process with consent requirements.

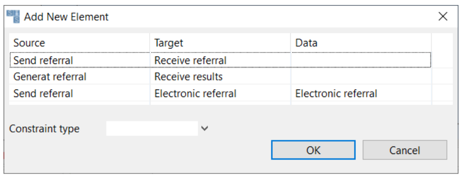
Figure 6.4 shows a screenshot of the pop-up window, where requirements are added on consent, in authorizations table.

The tool extends the STS-Tool by means of a plug-in. The tool is a standalone software written in Java 7, based on Eclipse RCP Framework. It is available for MAC OS, Microsoft Windows, and Linux. The developers that contributed in its development are Mauro Poggianella and Daniele Giovanella.



Source	Target	Data	Association
Send referral	Receive referral		Collection
Send referral	Electronic referral	Electronic referral	Processing

Figure 6.3: Screenshot from the tool of the ProPrivacy business process editor



Source	Target	Data
Send referral	Receive referral	
Generat referral	Receive results	
Send referral	Electronic referral	Electronic referral

Constraint type

Figure 6.4: Form provided within the ProPrivacy tool, for the modeling of the consent authorization requirements

6.3 Formalization of the ProPrivacy modeling language: formal framework

This section presents the formal framework of ProPrivacy, which includes a formal representation of BPMN 2.0 processes, including consent authorization requirements, and a formal definition of BPMN 2.0 execution rules.

We use **set theory** and we define atomic variables with strings in typewriter with a non-leading capital letter (e.g., $\mathtt{p}$, $\mathtt{a}$); sets are defined with strings in the calligraphic font for mathematical expressions with a leading capital letter (e.g., $\mathcal{P}$,

$\mathcal{A}$); relationship are defined in italics style with a leading non-capital letter (e.g., *executor*, *controlFlow*); constants are defined in typewriter font (e.g., `collection`, `processing`). Examples and an eventual DLV implementation of the definitions are provided in Appendix A.

Additionally, we provide a **disjunctive logic implementation** of the verification framework based on DLV. Examples and code excerpts of the DLV implementation of the framework are provided in Appendix A. The full DLV implementation code of ProPrivacy is available online at [1].

DLV predicates are used to represent BPMN 2.0 diagrams, including elements and associations. For example, the predicate *participant* is used to declare a participant, e.g. $participant(FamilyDoctor)$; while the predicate *executor* is used to assert the participant that executes a given activity, e.g. $executor(VisitThePatient, FamilyDoctor)$.

DLV inference rules are used to infer solution models given a BPMN 2.0 process, where a valid solution represent a possible execution of the process. Specifically, in the case of gateways, DLV disjunctive rules are used to generate alternative solution models.

6.3.1 Formalization of BPMN 2.0 syntax

We define a BPMN 2.0 business process and a BPMN 2.0 collaboration process.

Definition 6.3.1 (BPMN 2.0 business process)**.** A BPMN 2.0 business process is a tuple ($\mathcal{A}$, $\mathcal{E}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{P}$, *controlFlow*, *dataAssociation*, *executor*) where:

(a) $\mathcal{A}$ is the set of activities that can be executed by participants;

(b) $\mathcal{E} \subseteq \mathcal{E}^s \cup \mathcal{E}^e \cup \mathcal{E}^i$ is the set of events, external conditions that influence the execution of a process: start, end, and intermediate events specify when a business processes starts, ends, and waits for an external condition to happen;

(c) $\mathcal{G} \subseteq \mathcal{G}^{parallel} \cup \mathcal{G}^{alternative}$ is the set of gateways used to create variants in the flow of execution;

(d) $\mathcal{D}$ is the set of data objects taken as input or given as output by activities;

(e) $\mathcal{P} \subseteq \mathcal{P}^{pool} \cup \mathcal{P}^{swimlane}$ is the set of participants: pools and swimlanes;

(f) $controlFlow \subseteq (\mathcal{A} \cup \mathcal{G} \cup \mathcal{E} \setminus \mathcal{E}^e) \times (\mathcal{A} \cup \mathcal{G} \cup \mathcal{E} \setminus \mathcal{E}^s)$ is the control flow association,

(g) $dataAssociation \subseteq \mathcal{D} \times \mathcal{A} \times \{\texttt{input}, \texttt{output}\}$ is the data association,

[1] Robol, Marco (2020), “ProPrivacy”, Mendeley Data, v1 http://dx.doi.org/10.17632/sm9p7jk383.1

(h) $\mathit{executor} \subseteq \mathcal{P} \times (\mathcal{A} \cup \mathcal{E} \cup \mathcal{G})$ is the executor association.

A BPMN 2.0 business process is well-formed if and only if: (i) there is only one start event $|\mathcal{E}^s| = 1$, (ii) there is only one pool $|\mathcal{P}^{pool}| = 1$, and activities / gateways / events are always executed by some participant $\forall x.x \in \mathcal{AEG} \rightarrow \exists!\, p \in \mathcal{P}^{pool}.(x,p) \in$ *executor*.

Definition 6.3.2 (BPMN 2.0 collaboration process)**.** A BPMN 2.0 collaboration process is a tuple ($\mathcal{BP}$, $\mathcal{M}$, *messageFlow*) where:

(a) $\mathcal{BP} = \{\mathsf{bp}_1 ... \mathsf{bp}_n\}$ is a finite set of business processes in which $\mathsf{bp}_i = (\mathcal{A}_i, \mathcal{E}_i, \mathcal{G}_i, \mathcal{D}_i, \mathcal{P}_i, \mathit{controlFlow}_i, \mathit{dataAssociation}_i, \mathit{executor}_i)$ is the i-th bp in $\mathcal{BP}$ and $1 \leq i \leq n = |\mathcal{BP}|$,

(b) $\mathcal{M}$ is the set of messages exchanged between participants;

(c) $\mathit{messageFlow} \subseteq (\mathcal{A}_i \cup \mathcal{P}_i^{pool}) \times (\mathcal{A}_j \cup \mathcal{P}_j^{pool} \cup \mathcal{E}_j) \times (\mathcal{M})$ is the message flow association, where $i \neq j$ and $1 \leq i \leq |\mathcal{BP}|$ and $1 \leq j \leq |\mathcal{BP}|$.

From now on, based on the definition of a collaboration process, we now redefine sets by including elements from all the business processes in $\mathcal{BP}$.

Given a collaboration process cp=($\mathcal{BP}$, $\mathcal{M}$, *messageFlow*) where ($\mathcal{A}_i$, $\mathcal{E}_i$, $\mathcal{G}_i$, $\mathcal{D}_i$, $\mathcal{P}_i$, $\mathit{controlFlow}_i$, $\mathit{dataAssociation}_i$, $\mathit{executor}_i$) $\in \mathcal{BP}$ and $i \in \mathbb{R}$ s.t. $0 < i < |\mathcal{BP}|$, we define:

(a) $\mathcal{A} = \bigcup_{i=0}^{|\mathcal{BP}|} \mathcal{A}_i$,

(b) $\mathcal{E} = \bigcup_{i=0}^{|\mathcal{BP}|} \mathcal{E}_i$,

(c) $\mathcal{G} = \bigcup_{i=0}^{|\mathcal{BP}|} \mathcal{G}_i$,

(d) $\mathcal{D} = \bigcup_{i=0}^{|\mathcal{BP}|} \mathcal{D}_i$,

(e) $\mathcal{P} = \bigcup_{i=0}^{|\mathcal{BP}|} \mathcal{P}_i$,

(f) $\mathit{controlFlow} = \bigcup_{i=0}^{|\mathcal{BP}|} \mathit{controlFlow}_i$,

(g) $\mathit{dataAssociation} = \bigcup_{i=0}^{|\mathcal{BP}|} \mathit{dataAssociation}_i$,

(h) $\mathit{executor} = \bigcup_{i=0}^{|\mathcal{BP}|} \mathit{executor}_i$.

6.3.2 Formalizing BPMN 2.0 with privacy consent

We define BPMN 2.0 with privacy consent as a collaboration process with authorizations required for the execution of activities. Organizations need the authorizations of the users to use data that are protected by the GDPR. These authorizations are asked in blocks by the company to the user, through proposals made out to users, these set of authorizations compose the user consent.

Definition 6.3.3 (Consent)**.** Consent is a set of authorizations provided by the user to organizations to permit the use of data protected by the GDPR.

Given a collaboration process cp=($\mathcal{BP}$, $\mathcal{M}$, *messageFlow*), defined over the sets of elements $\mathcal{A}$, $\mathcal{E}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{P}$, *controlFlow*, *dataAssociation*, *executor*, we define $\mathcal{C}$ the set of consents, as a set of tuple $\mathsf{c_i}$=$(\mathcal{S}_\mathsf{i}, \mathcal{A}uth_\mathsf{i}), i \in \mathbb{R}$ s.t. $0 < i < |\mathcal{C}|$ where:

(a) $\mathcal{S}_\mathsf{i} \subseteq \mathcal{P}$ is the scope of the consent $\mathsf{c_i}$. The scope defines participants that are targeted by the consent.

(b) $\mathcal{A}uth_\mathsf{i} \subseteq \mathcal{P} \times \mathcal{D}ata \times \mathcal{A} \times \mathcal{OP}$ is the set of authorizations in the consent $\mathsf{c_i}$. An authorization $(\mathsf{Participant}, \mathsf{Data}, \mathsf{Activity}, \mathsf{Operation}) \in \mathcal{A}uth$ specifies a $\mathsf{Participant} \in \mathcal{P}$ that is authorized to operate on $\mathsf{Data} \in \mathcal{D}ata$ in the execution of $\mathsf{Activity} \in \mathcal{A}$, for the $\mathsf{Operation} \in \mathcal{OP}$, where:

    - $(\mathsf{Participant}, \mathsf{Activity}) \in$ *executor*

    - $\mathcal{D}ata = \mathcal{D} \cup \mathcal{M}$

    - $\mathcal{OP} = \{\mathtt{collection}, \mathtt{processing}, \mathtt{dissemination}\}$ $\mathcal{OP}$ is the set of privacy-relevant data operations authorizable, in a consent $\mathsf{c_i}$, to participants in $\mathcal{S}_\mathsf{i}$, where: (i) collection means receiving data from participants not in $\mathcal{S}_\mathsf{i}$, (ii) processing means using data by participants in $\mathcal{S}_\mathsf{i}$, and (iii) dissemination means sending data to participants not in $\mathcal{S}_\mathsf{i}$.

Given the set of consents $\mathcal{C}$ defined over a collaboration process cp, we define $\mathcal{A}uth$ as the union set of authorizations from all consent $\mathsf{c_i} \in \mathcal{C}$, such that $\mathcal{A}uth$=$\bigcup_{i=0}^{|\mathcal{C}|} \mathcal{A}uth_i$.

6.3.3 Formalizing BPMN 2.0 execution rules

We now formalize BPMN 2.0 execution rules, by providing the definitions of business process walks and execution traces. These concepts are needed to express constraints on the processes, as we will explain in Section 7 and 8.

Definition 6.3.4 (Walk)**.** A walk is a sequence of elements $[\mathsf{x}_1, ..., \mathsf{x}_n]$ for which a control flow $(\mathsf{x}_i, \mathsf{x}_{i+1}) \in$ *controlFlow* exists for any element x_i in the walk that has a successor x_{i+1}. Given a business process bp, we define $\mathcal{W}_\mathsf{bp}$ as the set of walks. For example, given the process bp in Figure 6.2, two walks between the elements e^s and e^e exist, one of which is the sequence of elements $[\mathsf{e}^s, \mathsf{g}_1, \mathsf{a}_1, \mathsf{g}_2, \mathsf{e}^e] \in \mathcal{W}_\mathsf{bp}$.

Definition 6.3.5 (Process execution trace)**.** A process execution trace $\mathsf{Trace} \in \mathcal{T}races_{\mathsf{bp}}$ is a sequence of elements $[\mathsf{x}_1, ..., \mathsf{x}_n]$ of a business process bp that represents the order of execution of such elements, for a given execution instance.

Execution traces are generated by applying BPMN 2.0 execution rules. For example, execution rules for an alternative gateway on the activities A and B say that either A or B is executed. We provide a summary of main BPMN 2.0 execution rules:

- Each start event initiates a different execution.
- In the case of an activity, the only control flow in output is taken.
- In the case of an intermediate event, the control flow is taken as soon as the event happens.
- In the case of parallel fork gateways, all control flows are taken and the branches are executed independently of one another, until the join node is reached.
- In the case of parallel join gateways, all incoming control flows need to be reached before taking the next control flow.
- In the case of alternative fork gateways, only one control flow is taken.
- In the case of alternative join gateways, the only control flow in output is taken as soon as the node is reached, without waiting for the other branches.
- The execution stops when an end event is reached.

6.4 Automated reasoning for compliance verification

This section introduces three business process constraints, in terms of the formal framework defined in Section 6.3, derived by the three GDPR principles of: consent minimization, freely given consent, and consent granularity.

Definition 6.4.1 (Minimization)**.** The GDPR data minimization principle is defined in Art.5(1) (c), then detailed in Recital 39: "*Personal data must be adequate, relevant and limited to what is necessary in relation to the purposes for which those data are processed*". The legal interpretation is to limit consents to the minimal set of necessary authorizations. In our formalization, authorizations provided in the consents must be used in the process.

Given a collaboration process cp=($\mathcal{BP}$, $\mathcal{M}$, *messageFlow*), defined over the sets $\mathcal{A}$, $\mathcal{E}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{P}$, *controlFlow*, *dataAssociation*, and *executor*, and given $\mathcal{C}$ the set of privacy consents $\mathsf{c_i}=(\mathcal{S}_\mathsf{i}, \mathcal{A}uth_\mathsf{i}), i \in \mathbb{R}$ s.t. $0 < i < |\mathcal{C}|$, and given $\mathcal{A}uth=\bigcup_{i=0}^{|\mathcal{C}|} \mathcal{A}uth_i$ the set of authorizations from all consents in $\mathcal{C}$, we provide the following rule:

$$
\begin{gathered}
\forall \mathsf{auth} \in \mathcal{A}uth \mid \\
\mathsf{auth} = (\mathsf{participant}, \mathsf{data}, \mathsf{activity}, \mathsf{operation}) \rightarrow \\
\text{(i) } (\mathsf{operation} = \texttt{collection}) \\
\rightarrow \left(\begin{gathered} \exists \mathsf{activity}' \in \mathcal{A} \mid \\ (\mathsf{activity}', \mathsf{activity}, \mathsf{data}) \in \mathit{messageFlow} \end{gathered} \right) \\
\wedge \\
\text{(ii) } (\mathsf{operation} = \texttt{processing}) \\
\rightarrow \left(\left(\begin{gathered} \exists \mathsf{inOut} \in \{\texttt{input}, \texttt{output}\} \mid \\ \exists (\mathsf{data}, \mathsf{activity}, \mathsf{inOut}) \in \mathit{dataAssociation} \end{gathered} \right) \right. \\
\left. \vee \left(\begin{gathered} \exists \mathsf{activity}' \in \mathcal{A} \mid \\ \exists (\mathsf{activity}, \mathsf{activity}', \mathsf{data}) \in \mathit{messageFlow} \end{gathered} \right) \right) \\
\wedge \\
\text{(iii) } (\mathsf{operation} = \texttt{dissemination}) \\
\rightarrow \left(\begin{gathered} \exists \mathsf{activity}' \in \mathcal{A} \mid \\ \exists (\mathsf{activity}, \mathsf{activity}', \mathsf{data}) \in \mathit{messageFlow}; \end{gathered} \right)
\end{gathered}
$$

Given a process and privacy consents, for each authorization in the consents, the authorized activity must be an activity of the given process, executed by the authorized participant, and: (i) if the authorization is on the operation of collection, the activity must receive the message in the authorization; (ii) else if the authorization is on the operation of processing, the activity must take as input or give in output a data object or send or receive a message; (iii) else if the authorization is on the operation of dissemination, the activity must send the authorized message.

Definition 6.4.2 (Freely given consent)**.** GDPR defines freely given consent in Recital 43.2: "Consent is presumed not to be freely given if it does not allow separate consent to be given to different personal data processing operations despite it being appropriate in the individual case, or if the performance of a contract, including the provision of a service, is dependent on the consent despite such consent not being necessary for such performance." For this principle, users should not be precluded from the possibility to use a service without consenting on unnecessary uses of their data.

In our framework, we constraint processes to always provide at least one way so that the process is executable without consent from the user.

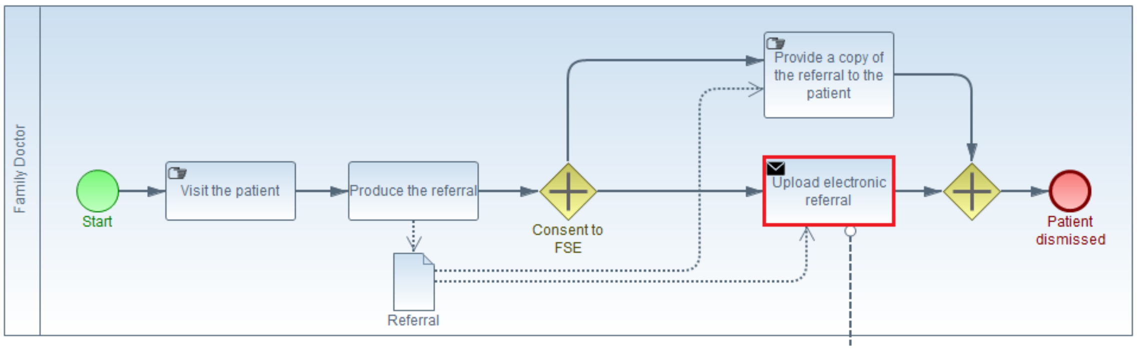


Figure 6.5: Example of violation of the freely given consent principle. The activity in red creates the problem because it requires an authorization to share data with another participant, while the freely given principle expects an alternative possibility to end the process without any authorization.

$$\forall \mathsf{start} \mid \mathsf{start} \in \mathcal{E}^s \rightarrow \exists \mathsf{x}_1, ..., \mathsf{x}_n \mid$$

(i) $$\mathsf{x}_1 = \mathsf{start} \wedge$$

(ii) $$\mathsf{x}_n \in \mathcal{E}^e \wedge$$

$$[\mathsf{x}_1, ..., \mathsf{x}_n] \in \mathcal{T}race_{\mathsf{bp}} \mid \forall i.1 < i < n \rightarrow$$

(iii) $$\nexists \mathsf{participant}, \mathsf{data}, \mathsf{operation} \mid$$

$$(\mathsf{participant}, \mathsf{data}, \mathsf{x}_i, \mathsf{operation}) \in \mathcal{A}uth$$

Formally, for each start event exists at least one valid execution trace, so that it does not include any activity requiring authorizations.

Figure 6.5 shows an example of violation of the freely given consent principle. The branch consisting in providing a copy of the referral to the patient, without uploading its electronic copy, cannot be taken autonomously because of the parallel gateway. An alternative gateway may be used to solve the problem.

Definition 6.4.3 (Consent granularity)**.** The GDPR provides conditions for consent in Article 4. Art.7(4) says: "When assessing whether consent is freely given, utmost account shall be taken of whether, inter alia, the performance of a contract, including the provision of a service, is conditional on consent to the processing of personal data that is not necessary for the performance of that contract." The legal interpretation is that the consent of the user is needed for all the purposes of the intended processing activity.

In our framework, we constrain the process to provide different way of execution for each activity requiring consent of the user, so that no other consents are requiredin the execution.

$$\forall \mathsf{activity} \in \mathcal{A} \mid \exists \begin{pmatrix} \mathsf{participant} \in \mathcal{P}, \\ \mathsf{data} \in \mathcal{D}ata, \\ \mathsf{operation} \in \mathcal{OP}, \\ \mathsf{consent} \in \mathcal{C} \end{pmatrix} \mid$$

$$\begin{pmatrix} \mathsf{consent} = (\mathcal{S}cope, \mathcal{A}uth), \\ (\mathsf{participant}, \mathsf{data}, \mathsf{activity}, \mathsf{operation}) \in \mathcal{A}uth \end{pmatrix}$$

$$\exists \mathsf{trace} \in \mathcal{T}race_{\mathsf{bp}} \mid \mathsf{activity} \in \mathsf{trace} \rightarrow$$

$$\forall \mathsf{activity}' \in \mathcal{A} \mid \mathsf{activity}' \in \mathsf{trace}, \mathsf{activity}' \neq \mathsf{activity}$$

$$\nexists \begin{pmatrix} \mathsf{participant}' \in \mathcal{P}, \\ \mathsf{data}' \in \mathcal{D}ata, \\ \mathsf{operation}' \in \mathcal{OP}, \\ \mathsf{consent}' \in \mathcal{C} \end{pmatrix} \mid$$

$$\begin{pmatrix} \mathsf{consent}' = (\mathcal{S}cope', \mathcal{A}uth'), \\ (\mathsf{participant}', \mathsf{data}', \mathsf{activity}', \mathsf{operation}') \in \mathcal{A}uth' \end{pmatrix}$$

Formally, for each activity requiring a consent, we constrain the process to provide at least one possible way of execution, so that no other consents are required by the execution.

A violation of the granularity principle may happens in the case that two separate consents are required within the same process, where this is designed without taking into account the possibility of having only one consent granted by the user, among the two required.

For example, consider a process for the provision of a medical performance in which: (i) the doctor access to medical data of the patient; and then (ii) he produces a medical report that he uploads on the information system of the hospital where this is by default automatically shared with other doctors. The two activities of (i) accessing patient medical data and (ii) sharing the medical report, require two different consents of the user. In such a case, the consent granularity property is violated. Even if the two activities requires by themselves different purpose-specific consents, de-facto, the process aggregates those consents, thus violating the granularity property.

6.5 Custom requirements

In addition to constraints given by the GDPR, and built-in into the framework, users may have further requirements specific to the domain of application, that may be expressed as custom constraints.

In this sub section, we present an additional component of the ProPrivacy formal framework, used to express requirements which constrain the execution of processes. More in detail, such requirements can constrain the order of execution of process activities, messages, and operations on data objects, all considered as steps of execution of a business process. For example, with custom requirements, the user may specify an order of execution of two tasks and then verify that it is respected in the business processes.

The contribution consists of: (i) DLV rules to generate process execution traces; and (ii) examples on how to express constraints on process executions.

The generation of process execution traces is based on DLV disjunctive rules that implement BPMN 2.0 execution rules to express a DLV problem that resolves to possible process execution traces. The DLV implementation is available in Appendix in Section A.0.7.

Custom requirements on the execution of processes are expressed as DLV constraints rules. For example, in order to force the execution of some specific task, a DLV constraint rule is used to invalidate process execution traces that do not include a step in which the task is executed. Example of custom requirements expressed in DLV are available in Appendix in Section A.0.7.

6.6 Evaluation

This section discusses the results of the application of ProPrivacy method to the heathcare case study, and the results of a scalability evaluation of the implementation of the reasoning framework.

6.6.1 Evaluation with Domain practitioners

The evaluation with domain practitioners has been done in a real case study provided by the Trentino health services provider (APSS), in the context of a research project consisting in the experimentation of the Secure Business Process Model and Notation 2.0 (SecBPMN2.0) method [63, 62] for the certification of processes for the GDPR. With an iterative process we refined and validated ProPrivacy to provide an effective method that can be used in real scenarios. APSS domain practitioners involved in this process had different backgrounds, different perspectives on privacy, and different levels of expertise on business processes. They included legal, business, and technical people, such as, lawyers expert in privacy, privacy executives, information systems engineers and technicians, experts on processes, and finally medical doctors.

Specifically, our interest was into evaluating the ability of the method in identifying privacy problems and promoting discussions between the stakeholders to solve such issues.

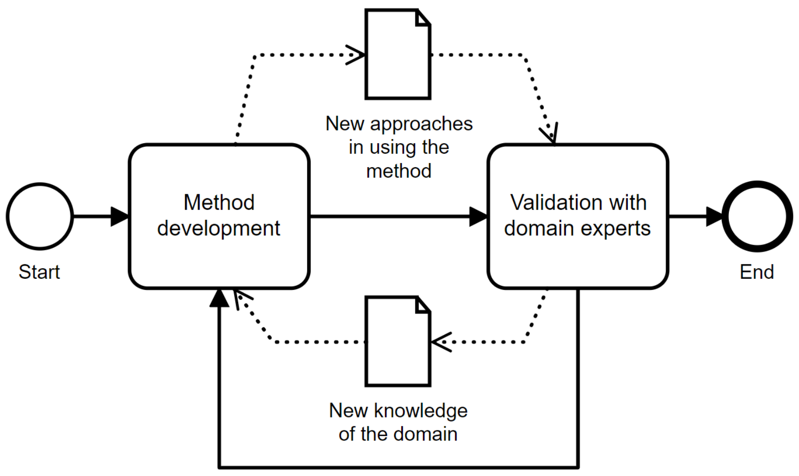

New approaches in using the method
Start
Method development
Validation with domain experts
End
New knowledge of the domain

Figure 6.6: Formative evaluation process

Evaluation process

Since the beginning of the development of the method, we adopted a formative evaluation process that involved domain practitioners from the healthcare domain.

Figure 6.6 shows the iterative process we adopted for the formative evaluation with domain practitioners from APSS. At each iteration with the practitioners we learned insights from the domain, which we presented back to them using diagrams represented in the language proposed by our method. We alternated this activities with research work in which we refined the method and worked on the APSS case studies proposed by the experts.

Such iterative process has been adopted along the whole development and evaluation process, however we can identify different phases of the work. At an early stage, the objective of the evaluation was that of understanding the need for ProPrivacy. Later on, after proposing ProPrivacy, we assessed its effectiveness in aiding the work of the practitioners, while capturing/expressing relevant GDPR principles. Domain practitioners applied our method during the design of representative business process models. After each design step, they provided us feedback on the method and suggestions on its modifications, to feed their needs. Finally, we conducted a last interview to evaluate the method.

Interview

We present here the final interview done, months after the conclusion of the project, to a small group of APSS experts, composed by information system engineers with competences in privacy and software engineering. The interview was divided into two parts, in the first one we discussed about privacy in APSS and the approach they adopted for complying with the GDPR, in the second one, after a quick demo of ProPrivacy, we discuss about the method, the tool, and its applicability in APSS. The first part of the interview was structured along the following questions:

- A0-What is your role and your expertise in APSS?
- A1-Can you tell about the importance of privacy in APSS? And what about the efforts and emphasis APSS put into compliance?
- A2-Did APSS adopted any specific method for reaching compliance with the GDPR? Did this include the use of any tool?
- A3-Was it possible to apply these methods systematically in all the organization? Would have been important to do that? How can this be done?
- A4-What about business processes? Is there any documentation of those? Did you use such documentation to verify the compliance with the GDPR?
- A5-Does APSS has special requirements that are specific for the domain and that would be useful to verify in the processes?

The second part of the interview was structured along the following questions:

- B1-With ProPrivacy, existing BPMN 2.0 diagrams can be reused and the load of work of the analysts is reduced to the modeling of authorizations of privacy policies. How do you see the global load of work of the analysts? Does the systematic approach proposed in ProPrivacy help the analysts in this?
- B2-Do you think that the verification of the properties that we propose in ProPrivacy (minimization, ...) would support the work of complying/demonstrating compliance with the GDPR?
- B3-How would you use the custom constraint framework proposed in ProPrivacy?
- B4-Would you adopt ProPrivacy today? Or would you wait for some improvement? Would you ask for some other specific features?

Findings

During the experimentation, domain practitioners fed us with continuous feedback on the applicability of ProPrivacy in APSS and helped us in improving our understanding of the regulations and of the FSE domain. The final interview mainly confirmed us the feedback we got during the experimentation, and gave us the possibility for a final discussion of the results.

Difficulties have emerged about the possibility for an extensive adoption and use of a systematic tool-supported method, such as our, in a large-scale and decentralized organization, such as APSS. Being a quite decentralized organization, in APSS there are no centrally adopted privacy standards for the different departments and there is a diverse privacy-organigram, where privacy experts and responsibilities are not unified into a specific position but are, rather, distributed across departments, from the technical one to the legal one, and the medical ones, where each head of department is responsible for his own team. Because of this, a single tool that can verify privacy across all departments is not easily adoptable. Constraints and standards are different in each department and knowledge is spread among members in each team.

APSS experts agree about the fact that ProPrivacy would be much more easy to use in more classical and structured organizations, where processes are largely documented and a common schema is adopted between departments. This is common to happen in the case of certifications, such as ISO, which require an organization to document a lot of aspects of its inner workings in a rigid structure, including roles, responsibilities, and processes. When ProPrivacy is used in organizations such as APSS, the biggest drawback is the load of work needed to document all the processes, which requires putting together a knowledge that is distributed among people in the organization given the current working model.

However, we found that the systematic nature of the method forced participants to collaborate. This, in a context with no common standards, such as APSS, was found really important by practitioners who were able to share their knowledge and get a more complete vision on privacy. For example, the granularity of consent was largely discussed at the time when engineers and technicians explained to lawyers and medical doctors the APSS implementation of obfuscation preferences and visibility settings in the FSE. The participants, compared the new platform with the old regional data sharing platform, spotting changes in the legal requirements and in the flexibility of managing consent.

We found indicators about the effectiveness of technical solutions of ProPrivacy. The visual representation of consent, annotated on business processes, provides a graphical feedback that was important to support the discussion between experts. Moreover, the built-in GDPR principles, formalized in ProPrivacy in terms of business process constraints, makes it possible to stakeholders to align their understanding of the GDPR and their knowledge on the implementation of privacy regulations in APSS. Finally, custom constraints have proven valuable in expressing constraints from organizational internal privacy practices, where the law imposes minimal requirements but demands organizations to implement further measures that shall be adequate with the privacy risk.

In conclusion, also considering the feedback we get from the interview, the results of the experimentation show that the use of diagrams, not common in APSS, has prompted discussion between practitioners and has allowed to share a knowledge otherwise owned only by single experts of the group. The systematic nature of the method showed difficulties in being applied to decentralized / non-standard organizations, such as, APSS. The availability of a vast source of documentation would be beneficial for a practical systematic applicability of the method.

6.6.2 Scalability

We evaluate the scalability of the ProPrivacy software engine with a set of tests where we measure the execution time over increasingly bigger business processes in input. The approach adopted for the tests is based on custom patterns for a systematic generation of business processes with consent.

Design of the experiment

In a scalability test, independent variables of the inputs are scaled and execution time is measured for each configuration of the inputs. In our case, the inputs of ProPrivacy are processes with consent, while independent variables we want to test are the number of activities and the number of alternative and parallel gateways, which, given their disposition, determine the number of possible execution traces. For example, given a process in which two activities are executed in parallel, two execution traces exist, one for each order of execution of the two activities. When the number of activities and gateways scales, the number of possible execution traces grows, depending on the disposition of the elements in the process. In order to assess the scalability of ProPrivacy software, we developed three different patterns to systematically generate and scale processes and consents. Consent is generated, accordingly to processes, so to satisfy GDPR requirements.

Figure 6.7 shows the three patterns used to scale the processes. In pattern (i), process is scaled by appending new branches composed by a sequence of activities of a constant length. In patterns (ii) and (iii), process is scaled in length, in (ii) by appending activities in existing branches, and in (iii) by also appending a block of elements composed by a fork gateways, followed by a fixed number of branches, each with a fixed number of activities in sequence, all ending in a join gateway that closes the block. The three patterns have been applied using both alternative and parallel gateways, for a total of six patterns.

The dataset containing the scripts used to run the scalability test is available at [56].

Method of execution

The tests have been ran on a machine, with an Intel(R) Core(TM) i5-6200U @ 2.3Ghz and 8GB of RAM. The operating system is Windows 10 by Microsoft(R).

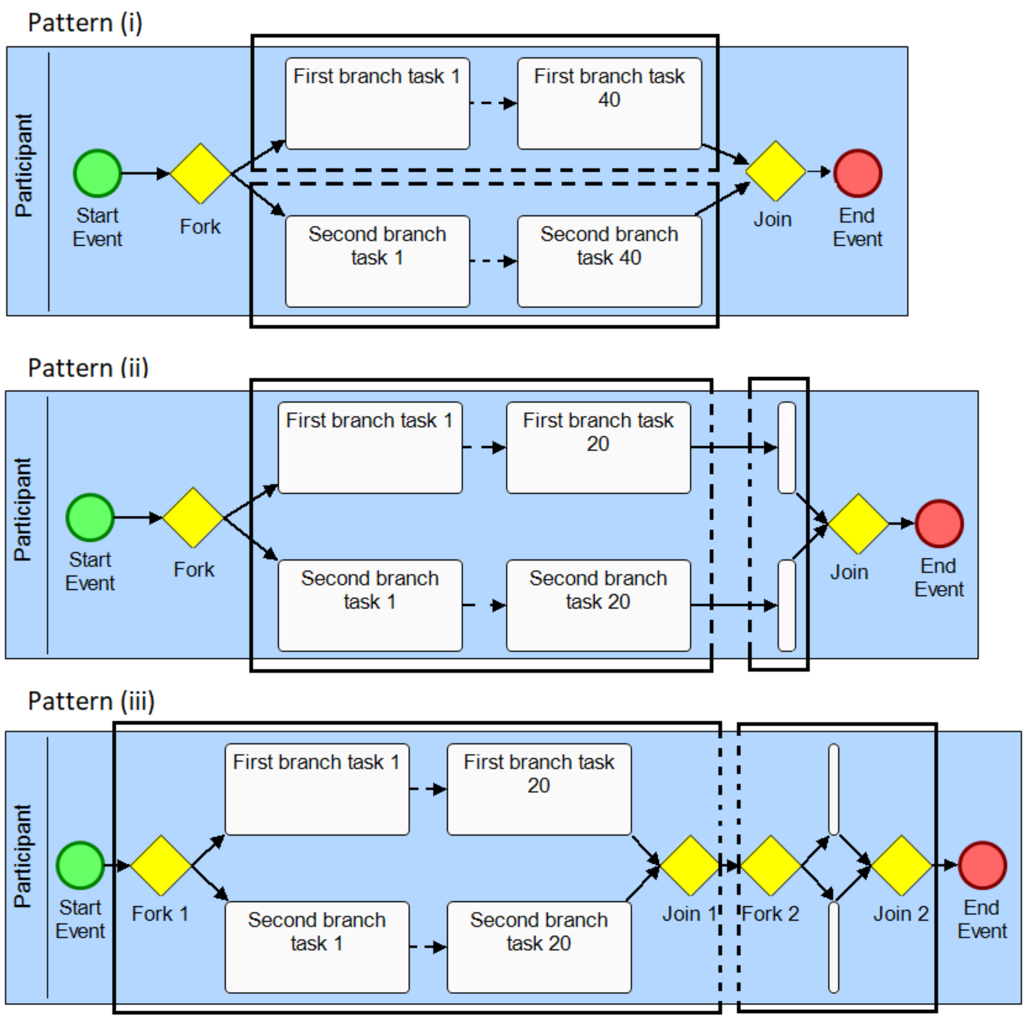


Figure 6.7: Patterns for the generation of processes

The algorithm that generates the inputs is implemented in Java. The test suites is executed by a Java program that runs the DLV executable with the ProPrivacy rules and the generated input, and measures the execution time. The execution time is computed as the average of 4 repeated runs, selected from an overall of 5 runs where the longest is removed. Each test is composed by 10 runs, each one with a linearly increased size of the process. Four series have been considered in each test, each with a different configuration.

We evaluate the scalability of ProPrivacy by comparing the execution time with the size of the input, that we relate to the number of activities. At every step we add 40 total activities that, depending on the pattern, are disposed in a different configuration.

Pattern (i) In the first test, the process scales using the pattern at the top of

Figure 6.7. At each step, 40 new activities are added on *new branches* of the only fork gateway present in the whole process. Four series of tests have been ran, each with a different configuration of the pattern that is designed to keep the same number of activities in runs of different series. Specifically, at each step are added 40, 4, 2, 1 new branches, each with respectively 1,10,20,40 activities in sequence. For example, 40 sequential activities are added by creating a new branch from the gateway "fork".

We design this test to analyze the execution time of the software engine with an increasingly number of breaches (independent variable) and check how the complexity of the algorithm is linked to the number of branches.

Pattern (ii) In the second test the process scales using the pattern in the middle of Figure 6.7. At each step, 40 new activities are added in existing branches, at the end of the process, before the "join" gateway. Four series of test have been executed, each of them has a process with one fork gateway and respectively 10, 5, 4, 2 branches, in which respectively 4, 8, 10, 20 activities are appended in sequence at each step. As before, the number of activities is the same among runs of different series

We designed this test to check how the execution time of the software is linked to the number of activities in sequence (independent variable).

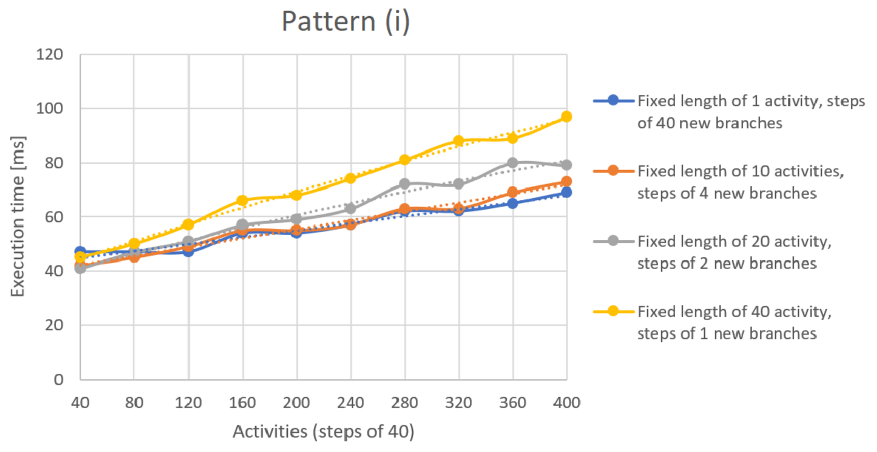
Pattern (iii) In the third test the process scales using the pattern at the bottom of Figure 6.7. At each step a new fork gateway is appended at the end of the process with a fixed number of branches respectively 10, 5, 4, 2. In each branch respectively 1, 2 , 10, 20 new activities are appended, a join gateway closes the branches. As before, the number of activities is the same among runs of different series

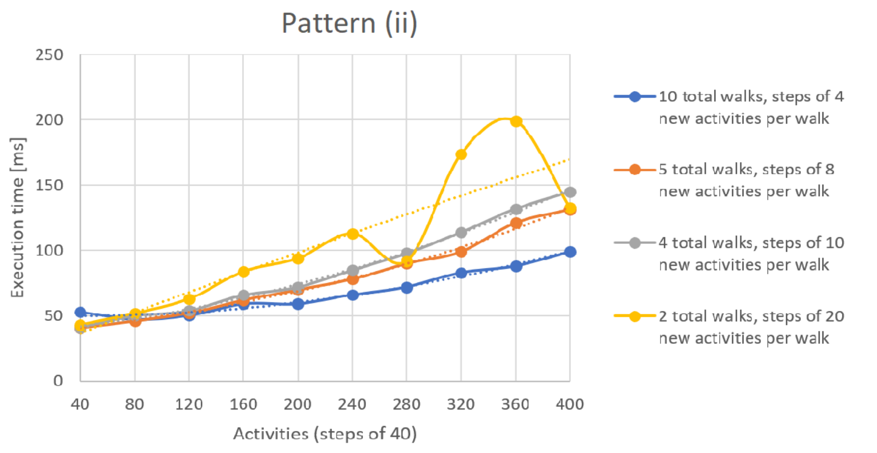
We designed this test to check if the execution time of the software is impacted by the number of possible execution traces (independent variable) of the business process. Indeed, in this test, the number of possible executions traces grows each time a pair of gateways are opened and closed. For example, in Figure 6.7, if we consider only one activity per branch, the first part of the process, between "fork 1" and "join 1", determines two possible execution traces, if we add the second pair of gateways, the number of possible execution traces is 4. If we add another pair of gateways, after the second one, the number of possible execution traces grows to 8, and so on exponentially.

Discussion of the results

We evaluated the scalability of our DLV implementation of ProPrivacy. We systematically scaled processes using patterns where activities are disposed in different configurations.

Figure 6.8 shows the results of the tests when patterns have been applied using alternative gateways. Similar results have been obtained using parallel gateways instead of alternative ones and no major differences have been detected between the two. Therefore, in the following, we discuss about the results in the case of





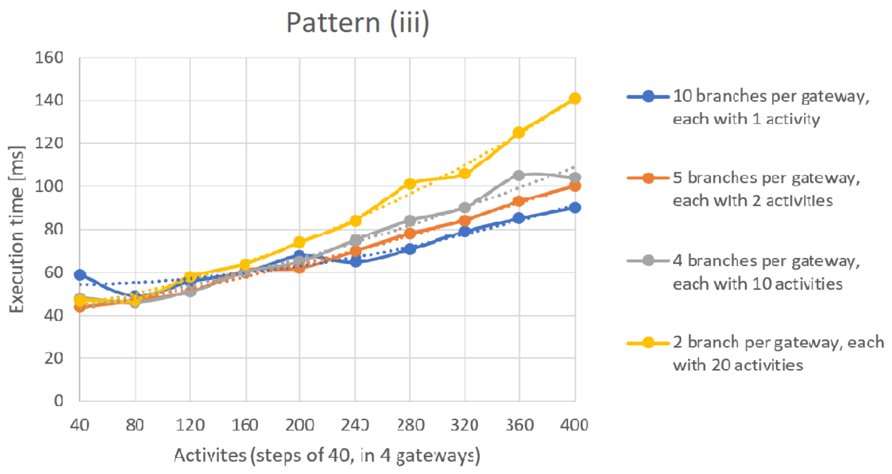


Figure 6.8: Results of scalability test

alternative gateways, but those can be generalized and applied also in the case of parallel gateways.
Pattern (i) The top part of Figure 6.8 shows the results of the scalability test using the first pattern defined in Figure 6.7. The trend is linear and the maximum amount of time taken for the most complex execution is less than 100 ms. The results are promising since the execution time is so short that it will not influence the execution of the ProPrivacy method, users can even launch the analysis multiple times, changing slightly the processes or the rules, to understand the impacts of their changes. The linearity of the trends confirms that the execution time will remain acceptable even with bigger processes, even if, we believe, most of the analyzed process will be well below the 400 activities. The series of test have similar execution times, however, the lower the number of branches the higher is the execution time. This shows that, for what concerns our algorithm, the number of sequential activities has an higher impact in the execution time, than the number of branches.
Pattern (ii) The second graph in Figure 6.8 shows the results of the the second pattern in Figure 6.7. Also in this case the trends are linear and the execution time is short, confirming the results of the previous test. The number of activities has a higher impact on execution time, then the number of branches, confirming that what has been discussed in the previous tests. The series represented with the yellow line shows an unusual behaviour between 280 and 360 activities, we repeated the test multiple times and we obtained similar results we, therefore, believe that this is due to an optimization of DLV engine, rather that to our algorithm.
Pattern (iii) The last graph in Figure 6.8 shows the results of the the pattern in the lowest part of Figure 6.7. As mentioned before, in this test the number of possible execution traces is exponential, yet the execution time of all series is linear and the upper limit is 140 ms. All the observations done for the other tests are confirmed in this test as well.

It is important to note that the processes have been systematically generated using fixed patterns but, in real world, processes could be much more various and complex. However, the size of the processes that we generated, in terms of number of activities, can be compared to real processes. To be noticed that with 400 activities the execution time is under 2 seconds, even with an exponential behavior. Moreover, we want to stress the fact the ProPrivacy is an off-line design method, where performances are not of top importance.

6.7 Discussion

We presented ProPrivacy, a method to automate the analysis of consent on business processes, with respect to GDPR requirements, and guide analysts in the re-engineering of such processes. We presented an iterative approach supporting ProPrivacy, which includes a modeling language based on BPMN 2.0 and automated reasoning capabilities. The modeling language is specified into a formal framework to support automated reasoning. The method has been evaluated with domain practitioners, and we also presented a scalability study of the automated analysis.

The verification of consent on business processes, with respect to specific GDPR principles, can contribute in certifying the compliance with the GDPR. However, other aspects of the regulation may need to be taken into consideration and verified on business processes. There may be requirements of the regulations that cannot be verified at a social or at a procedural level, and for which other requirements models may be more appropriate. Complementary approaches may be integrated into ProPrivacy to obtain a more efficient and broader compliance with the GDPR.

Additionally to GDPR compliance, companies may be interested in analyzing other effects of privacy policies, with the aim of increasing users approval and, at the same time, increase revenue and retain users, by means of service personalization and fidelization programs.

Future work includes an extension of the framework for the identification of eventual data breaches, based on the analysis of critical data flow in the business processes. For example, data taken from a data object could be disclosed, therefore violating the correct flow of data, in the case that the same procedure handle several data artifact that contain different data sets. Processes could, by-design, avoid such violations, controlling the flow of data with strict disclosure requirements, which can be enforced by both machines or humans.

Chapter 7

Maintaining compliance with evolving consent requirements

Consent is a key element in privacy, and it has become a critical element under the EU General Data Protection Regulation (GDPR). Under GDPR, consent is one of a few legal bases available that can be used to process user data (see Article 6-1(a)) and, in most cases, consent is the only viable basis. Consent constitutes legal evidence of user awareness about their data being collected, used, and shared by companies. Under the GDPR, the user is protected because the demonstration of a valid acquisition of the consent is a responsibility of the company, i.e., the user does not need to initiate a request to receive this protection. To this end, the company must present information about how the data will be processed, and then request consent from the user before processing the data. Furthermore, the user may revoke their consent at a later date, which means that the company can no longer process data collected after the revocation (see Article 7). However, the company may continue to process data collected under the previously granted consent, if they choose to do so. In addition to privacy policies, consent can be obtained in other ways prior to collection, including just-in-time consent after a user has already begun using the service, but prior to taking an action within the service [66].

User decisions about granting consent can be driven by the perception of trust in a company with respect to their history of bad privacy practices. For example, recent disclosures by Facebook of personal data leaks to third-parties [19] led some users to restrict their privacy preferences, which control who can access their data, and others to delete their account, thus opting out entirely.

Internally, organizations may make changes to their data practices several times a year. Evidence of changes can be observed in the revision histories of evolving privacy policies. Figure 7.1 shows the changes to the privacy policy of Waze, a popular mobile app for automobile navigation: the y-axis shows the number of statements per policy revision, with the policy revision dates along the x-axis; exact statement matches appear in blue, new statements appear in red, and statements with changes to wording appear in orange. Some of these changes are due to changes in boilerplate language (e.g., how the company or user are referenced), or to data purposes. Under the GDPR, changes to data practices and purposes require consent. While Waze in particular underwent a number of changes from late 2012 to mid-2014, there were significant changes from late 2017 through mid-2018, at which point the GDPR went into effect.

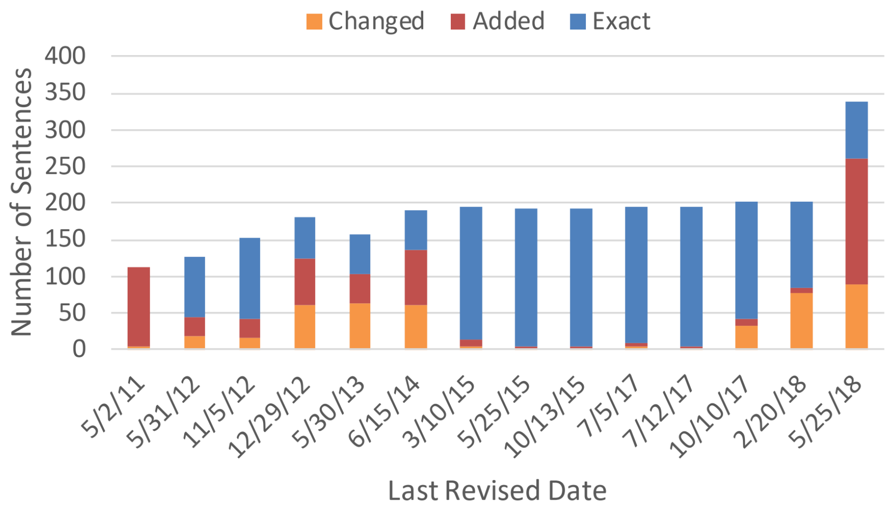


Figure 7.1: Revision History of the Waze Privacy Policy from 2011-2018.

As a matter of requirements engineering, under the GDPR companies must tag their data to know when it was collected and when they obtained consent. Because the GDPR requires that consent be granular, including that purposes be distinguishable (see Recitals 32 and 43), companies should also tag this data with purposes for which consent was granted. Notably, companies may collect data as a consequence of their system design, but they may not process the data for a specific purpose without consent. At scale, one can imagine that companies who are in competitive markets will be looking for new opportunities to process user data, leading to changes to their practices. In addition, users may either be uncomfortable with new purposes, or shift their trust in companies due to improper data handling by the company or the market. To address this problem, we propose a formal framework, expressed in Description Logics (DLs), to support users and companies in the understanding of consent under evolving policies. The framework formally expresses how to automatically verify the compliance of data access, given user consent. The contributions of this work include: (i) a formal framework that can be used by companies and users to reason about data access under multiple consent granting and revocation scenarios, and (ii) an evaluation of the framework based on simulated data generated with the use of Finite State Machines (FSMs) representing realistic scenarios.

7.1 Challenges of evolution

Description Logics (DLs) [12] are the de-facto languages for ontologies and the Semantic Web. DLs are a subset of first-order logic languages, less expressive, but that guarantee resolvability. We chose DL because of its ontology orientation, since we need to represent and verify hierarchical relationships between the data types used in privacy policies. For example, users can provide consent on some broad category of data, such as, personal information, or they can provide separate consents on narrower categories, such as, e-mail address or phone number. These hierarchical relationships can lead to inconsistencies and conflicts in deciding if data can be processed [16]. We propose the use of DLs to build consistent taxonomies of data types used in privacy policies. However, in the formalization of consent evolution, DLs have limited expressiveness and an open world assumption, which introduce specific challenges as we discuss in the following.

DLs languages have limited expressivity, that has been introduced in favor of decidability [40]. Relations are limited to binary and cannot be extended to include other dimensions, such as, temporality. In our case, this is a challenge to extending the static representation of policies to include policy evolution.

The open world assumption means that unknown facts are considered neither false nor true. In contrast, a closed world assumption, also called negation by failure, considers unknown facts false by default. In DL, if one declares that "approved consent" is equivalent to "not consent withdrawn", consent individuals can still be in an unknown state in which they are neither consented nor withdrawn.

Monotonicity is a desirable property for update functions of any formalization, because additions to a knowledge base with this property do not invalidate prior facts. Monotonic update functions are simpler and efficient, and ensure that the existing knowledge base is never changed and only extended by new facts. However, consent evolution appears to have a non-monotonic behavior. For example, in the case of a withdrawal event, the update function should change existent authorizations in a non-monotonic way. To ensure monotonicity, we include the evolution in our representation, so that the whole history of changes to consent and authorizations is maintained. We use temporal steps to define the validity of authorizations over time. Consent is withdrawn from the temporal step in which the authorization is no longer valid. In contrast, consent is approved at the temporal step in which authorizations start to be valid. This distinction will be illustrated further later in Section 7.2.5.

Representation of time and temporal concepts is not specifically supported in DLs. There are temporal extensions for DLs [9], but they all introduce overwhelming extensions of the language and an increase in computational complexity, which we aim to avoid in favor of understandability and efficiency. For example, some temporal extensions are based on the representation of many time-related concepts, relationships, and constraints, such as, an instant or interval of time, the concepts of before, after, meanwhile, started before, ended after, and so on. Moreover, the representation on temporal concepts, such as the interval of time in which a consent was granted and then withdrawn, can be approached in different ways. A strong simplification of the representation of temporality is to focus only on either forward-time or backward-time. In our framework, we have multiple, evolving consents, where a change in consent affects authorizations only in the future and never in the past. Thus, our temporality representation is based on forward-time.

7.2 Modeling consent evolution: formal framework

This section presents the formal framework on consent evolution. The formal framework is specified in Description Logics (DLs) and includes key concepts to express policies, data types, recipients or purposes, modalities, users, data collection, data access, consent, and time steps. The following sections present the main concepts of this framework.

In the notation that we use in this section, lowercase letters are used for individuals and uppercase letters are used for concepts, where individuals belonging to concepts is expressed with the symbol $\in$, for example $d \in D$, then we use the symbol $\subseteq$ to express subsumption between concepts, and the symbol $\cap$ to express intersection of concepts.

7.2.1 Policies

A privacy policy is a set of desired authorizations, needed by organizations to access and use data about users. Such authorizations are granted by each user for his own data. Organizations create these policies to cover the purposes for which they use data in their business. While policies are static and cannot be modified, new versions can be created. In the US, privacy policies typically include statements that users will agree to new versions of the policy. Under GDPR, users have the right to opt-in and thus companies can no longer assume that users are covered by the new policy. Changes to a policy occur for several reasons (1) to make terminology consistent with legal practices; or (2) to describe a new or modified service. In our formalization, we focus specifically on changes of kind (2) that affect authorizations to access and use data.

Each authorization in a policy specifies a data type, a recipient who accesses the data type for a given purpose, and a modality , as follows:

Data types

Data types, such as, e-mail address, are classes of data used in policies to define the scope of an authorization. In the taxonomy of data used in a policy, data types can subsume other data types, for example, contact information subsumes email address and phone number. Parent types in the taxonomy represent classes of data that are broader than children types. In a policy, broader types are used to allow more flexibility, while narrower types of data are used to decrease policy vagueness.

In description logics, data types are sub-concepts of the concept Data. For an arbitrary data type D, this is written as $D \subseteq Data$. Data types are either subsumed one to another, $D1 \subseteq D2$, or disjoined, $D1 \cap D2 = \emptyset$. For example, contact information subsumes email address, while birth date is disjoined from eye color.

Recipient roles and purpose

Data is accessed by an agent or process fulfilling a recipient role, which is assigned to fulfill a specific and separable data purpose. For example, the advertiser role fulfills a marketing or advertising purpose, and the web analyst role fulfills the web analytics purpose. Roles can be hierarchical. For example, in social networking, the role friends serves the purpose of socially engaging on the network. Individuals in this role have a direct relationship to a user. The friends-of-friends role, however, includes both individuals with a direct and indirect relationship, thus this role subsumes the friends role. Finally, a user can define different sub-roles for acquaintances, family members, schoolmates, etc., under the broader friends role.

In description logics , recipients are sub-concepts of the concept Recipient. For an arbitrary recipient R, this is written as $R \subseteq Recipient$. Recipients are either subsumed one to another, $R1 \subseteq R2$, or disjoined, $R1 \cap R2 = \emptyset$. For example, acquaintances are disjoint with family members, since we assume that a person cannot be both an acquaintance, which is defined as a person with whom one has limited familiarity, and a family member, which is a person with whom one is familiar.

Purpose is a combination of data types and recipient roles. When a role is used to specify the recipient, the access to the data is limited to the purposes related with the duties of the role in combination with the data types specified in the authorization. When we consider that, sharing deserves special attention, since it can be a source of re-purposing that harms privacy. Some work has been done to formalize controls for sharing [17]. Repurpose is when data is accessed (used or shared) for a different purpose with respect to the purpose for which it was initially collected (collection). A violation is when data is repurposed without proper consent, differently, if proper consent is provided, repurpose is legitimated.

In description logics, repurpose is when consent for collection belongs to $DataType1 \cap Recipient1 \cap Collection$ and a consent for use exists that belongs to $DataType1 \cap$

$Recipient2 \cap Use$, but no sharing consent exists that legitimate repurpose, such as, $DataType1 \cap Recipient1 \cap Sharing$.

Modalities

Modalities are the data actions that can be authorized. In privacy, it is common to refer to three main modalities: collection, use, and sharing. While avoiding data collection can eliminate privacy risk, several mitigations can be adopted to reduce the risk after collection, e.g., by limiting use and sharing when collection cannot be prevented due to software design.

In description logics, the modality is specified in authorization axioms. Collection authorizations are of the form $D \cap R$, where $D \subseteq DataandR \subseteq Recipient$, while access authorizations are of the form access $(D \cap R)$, where use is in the case of a recipient that is the same of the collection, while sharing is in the case of a different recipient.

7.2.2 Users

Data are collected from users who consent on the collection, use and sharing of their own data. Therefore, both consent and collected data are user-specific.

In description logics, each user is a sub-concept of the root concept User. Since data and consent are user-specific, all user concepts are disjoined one to another. For example, two users are written as $U1 \subseteq User$, $U2 \subseteq User$, and $U1 \cap U2 = \emptyset$.

7.2.3 Data collection

User data is collected by organizations and then used to provide or improve the quality of services. Data can be collected directly from the users or through third parties. Consent-based authorizations must account for time of collection, because consent can minimally be granted for data collected after the time of consent. On the other hand, access can maximally be granted for data collected after and also before the time of consent, by retroactive consent, which we discuss later in Section 7.2.5. Thus, depending on the collection time and type of consent, some data may not be accessible.

The collection log is used to maintain a record of compliance with user consent. The log is continuously updated each time a data element is collected. Each log entry is data type- and user-specific, which means that it includes the user from whom the data is collected and specifies the type of collected data. For example, the e-mail address of a specific user. Additionally, the entry includes also the recipient that collects the data. In the next sections, we will extend the log to include also the collection time.

In description logics, the collection log is represented by individuals of the intersection between a user, a data type, and a recipient. For example, $dataCollection1 \in U1 \cap D1 \cap R1$, where $U1 \subseteq User$, $D1 \subseteq Data$ and $R1 \subseteq Recipient$.

7.2.4 Data access

User data previously collected can be later accessed in the case of use or in the case of sharing. The access log is used to verify accesses to already collected data, with respect to consents of the user of the data. Each log entry includes the collected data and the recipient of the access. We will extend this in the next sections to also include the access time.

In description logics, each access log entry is an individual of concept Recipient in access relation with a data collection, which is defined, as introduced in the previous section, as $dataCollection \in U1 \cap D1 \cap R1$. For example, if we consider a data collection $dataCollection1 \in U1 \cap D1 \cap R1$, the data access by a recipient R2 is $dataAccess1 \in R2 \cap access(dataCollection1)$.

7.2.5 Consent

Consent is the acknowledgement by the user to grant an authorization desired by the company as specified in the privacy policy. Specifically, by consenting to a policy, the user grants the authorizations on his or her data. In description logics, authorized actions define the permitted collection log and access log individuals. Consents impose class axioms to constrain the collection and access logs. For example, consent on collection by $U1$ for the data $D1$ to the recipient $R1$ is written as $U1 \subseteq (D1 \cap R1)$, so that no collection from user $U1$ is allowed that is not of data type $D1$ and collected by the recipient $R1$, where $U1 \subseteq User$, $D1 \subseteq Data$ and $R1 \subseteq Recipient$; Consent on access by user $U1$ to recipient $R2$ for data $D1$ is written as $U1 \subseteq R2 \cap \forall access(D1)$, where the access relation is expressed as a universal restriction.

Retroactivity

Consent that only authorizes use of data collected in the future, is called non-retroactive consent. Retroactive consent grants authorization in the policy for both newly and previously collected data. Retroactivity is important, because it could be the only way for organizations to access historical data and to re-purpose this data under a new policy. However, it could also be dangerous, because old data could suddenly become accessible under a retroactive consent, during a time when users believed it was not accessible. The access to data collected in the past, and for which consent to further use has been revoked, can be re-obtained with a retroactive consent.

Consent withdrawal is the opposite of consent approval. When consent is withdrawn, previously granted authorizations are revoked. However, the data may remain accessible because of other consents, of which authorizations remain untouched. Similar to non-retroactivity of approval, non-retroactivity of withdrawal means that, even if the authorizations are no longer valid for newly collected data, data collected under a previous authorization can still be processed. A non-retroactive withdrawal can be dangerous for the user, because data authorized in the past remain accessible in the future. Again, similar to retroactive approval, retroactive withdrawal revokes authorizations for using both data collected in the past or data that could be collected in the future.

Evolving consent

Users authorize the collection, use and sharing of their data by consenting on a privacy policy. Later, they can revoke these authorizations by withdrawing their consent. This can happen several times in the case users change their privacy preferences or when organizations release a new privacy policy. The consent log includes all the events of consent approval and withdrawal for each user. Each log entry includes: (i) the time of consent/withdrawal, (ii) the user, and (iii) authorizations approved and revoked.

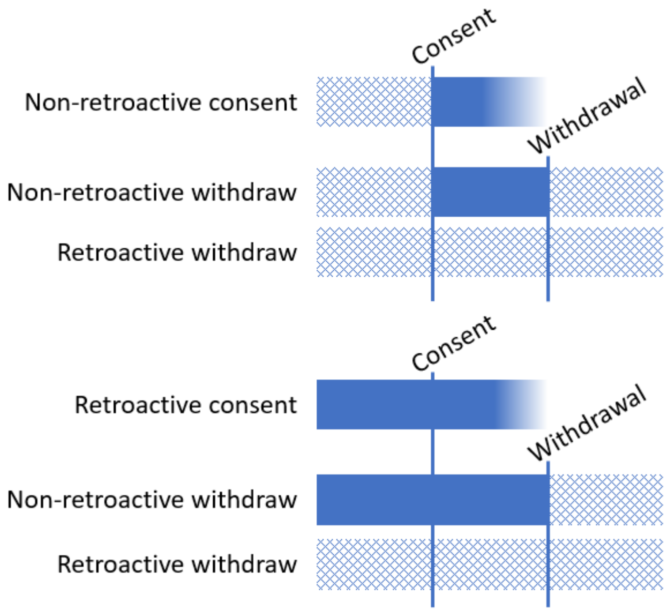


Figure 7.2: Retroactivity.

Figure 7.2 shows the effects of retroactive consent and withdrawal on authorizations. Under GDPR, companies are permitted to use retroactive consent and non-retroactive withdrawal. There are no restrictions in GDPR preventing companies from using non-retroactive consent or retroactive withdrawal, which provide users with greater privacy protection, but which would make less data available for business use.

Four situations are shown in Figure 7.2, given by the combination of consent and withdrawal retroactivity. Horizontal shaded bars show current authorizations by collection time, where time is represented on the x-axis. The dark shading shows where data collected at a specific time is accessible; the light shading shows where data is inaccessible. The vertical lines show times where consent is approved or withdrawn. Non-retroactivity affects authorizations given the collection time of the data. Non-retroactive consent, in the top part of the figure, grants the access only to data collected in the future. While non-retroactive withdrawal, in the second and fifth row in the figure, revoke access to data collected in the future and not to previously authorized data access.

In description logics, consent evolution is defined as subsumption of authorizations. In particular, current authorizations are subsumed by the disjunction of previous and new authorizations, from which withdrawn authorizations are removed, $PostAuth \subseteq PreAuth \cup NewAuth$
$WithdrawnAuth$, where it holds that $PostAuth \cap WithdrawAuth \subseteq \emptyset$.

More in detail, consider an initial authorization $D1 \cap R1$ for the collection of $D1$ by $R1$, then a new authorization $R1 \cap access(D1)$ is given for the access by $R1$ of data $D1$, finally the collection authorization $D1 \cap R1$ is withdrawn. The final authorization is therefore $((D1 \cap R1) \cup (R1 \cap (access(D1)))) \setminus (D1 \cap R1)$, which is equal to $R1 \cap access(D1)$.

7.2.6 Time

Time has to be considered when discussing evolution. Consent and withdrawal affect authorizations, making them evolve over time. Moreover, if we consider retroactivity, authorizations themselves consider time of data collection, and may or may not apply to specific data from the past. In description logics, time steps are represented by sub-classes of the concept Time. Temporal order is defined by subsumption. We adopt forward-time convention, where after is expressed as $\subseteq T$, while past is expressed as $\subseteq notT$, and an interval of time can be expressed as $\subseteq Tstart \cap notTend$.

Time of collection

Collected data are individuals of type User, Data, and Recipient. We also make it of type Time to express time of collection e.g. $dataCollection1 \in User \cap Data \cap Recipient \cap Time$. For example, when a data D1 is collected from user U1 by recipientR1 at time T1, the following individual is created: $dataCollection1 \in U1 \cap D1 \cap R1 \cap (T1 \cap notT2)$;

Time of access

Data access is expressed as an individual of type Recipient in access relation with some collected data. We make this data access individual also of type Time to express time of access e.g. $access1 \in Recipient \cap access(dataCollection1) \cap Time$. For example, in the case of access to dataCollection1 by recipient R1 at time T2: $access1 \in R1 \cap access(dataCollection1) \cap (T2 \cap notT3)$.

Time of consent

Consent expresses authorizations for data collection and access, expressed as axioms on the collection and access log. Authorizations always apply on future logged events and never on old ones, even in the case of access to data collected in the past. For example, the repurpose of previously collected data with a retroactive consent can affect only future accesses to the data. Every time a consent is given or withdrawn, authorizations are modified, added or removed. For example, given a time sequence $T1$, $T2$, $T3$, when a new consent $D2 \cap R2$ is retroactively given at $T2$ by $U1$, previous authorizations are still valid: $T2 \cap U1 \subseteq (T1) \cup (D2 \cap R2)$. Then, when consent $D2 \cap R2$ is retroactively withdrawn at T3: $T3 \cap U1 \subseteq (T2) \setminus (D2 \cap R2)$. In which case the authorization $D2 \cap R2$ remains valid only between T2 and T3.

In the case of use or sharing, consent can be given retroactively, which means that also data collected in the past can be accessed. For example, in the case of consent given at T1 by U1, for the sharing of D1 by R2, non-retroactive consent is written as $(T1 \cap U1) \subseteq access(T1 \cap D1)$, while retroactive consent is written as $(T1 \cap U1) \subseteq access(D1)$.

When access withdrawal is non-retroactive, old data can still be accessed, but not new ones. For example, in the case of withdrawal at T2 by U1, for the sharing of D1 by R2, non-retroactive withdrawal is written as $(T2 \cap U1) \subseteq T1 \setminus access(D1 \cap T2)$, while retroactive withdrawal is written as $(T2 \cap U1) \subseteq T1 \setminus access(D1)$.

7.2.7 Consent granularity and privacy preferences

Users have their own specific privacy preferences that depend on their needs and on the confidentiality of the data and the trust they have with the organization with whom they share their data. The GDPR refers to the granularity of consent and requires consent to be granular with respect to purposes for which data are collected, used, and shared. In addition to consent, organizations can offer the possibility for users to have more fine-grained control of their data collection, use and sharing. These preferences can be usually expressed by the users, as general or content-specific settings.

We consider three levels of granularity of privacy preferences, from the more coarse-grained control to the more fine-grained: (i) data type level, (ii) recipient/-

data specific, (iii) individual preferences.

In the first level, preferences are used to control the type of data to be shared. For example, consider a navigation application that features real-time location sharing among groups of friends. When using the application only for routing, the user can decide to avoid the application to share his position, while, in the case the application is used for real-time position sharing, the user can modify the group of people with whom the position is shared, which is a preference of the second level which we discuss in the next paragraph.

At the second level of privacy preferences, the user can authorize specific data to specific recipients. This is the case of Facebook custom groups, which allow the users to change the visibility of their posts by adding or removing friends from the groups to which the posts are visible. Adding or removing a friend from a custom group has a retroactive effect, since it also affects older posts. A different case is the default visibility setting, which, applies only to new posts, thus revealing a non-retroactive effect.

The third level allows for the most fine-grained control possible, on specific pieces of data, for which the user decides the specific recipient that is authorized to access the data. For example, the messaging platform within Facebook allows to communicate and share information with a specific user.

Settings are modified by users over time given their privacy preferences, in addition, organizations change their systems to introduce new services. This can lead to unplanned violations of privacy. For example, if Facebook would change its platform, then a user setting could become reset to default settings or new underlying accesses could appear in the access log due to the re-design, in violation of the consent or preferences. In such instances, organizations like Facebook would desire to notify users and ask them to revise their privacy preferences in parallel to rolling out the new platform change.

7.2.8 Updating the knowledge base

The Knowledge Base (KB) includes facts and constraints about the concepts that we have seen until now: consent, data collection, and data access. In description logics, it is a collection of axioms expressed over concepts, relations and individuals. Because of evolution, logs grow over time and so does the KB, which should be consequently updated. To easily update the KB, we provide an update function for each main event. These update functions are intended to be called following the order in which events occur, whereas applying the functions to non-temporally ordered events could result in an inconsistent KB. The collection, access, and consent logs are decoupled and every update of the KB only increases it and never modifies old facts or axioms (monotonic update functions), also in the case of retroactive consent or withdrawal. The KB is updated in the case of: (i) data collection, (ii) data access, and (iii) consent approval or withdrawal.

In the event of data collection

A data collection event generates a new individual in the conjunction given by the data type, the user from whom the data is collected, the recipient who collects the data, and the current time interval. Note that within the same interval of time, several data may be collected. Listing 7.1 reports the update function for a data collection event in pseudo-code.

Listing 7.1: Data collection

Given the collection of a data D from the user U by the recipient R in the time interval $T1 \cap notT2$, declare a new individual:

$$d \in D \cap R \cap U \cap (T1 \cap notT2)$$

In the event of data access

A data access event generates a new individual in the access relation with a previously collected data, in conjunction with the recipient accessing the data and the time of access. Note that within the same interval of time, several accesses may happen. Listing 7.2 reports the update function for a data access event in pseudo-code.

Listing 7.2: Data access

Given a data access by recipient R in the time interval $T1 \cap notT2$ to a collected data d, declare a new individual:

$$a \in R \cap T \cap access(d)$$

In the event of consent approval/withdrawal

A consent approval or withdrawal event generates a new class of times for future data collections and accesses, and restricts or expands operations that are authorized in the new class of times. Listing 7.3 is the update function for retroactive consent approval. Listing 7.4 is the update function for non-retroactive consent approval. Listing 7.5 is the update function for retroactive consent withdrawal. Listing 7.6 is the update function for non-retroactive consent withdrawal. In the following listings, we always consider T1 as the new class of times, where T0 was the previous class of times valid before the last consent event.

Listing 7.3: Retroactive consent approval

Given a retroactive consent by user U1, for access to data D1 by recipient R1, assert the axiom:

$$T1 \cap U1 \subseteq (T0 \cap U1) \cup access(D1)$$

Listing 7.4: Non-retroactive consent approval

Given a non-retroactive consent by user U1, for access to data D1 by recipient R1, assert the axiom:

$$T1 \cap U1 \subseteq (T0 \cap U1) \cup access(D1 \cap T1)$$

Listing 7.5: Retroactive consent withdrawal

Given a retroactive consent withdrawal by user U1, for access to data D1 by recipient R1, assert the axiom:

$$T1 \cap U1 \subseteq (T0 \cap U1) \setminus access(D1)$$

Listing 7.6: Non-retroactive consent withdrawal

Given a non-retroactive consent withdrawal by user U1, for access to data D1 by recipient R1, assert the axiom:

$$T1 \cap U1 \subseteq (T0 \cap U1) \setminus access(D1 \cap T1)$$

7.2.9 Automated analysis

The analysis of consent, withdrawal, data collection, and data access logs can help in the detection of violations of consent. The analysis automation is based on satisfiability of the model in description logics. Authorization axioms, transduced from consent logs, constrain the collection and access individuals, transduced from data logs. If a data collection or access is unauthorized, an axiom is violated, and the log entry individual results will be inconsistent.

Purpose limitation

The GDPR defines the purpose limitation principle, which means that data can only be used for a new purpose, (1) if the new purpose is comparable to the original purpose, (2) if consent is obtained for the new purpose, or (3) if there is a lawful basis. Herein, we formalize the first two conditions needed to guarantee the purpose limitation principle, (1) $R2 \subseteq R1 or (2) T2 \subseteq T1 \cup (R2 \cap accessData)$.

For example, consider data initially collected and used for some specific purpose $dataCollection1 \in R1 \cap T1$, authorized by consent $T1 \subseteq R1$. Later, the same data is accessed for a different purpose, $dataAccess1 \in R2 \cap T2 \cap access(dataCollection1)$. This operation is authorized if either: (i) the new purpose is comparable to (subsumed by) the original purpose, $R2 \subseteq R1$, where $T2 \subseteq T1$; or (ii) the new consent includes the new purpose $T2 \subseteq T1 \cup R2$.

7.3 Worked examples

This section presents four realistic scenarios of policy and consent evolution, using the formalism to support the reader in a better understanding of the consent monitoring scenario itself.

7.3.1 Organization expansion

A bus company is going to introduce a new service, which will allow the users to use their smartphones to check the routes and the position of the busses in real time using a mobile app. This application will help the users to use the company buses and, meanwhile, it will allow the company to collect data to optimize the bus routes.

The bus company creates a new privacy policy that describes the purposes for using the new user data. The company acquires the new consent of the users directly through the app. Users not interested in using the app will remain under the old policy. However, the company allows users the possibility to deactivate access to their position data for statistical analyses with an opt-out-able preference.

Listing 7.7: Evolution example

Consider the following data hierarchy:

$$Location \subseteq UserData \subseteq Data$$

At T1 a new policy is created:

$$T1 \subseteq ((Location \cap Statistics)) \cup (Statistics \cap access((Location) \cap T1))$$

At time T2 consent for the analysis of position is withdrawn:

$$T2 \subseteq T1 \setminus (Statistics \cap access(Location))$$

7.3.2 Historical data

Users of the bus company have used rechargeable smartcards for years. The company has been recording all data from the smartcard system. Now, they want to use this data to analyze and optimize bus routes. To do so, a retroactive consent is acquired for the new policy to permit access to historical location data of the users and to re-purpose (following the purpose limitation principle) this data for routing optimization analysis.

Listing 7.8: Historical data example

Consider T3 the time of new consent and T2 time of old consent, so that:

$$T3 \subseteq T2 \cup (Statistics \cap access(Data))$$

7.3.3 Opt-in-able privacy preference

Due to the increased user discomfort from being tracked by the app, the bus company changes the policy again. In the new policy, the use of location data for routing optimization is by default turned off, even if the data is still being collected. The user can manually opt-in to the preference to authorize the company to use their location data, which includes the data from the app and additional data provided by third parties.

Listing 7.9: Opt-in-able example

Consider the following data hierarchy:

$$ThirdPartyLocation \subseteq Location$$

Time of new policy is T4:

$$T4 \subseteq T3 \cup Location \setminus (Statistics \cap access(Location))$$

7.3.4 Consent withdrawal

After a major scandal for data breaches that involves the bus company, concerned users begin to retroactively withdraw their consent to the bus company policy after every use of the mobile app. In order to use the app, users are still asked to consent the policy, however. By using a non-retroactive consent, they limit the access to only new data. Doing so, the app user experience lacks all the data-driven functionalities, such as preferred routes, notifications of preferred buses delays, and bus suggestions based on user schedule.

Listing 7.10: Non-retroactive consent and retroactive withdrawal example

Non-retroactive consent is given by concerned user U1:

$$((U1 \cap T0)) \subseteq ((Location \cap T0)) \cup access((Location \cap T0))$$

Consent is retroactively withdrawn by concerned user U1:

$$((U1 \cap T1)) \subseteq T0 \setminus Location \setminus access((Location)) \subseteq \emptyset$$

7.4 Evaluation

In this section we present an evaluation of our formalism, where we demonstrate how it can be used to detect violations in realistic scenarios. We generate realistic logs by simulating the system using Finite State Machines (FSMs) that represent the evolution scenarios. Companies do not need to create and use the FSMs to simulate the logs if they can transduce real logs from running systems. We present three scenarios of consent violation and we show how to detect the violations on simulated logs with the help of our formalism.

7.4.1 Waze policy evolution

The Waze privacy policy evolved multiple times from 2011 to 2018 (see Figure 7.1). In 2012, the term Unique Device Identifier (UDID) appeared for the first time in the Waze policy in the following sentence: "Personally identifiable information may also be shared with Waze's other partners and service providers, with the express provision that their use of such information must comply with this policy. For example, Waze may share your mobile device's Unique Device Identifier (UDID) with ad networks, for the purposes described above." By including UDID in their policy, Waze may assume under US law that they have consent to share the UDID collected within their app with third party advertisement companies for the purpose of targeted advertising.

We now consider a hypothetical policy evolution to illustrate how introducing UDID into their policy could be the result of a violation detected using our framework. In this example, personal data are used for online advertising purposes: (1) non-targeted advertising, which is directed at categories of users based on their demographic data (gender, age, postal code); (2) targeted advertising, which is directed to specific users by advertisers using the Unique Device Identifier (UDID). We envision that the company begins collecting and sharing the UDID with advertisers without the consent of the user and then later reviews the policy in order to request consent for using the UDID. Figure 7.3 presents an FSM representation of the evolution: at the top, the earlier non-targeted advertising process; at the bottom, the evolved targeted advertising process.

In state A1 and A2 of Figure 3, a new user consents to the advertisement policy, then, in state B1 of the non-targeted process, the system shares users' demographic data, such as their gender, age or birth date, and postal code, with an advertiser, allowing for demographic-based, non-targeted advertising. In state B2 of the targeted process, the system shares the user's UDID, allowing for targeted advertisements.

The FSM diagram can be used to generate realistic data and consent logs. Herein, we focus on system evolution and consent evolution. The "Consent to advertisement policy" events that identify transitions between states Ax and Bx generate a consent log entry of the form $TimeOfAccess \cap AccessedBy \subseteq access(Data \cap CollectedBy)$.

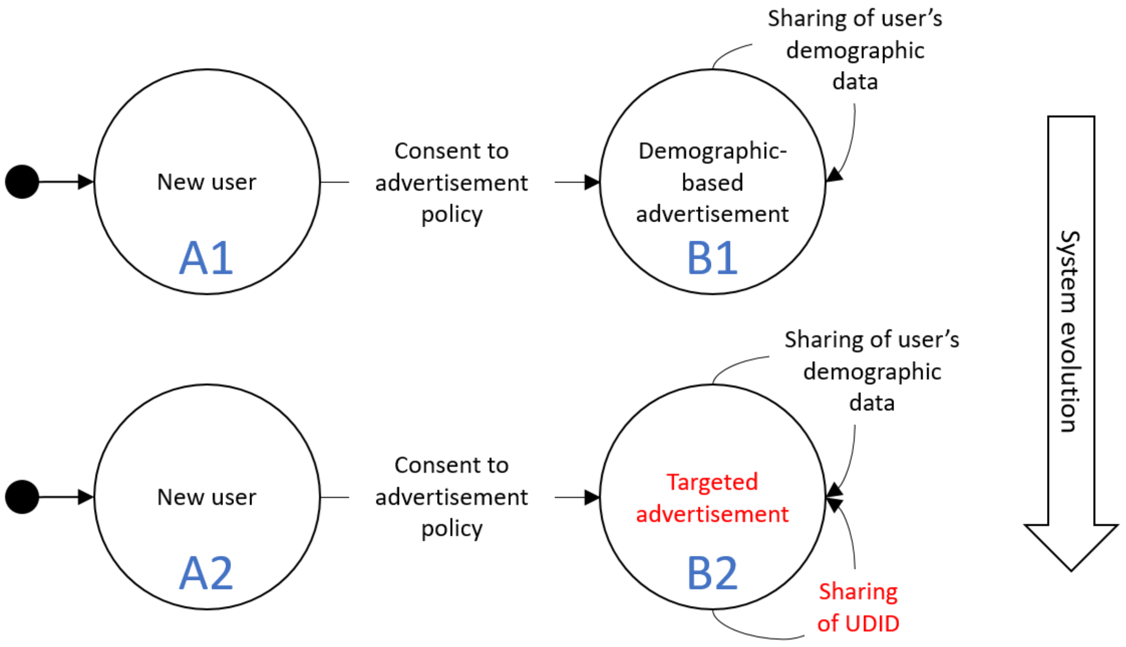

Sharing of user's demographic data
New user
A1
Consent to advertisement policy
Demographic-based advertisement
B1
System evolution
Sharing of user's demographic data
New user
A2
Consent to advertisement policy
Targeted advertisement
B2
Sharing of UDID

Figure 7.3: Finite State Machine (FSM) representation of targeted advertisement evolution.

The "sharing" events generate an access/sharing log entry of the form $share \in T \cap Recipient \cap access(Data)$.

Table 7.1 shows an example of a realistic log generated with the use of the FSMs of Figure 7.3. Initially, at line 1, consent to the advertisement policy allows the sharing of demographic data Demographic. At line 2, demographic data is accessed by advertisers Adv. At line 3, the system has evolved to allow UDID to be shared with advertisers Adv. However, the UDID is shared without consent of the user. Later, at line 4, the company updates the policy and requests consent from the user. Because of this, sharing of UDID at line 5 is now authorized.

Line	FSM state	Log
1	A1	$T0 \cap Adv \subseteq access(Demographic)$
2	B1	$share1 \in T0 \cap Adv \cap access(Demographic)$
3	B2	$share2 \in T0 \cap Adv \cap access(UDID)$
4	B2	$T1 \cap Adv \subseteq access(Demographic \cap T1) \cup access(UDID \cap T1)$
5	B2	$share2 \in T1 \cap Adv \cap access(UDID)$

Table 7.1: Targeted advertisement evolution

A violation is detected by the framework in the sharing of UDID at time T0. The individual share2 is not valid because at time T0, Adv are authorized to access only to Demographic data. Such violations can occur because software evolves independently of policies; however, they can be discovered in this framework and used to remediate the problem, including updating the policy.

Consent at line 4 is non-retroactive: this consent authorizes access at T1, only for data collected at T1 and not before. The combination of a time where consent was not given or was withdrawn with a non-retroactive consent, can create windows of collection time in which data cannot be accessed.

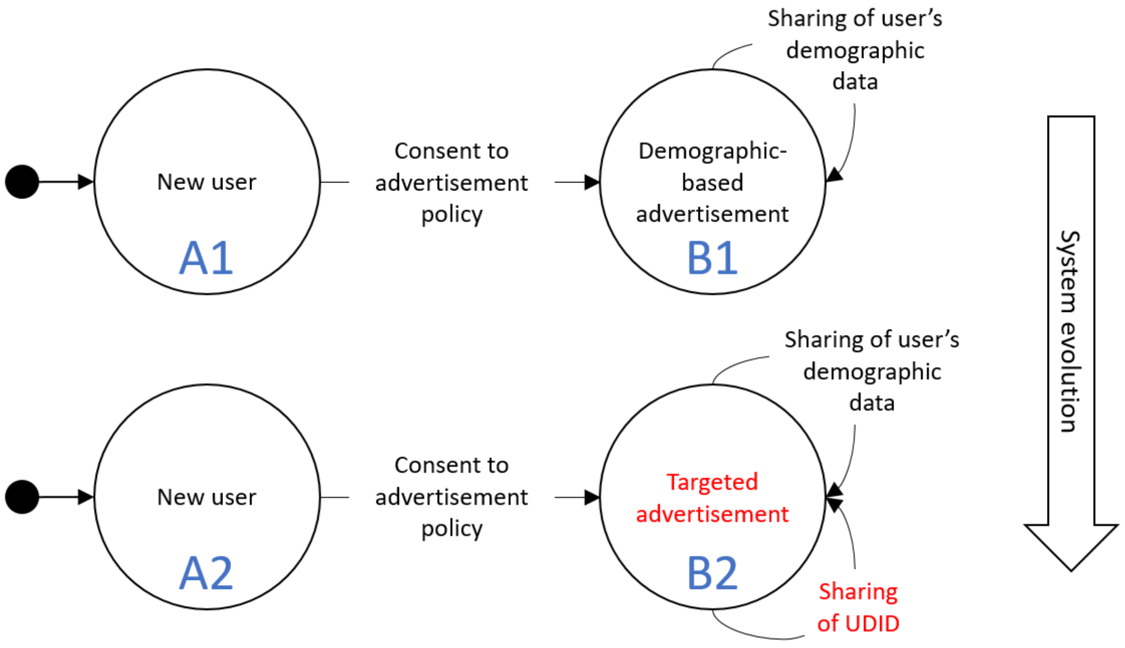
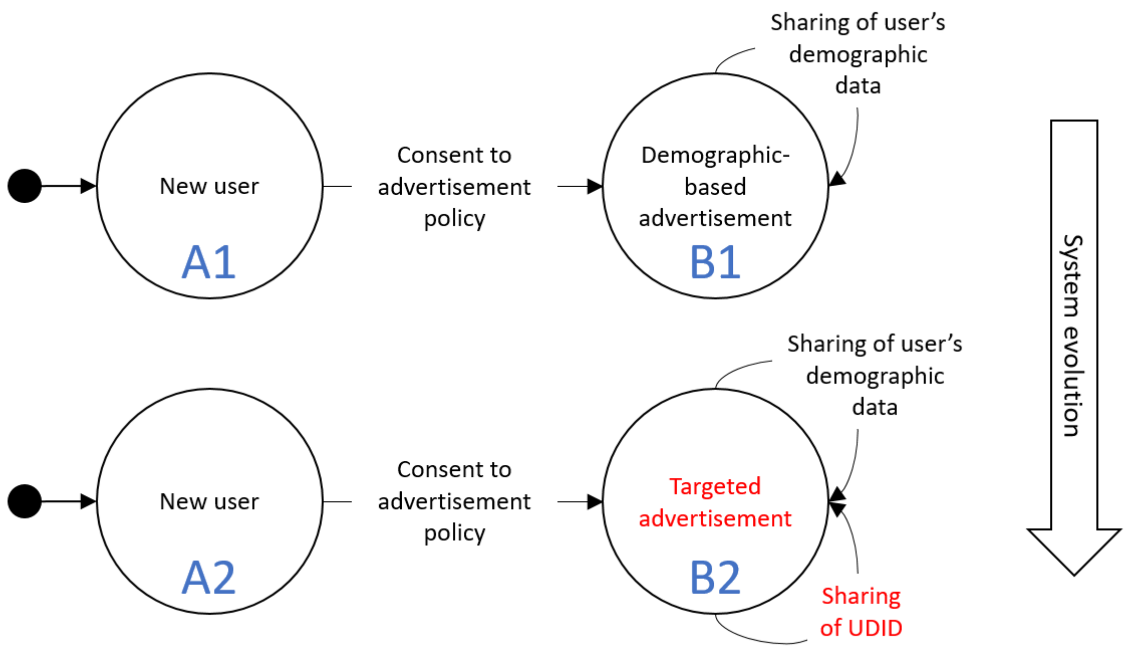

Sharing of user's demographic data
New user
A1
Consent to advertisement policy
Demographic-based advertisement
B1
System evolution
Sharing of user's demographic data
New user
A2
Consent to advertisement policy
Targeted advertisement
B2
Sharing of UDID

Figure 7.4: Inaccessible data.

In Figure 7.4, a withdrawal followed by a non-retroactive consent can leave a window of old data inaccessible in the future. Data collected at three successive time points T1, T2 and T3 are represented by squares, circles and pentagons, respectively: red shapes indicate inaccessible data, green shapes represent accessible data. Data accessibility is governed by granting and withdrawing consent under policy 1 and policy 2. The consent to either one of the two policies is sufficient to grant access to the data, while withdrawal of consent to both the two policies is needed to prevent access to the data. From left to right, access to data collected at time T1 is initially granted by consent on policy 1. Such consent is later withdrawn non-retroactively so that data collected later at time T2 is made inaccessible, while data previously collected at T1 remains accessible. Until now, consent on policy 2 does not grant any access to the data. Finally, non-retroactive consent on policy 2 grants access to data collected later at time T3. At time T3, data collected at time T2 remains inaccessible while access to data collected at time T3 is granted by policy 2 and access to data collected at time T1 is granted by policy 1.



Sharing of user's demographic data
New user
A1
Consent to advertisement policy
Demographic-based advertisement
B1
Sharing of user's demographic data
New user
A2
Consent to advertisement policy
Targeted advertisement
B2
Sharing of UDID
System evolution

Figure 7.5: Evolution of policies in a scenario of multiple users granting and revoking consent.

Figure 7.5 shows a hypothetical scenario to illustrate how policy and feature evolution interact with users granting and revoking consent on multiple policies. Initially, all users have consented to Policy 1. Later, the company rolls out a new feature that they target to early adopters under the new Policy 2, while other users remain under the old Policy 1. After some time when the new feature is evolved and user satisfaction around the feature appears stable, the company rolls out the feature to the entire user base under Policy 3, which is an enhanced, retroactive one with regard to Policy 2. At this time, all the data collected under Policy 2 are repurposed under Policy 3 to cover data collected for both the early adopters and the wider user base with regard to the new feature.

7.4.2 Facebook case study

A bug in Facebook in May 2018 caused 14 million users' default privacy setting to change to public without any warning. Users who were posting content in private mode, for example, visible only to his friends or to a specific group of users, would have unexpectedly found the setting for default audience switched to public, so that all new posts were going public by default. Figure 7.6 shows how the default audience setting works. Depending on this setting the default visibility of new content that is posted is to the public or to a specific group of friends.

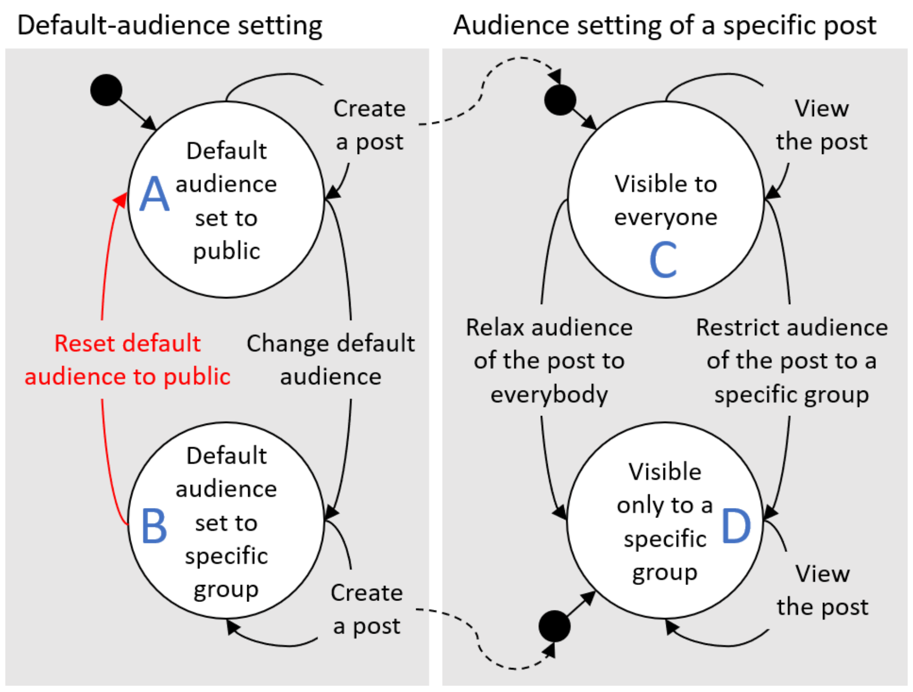

Default-audience setting
Audience setting of a specific post
Default audience set to public
A
Create a post
Change default audience
Reset default audience to public
Default audience set to specific group
B
Create a post
Visible to everyone
C
View the post
Relax audience of the post to everybody
Restrict audience of the post to a specific group
Visible only to a specific group
D
View the post

Figure 7.6: Finite State Machine (FSM) representation of Facebook default audience setting behavior.

The two states A and B of the FSM on the left represent the default audience setting. Depending on this, the audience setting of new posts is by default set to public or to a specific group of friends. The states C and D of the FSM on the right represent the audience to whom a specific post is visible. In red the effects of the bug, which reset the privacy preference setting on behalf of the user.

The FSM diagram can be used to generate realistic data and consent or preferences logs. More in details, in this scenario, we focus on a privacy preference that is valid for future posts and on a privacy preference that is specific for each post. In this scenario we omit the data type since we only consider data of type Post. The "Create a post" transitions generate a data collection log entry of the form $post \in T$, while the two transitions "View post" generate an access log entry of the form $dataAccess \in T \cap access(post) \cap Audience$. The "Change/reset default audience" transitions between states A and B generate a consent/preference log entry of the form $T \cap access(T) \subseteq Audience$. Similar are the two transitions between states C and D to change the audience of a specific post: $T \cap access(post) \subseteq Audience$.

Table 7.2 shows an example of a realistic log generated with the use of the FSMs of Figure 5. In square brackets are the states of the FSMs, e.g. [i;ii;iii], where (i) is the state of the default audience setting, (ii) is the state of the audience state of post1, (iii) is the audience setting of post2. The underscore symbol "_" is used to mark a non-initialized FSM. Initially, at line 1, default audience for new posts is set to public. Then, at line 2, default audience for new posts is restricted to Friends. At line 3 and 4, a post is created by the user and viewed by a friend. Later, at line 5, the violation happens, Facebook resets default audience without user consent. Then, at line 6, the user posts $post2$, which can be seen by the public $access(post2) \subseteq Public$, even if the user may think it is visible only by friends $access(post2) \subseteq Friend$. Finally, at line 7, the post is viewed by someone who is not his friend.

Red color in line 5 of Table 7.2 shows the bug that causes the violation. Red color in Line 5 shows the violation: content is posted and made visible to the public. However, with regard to user intent, this is not allowed, because he had previously set the default audience preference to Fiends and not to Public, where $Fiends \subseteq Public$.

Line	FSM state	Log
1	[A; _; _]	$T0 \cap access(T0) \subseteq Public$
2	[B; _; _]	$T1 \cap access(T1) \subseteq Friends$
3	[B; _; _]	$post1 \in T1$
4	[B; D; _]	$dataAccess1 \in Friends \cap access(post1) \cap T1$
5	[A; D; _]	Bug: $T2 \cap access(T2) \subseteq Public$
		User thinks: $T2 \cap access(T2) \subseteq Friends$
6	[A; D; C]	$post2 \in T2$
7	[A; D; C]	$dataAccess2 \in (Public \setminus Friends) \cap access(post2) \cap T2$

Table 7.2: Facebook

A violation is detected by the framework if we omit to update the user consent when the preference is changed by Facebook without the authorization of the user. In the example of Table 7.2, $dataAccess2 \in (Public \setminus Friends) \cap access(post2) \cap T2$, where $post2 \in T2$, is not valid because user consent imposes $T2 \cap access(T2) \subseteq Friends$, while $dataAccess2 \in T2 \cap accessT2$, but $dataAccess2 \notin Friends$.

7.4.3 Google case study

Google location history setting allows users to stop google from recording their locations . However, in 2018, the setting presented to the users was misleading: "with Location History off, the places you go are no longer stored". The setting does not stop the collection of all location data by google, which may still be collected by other specific services, such as, Google Search and Google Maps. Figure 7.7 gives a representation of Google tracking behavior as a Finite State Machine (FSM), where the location history setting is turned-on in the states on the left (A and D) and it is turned-off in the states on the right (B and C), while background activity happens in the states on the bottom (A and B) and app or web activity happens in the states on the top part (C and D).

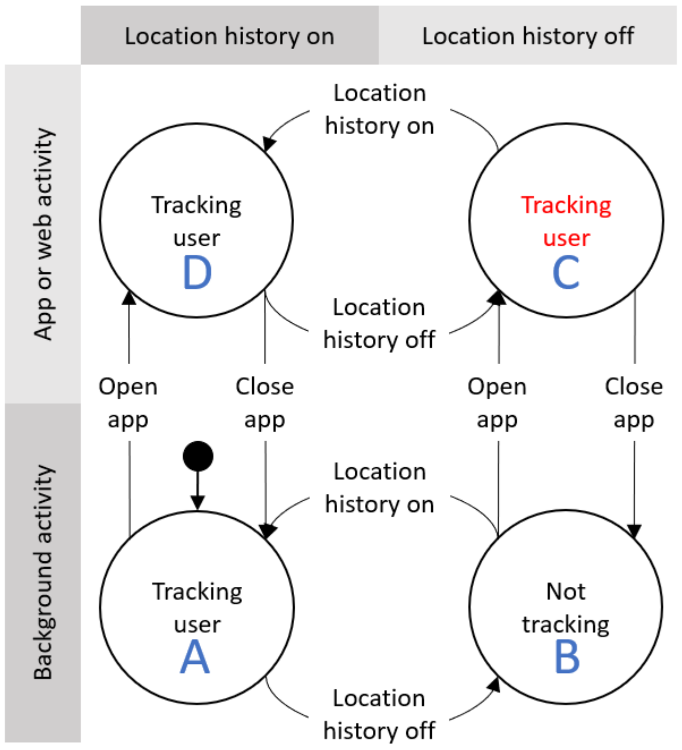


Figure 7.7: Finite State Machine (FSM) representation of Google tracking behavior and location history privacy preference setting.

We show in red the part of the system where the violation occurred, consisting in Google collecting data about the location of the user with location history setting being turned-off. Google legitimately tracks users in the states A and D, when location history is turned-on, but also in state C, illegitimately, when location history is turned-off, violating the privacy preference of the user.

The FSM can be used to generate realistic data and consent/preference logs. More in detail, we focus on a privacy preference that affects future collections of user location data. In the following formalism we will refer to Web, App, and Background as recipients of the location data collection. The location of the user is collected by the app in the states D and C, and this generates a data collection log entry of the form $locationData \in T \cap Location \cap (Web \cup App)$. While, in state A, the collection of user location generates a data collection log entry of the form $locationData \in T \cap Location \cap (Background)$. The events "Location history on/off", between states A and B and between states C and D, generate a preference log entry, in the case preference is turned-off $T \cap Location \subseteq Background$, while, in the case location history is turned-on $T \cap Location \subseteq (Web \cup App \cup Background)$.

Table 7.3 shows an example of a realistic log generated with the use of the FSM of Figure 6. In square brackets is the current state of the FSM. Initially, at line 1 location history setting is turned on by default and location of the user is collected in background. Then, at line 2, location history is turned-off by the user. However, at line 3, location data are still collected when using web or local apps.

Red color in Line 2 in Table 7.3 shows the violation of Google in applying the location history setting, with respect to how it is presented to the user. When location history is turned-off, Google still tracks the user in the case of web or apps activity.

Line	FSM state	Log
1	[A]	$T1 \cap Location \subseteq Web \cup App \cup Background$
		$locationData1 \in T1 \cap Location \cap Background$
2	[B]	Google applies: $T2 \cap Location \subseteq Web \cup App$
		User thinks: $T2 \cap Location \subseteq \emptyset$
3	[C]	$locationData2 \in T2 \cap Location \cap App$

Table 7.3: Google location tracking

A violation is detected by the framework with respect to preference of the user. In the example of Table 7.3, $locationData2 \in T2 \cap Location \cap App$ is not valid because user preference require $T2 \cap Location \subseteq \emptyset$, which would implies $locationData2 \in \emptyset$, which is a non-valid individual.

7.5 Discussion

Consent, under the GDPR, can be withdrawn at any time by the user. When this happens, companies must quickly react by stopping the processing of user data and blocking any further access to such data. Data collected under previous consent may still be accessed if consent is withdrawn non-retroactively or if there is another valid legal basis or a different consent that covers that same data. The GPDR provides the right to be forgotten, which is similar to withdrawing consent at any time. Recall that retroactive withdrawal is a stronger protection than non-retroactive withdrawal, because retroactive withdrawal removes any past authorizations to access data previously collected. This effectively means the previously collected data can still reside in an organizations database, but it is no longer accessible under the consent that was withdrawn. The right to be forgotten is a stronger protection than retroactive withdrawal, because the users can now ask for their data to be deleted, in addition to the halting of all processing activities. Companies must delete all user data and "forget" anything related to said user. In the case of data collected under a different legal basis than consent, for example, in the case of public information whose collection did not require an explicit consent of the user, the right to be forgotten can also be used to delete that data. If data are accessed, used and shared for multiple purposes, all these activities must stop and the same is true for third parties, who are asked to delete the data and stop further processing activities of the user.

The FSM used to simulate data can be used to represent how confusing layouts can affect the probability of users to opt-in/out from privacy preferences. In a report from the Norwegian Consumer Councils, FSM have been used to show how standard setting, cunning design choices, confusing layout, and illusion of choice, are used by Facebook and Google to actually give users no choice about their privacy.

7.6 Conclusion

We presented a framework for the representation and analysis of evolving consent to support companies in better understanding how consent changes affect their ability to access data in a consent-based and compliant manner. This work is an attempt to show the complexity of consent evolution and demonstrate the ability of description logics in modeling this. Future work includes a further evaluation using real case studies. We also plan to study how user consent- and preference-choices could be affected by their perception of privacy mitigations. Where users may be more concerned about not using specific data types for specific purposes, companies may want to separate those data types into separate user consent to avoid having users opt-out entirely. An additional direction of research is about analyzing authorizations for specific purposes and forecasting for potential privacy violations. Finally, we plan to use FSMs to compute precise probabilities of users to opt-in/out from privacy settings to further reason over the implications of multiple consent scenarios on data availability for statistical inference.

Chapter 8

Discussion, Conclusion and Future Work

This Thesis propose a privacy framework for consent that includes: (i) a comprehensive method for the compliance of complex systems with privacy regulations, based on modeling and analysis; and (ii) a work on consent evolution which includes a formal framework for the representation and analysis of data collection and access logs in a multiple consent granting and revocation scenarios.

The research contributions of this Thesis includes (i) a tool-supported method with guidelines in the form of a process that guides analyst in the design of compliant systems; (ii) a goal-oriented modeling language and a reasoning framework for the analysis of consent and privacy requirements and a first evaluation based on a real case study in the medical domain; (iii) a business process modeling language with privacy annotations and a reasoning framework for the analysis of processes with respect to privacy regulation principles and custom constraints, a formative evaluation and scalability evaluation; (iv) a framework for the representation and analysis of evolving consent and its validation based on realistic scenarios simulated with the use of FSM.

The findings include answers to the research questions as follows:

RQ1 What are the socio-technical aspects of a system that must be considered to comply with privacy regulations?

To support the compliance with privacy regulations, the analysis of a system should include aspect about the sharing and use of personal data. A socio-technical model include many aspects regulated to by privacy laws, including dependency between participants, document transmission, and information authorization. Processes include additional aspects that can be considered to comply with privacy regulations. The procedural aspect that is missed in the socio-technical model, can be integrated by refining goal models into business processes.

RQ2 Given socio-technical aspects regulated by privacy laws, what are the properties of a socio-technical model that verify the compliance with regulations?

The analysis of compliance can be supported by verifying properties that are inspired to principles given by regulations. Privacy principles can be interpreted and described as properties of the models, both at the socio-technical level and at the business process level.

RQ3 How to represent, graphically and formally, privacy and consent requirements?

A multi-view representation represents different aspects of a unique model into separate diagrams. When representing privacy requirements of a complex system, multi-view representation can provide a way to simplify the diagrams while still represent many different aspects, from goals, through information, to business processes. To support automated analysis, modeling languages can be defined into reasoning frameworks by formal languages, such as, set theory or logic languages, including disjunctive logic and description logic.

RQ4 Who are target users of the method and what process should they follow?

A tool-supported method provides guidelines for the system analysts, for example, in the form of a process, so to guide them in the analysis of requirements to design a system in compliance with privacy regulations. Different users participate in different phases of the process and contribute each with his own competences. To design a system in compliance with privacy regulations, a lot of competences are required, therefore, a comprehensive compliance method considers many target users, including system analysts, domain experts, lawyers, and privacy experts. An iterative process creates a collaborative environment in which experts continuously share their knowledge and contribute in the refinement of the system specifications.

Future work

Future work includes extensions of the method to consider other privacy requirement beyond those related to consent. This will require a further in-depth analysis of the GDPR and other privacy regulations.

We plan to propose the use of goal models to describe effects of consent choices and privacy preferences in complex environments where many parties are involved and many data are collected and used, such as, websites or mobile apps where data are often collected by third parties for marketing purposes.

Another direction of research consists in the development of techniques for the identification of data breaches, based on the analysis of critical data flow in business process models. This work may require to extend the modeling language to include new elements. Additionally it will require a re-visitation of the reasoning framework.

Last topic is on consent evolution, where we plan to rework the framework to include and implement a simulation environment to test the framework with realistic data logs and consent events.

We are also working toward the development of a comprehensive supporting tool that includes both the modeling editors for goal-based models and business processes models. We plan a further evaluation of the whole method when the tool will be available based on real case studies and by involving domain practitioners.

List of Figures

Bibliography

[1] DLVSYSTEM S.r.l. | DLV.

[2] Driver's Privacy Protection Act of 1994. 1994.

[3] Mark S Ackerman, Lorrie Faith Cranor, and Joseph Reagle. Privacy in e-commerce: examining user scenarios and privacy preferences. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 1–8. ACM, 1999.

[4] Alessandro Acquisti, Laura Brandimarte, and George Loewenstein. Privacy and human behavior in the age of information. *Science*, 347(6221):509–514, 2015.

[5] James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[6] Asim Ansari, Skander Essegaier, and Rajeev Kohli. Internet recommendation systems, 2000.

[7] Annie I Anton. Goal-based requirements analysis. In *Proceedings of the second international conference on requirements engineering*, pages 136–144. IEEE, 1996.

[8] Annie I Antón and Julia B Earp. Strategies for Developing Policies and Requirements for Secure Electronic Commerce Systems. Technical report, 2000.

[9] Alessandro Artale and Enrico Franconi. A survey of temporal extensions of description logics. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):171–210, 2000.

[10] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, and Matthias Schunter. Enterprise privacy authorization language (epal). *IBM Research*, 2003.

[11] Paul Ashley, Satoshi Hada, Günter Karjoth, and Matthias Schunter. E-p3p privacy policies and privacy authorization. In *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, pages 103–109. ACM, 2002.

[12] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In *Handbook on ontologies*, pages 3–28. Springer, 2004.

[13] Adam Barth, Anupam Datta, John C Mitchell, and Helen Nissenbaum. Privacy and contextual integrity: Framework and applications. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 15–pp. IEEE, 2006.

[14] Michael JA Berry and Gordon S Linoff. *Data mining techniques: for marketing, sales, and customer relationship management.* John Wiley & Sons, 2004.

[15] Travis D Breaux, Hanan Hibshi, and Ashwini Rao. Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. *Requirements Engineering Journal*, (3):281–307.

[16] Travis D Breaux, Hanan Hibshi, and Ashwini Rao. Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. *Requirements Engineering*, 19(3):281–307, 2014.

[17] Travis D Breaux, Daniel Smullen, and Hanan Hibshi. Detecting repurposing and over-collection in multi-party privacy requirements specifications. In *2015 IEEE 23rd international requirements engineering conference (RE)*, pages 166–175. IEEE, 2015.

[18] Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, and Anna Perini. Tropos: an Agent-Oriented Software Development Methodology. *JAAMAS*, 8(3):203–236, 2004.

[19] Carole Cadwalladr and E Graham-Harrison. The cambridge analytica files. *The Guardian*, 21:6–7, 2018.

[20] Carole Cadwalladr and Emma Graham-Harrison. Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian*, 17, 2018.

[21] Lorrie Cranor. *Web privacy with P3P.* " O'Reilly Media, Inc.", 2002.

[22] Fabiano Dalpiaz, Elda Paja, and Paolo Giorgini. *Security requirements engineering: designing secure socio-technical systems.* MIT Press, 2016.

[23] Remco Dijkman, Marcello La Rosa, and Hajo A Reijers. Managing Large Collections of Business Process Models-Current Techniques and Challenges. *Computers in Industry*, 63(2):91–97, 2012.

[24] Marlon Dumas, Marcello La Rosa, Jan Mendling, Hajo A Reijers, et al. *Fundamentals of business process management*, volume 1. Springer, 2013.

[25] Benjamin Edelman. Adverse selection in online "trust" certifications and search results. *Electronic Commerce Research and Applications*, 10(1):17–25, 2011.

[26] 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.

[27] Batya Friedman, Peter H Khan Jr, and Daniel C Howe. Trust online. *Communications of the ACM*, 43(12):34–40, 2000.

[28] Simson L. Garfinkel. De-Identification of Personal Information. Technical report, 2015.

[29] Paolo Giorgini, Fabio Massacci, John Mylopoulos, and Nicola Zannone. Modeling security requirements through ownership, permission and delegation. In *Proceedings of 13th IEEE International Conference on Requirements Engineering.*, pages 167–176. IEEE, 2005.

[30] Paolo Giorgini, John Mylopoulos, and Roberto Sebastiani. Goal-oriented requirements analysis and reasoning in the tropos methodology. *Engineering Applications of AI*, 18(2):159–171, 2005.

[31] Seda Gürses, Carmela Troncoso, and Claudia Diaz. Engineering privacy by design. 2011.

[32] Qingfeng He, Annie I Antón, et al. A framework for modeling privacy requirements in role engineering. In *Proc. of REFSQ*, volume 3, pages 137–146, 2003.

[33] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *Management Information Systems Quarterly*, 28(1):6, 2008.

[34] Jerry R Hobbs and Feng Pan. Time ontology in owl. *W3C working draft*, 27:133, 2006.

[35] Jaap-Henk Hoepman. Privacy design strategies. In *IFIP International Information Security Conference*, pages 446–459. Springer, 2014.

[36] S Ingolfo, A Siena, and J Mylopoulos. Goals and Compliance in Nomos 3. *International Conference on Conceptual Modeling*, pages 275–288, 2014.

[37] Christos Kalloniatis, Evangelia Kavakli, and Stefanos Gritzalis. Addressing privacy requirements in system design: the PriS method. *Requirements Engineering*, 13.3:241–255, 2008.

[38] Michel CA Klein and Dieter Fensel. Ontology versioning on the semantic web. In *SWWS*, pages 75–91, 2001.

[39] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The dlv system for knowledge representation and reasoning. *ACM Transactions on Computational Logic (TOCL)*, 7(3):499–562, 2006.

[40] Hector J Levesque and Ronald J Brachman. Expressiveness and tractability in knowledge representation and reasoning 1. *Computational intelligence*, 3(1):78–93, 1987.

[41] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *IEEE 23rd International Conference on Data Engineering*, pages 106–115, 2007.

[42] Jialiu Lin, Bin Liu, Norman Sadeh, and Jason I Hong. Modeling users' mobile app privacy preferences: Restoring usability in a sea of permission settings. In *10th Symposium On Usable Privacy and Security ({SOUPS} 2014)*, pages 199–212, 2014.

[43] Lin Liu, Eric Yu, and John Mylopoulos. Analyzing security requirements as relationships among strategic actors. In *Submitted to the Symposium on Requirements Engineering for Information Security (SREIS'02), Raleigh, North Carolina*, 2002.

[44] Lin Liu, Eric Yu, and John Mylopoulos. Security and privacy requirements analysis within a social setting. In *Proceedings. 11th IEEE International Requirements Engineering Conference*, pages 151–161. IEEE Comput. Soc, 2003.

[45] Lin Liu, Eric Yu, and John Mylopoulos. Security and privacy requirements analysis within a social setting. In *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*, pages 151–161. IEEE, 2003.

[46] Ashwin Machanavajjhala, Daniel Kifer, and Johannes Gehrke. l-Diversity: Privacy Beyond k-Anonymity. *ACM Trans. Knowl. Discov. Data*, 1(52), 2007.

[47] Aleecia M McDonald and Lorrie Faith Cranor. The cost of reading privacy policies. *Isjlp*, 4:543, 2008.

[48] Tim Moses et al. Extensible access control markup language (xacml) version 2.0. *Oasis Standard*, 200502, 2005.

[49] Haralambos Mouratidis, Paolo Giorgini, and Gordon Manson. Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems. Technical report, 2003.

[50] John Mylopoulos, Lawrence Chung, and Brian Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on software engineering*, 18(6):483–497, 1992.

[51] Natasha Noy, Alan Rector, Pat Hayes, and Chris Welty. Defining n-ary relations on the semantic web. *W3C working group note*, 12(4), 2006.

[52] Elda Paja, Fabiano Dalpiaz, and Paolo Giorgini. Modelling and reasoning about security requirements in socio-technical systems. *Data & Knowledge Engineering*, 98:123–143, 2015.

[53] Elda Paja, Fabiano Dalpiaz, Mauro Poggianella, Pierluigi Roberti, and Paolo Giorgini. Specifying and reasoning over socio-technical security requirements with sts-tool. In *ER*, pages 504–507. Springer, 2013.

[54] Jaehong Park and Ravi Sandhu. The ucon abc usage control model. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):128–174, 2004.

[55] European Parliament and Council of the European Union. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Union*, L119/59, 2016.

[56] Marco Robol. Proprivacy v1. *Mendeley Data*, doi:10.17632/sm9p7jk383.1, 2019.

[57] Marco Robol, Travis D Breaux, Elda Paja, and Paolo Giorgini. Consent verification under evolving privacy policies. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 422–427. IEEE, 2019.

[58] Marco Robol, Travis D Breaux, Elda Paja, and Paolo Giorgini. Consent verification monitoring. 2020.

[59] Marco Robol, Travis D Breaux, Elda Paja, and Paolo Giorgini. Toward gdpr-certified business processes. 2020.

[60] Marco Robol, Elda Paja, Mattia Salnitri, and Paolo Giorgini. Modeling and reasoning about privacy-consent requirements. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 238–254. Springer, 2018.

[61] Marco Robol, Mattia Salnitri, and Paolo Giorgini. Toward gdpr-compliant socio-technical systems: modeling language and reasoning framework. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 236–250. Springer, 2017.

[62] Mattia Salnitri. *Secure Business Process Engineering: a socio-technical approach.* PhD thesis, University of Trento, 2016.

[63] Mattia Salnitri, Fabiano Dalpiaz, and Paolo Giorgini. Designing secure business processes with secbpmn. *Software & Systems Modeling*, pages 1–21, 2015.

[64] Mattia Salnitri, Elda Paja, and Paolo Giorgini. Maintaining Secure Business Processes in Light of Socio-Technical Systems' Evolution. In *EEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 155–164. IEEE, sep 2016.

[65] Mattia Salnitri, Elda Paja, Mauro Poggianella, and Paolo Giorgini. Sts-tool 3.0: Maintaining security in socio-technical systems. In *CAiSE Forum*, pages 205–212, 2015.

[66] Florian Schaub, Rebecca Balebako, Adam L Durity, and Lorrie Faith Cranor. A design space for effective privacy notices. In *Eleventh Symposium On Usable Privacy and Security ({SOUPS} 2015)*, pages 1–17, 2015.

[67] Alberto Siena, Ivan Jureta, Silvia Ingolfo, Angelo Susi, Anna Perini, and John Mylopoulos. Capturing variability of law with nómos 2. *ER*, 7532:383–396, 2012.

[68] Alberto Siena, Anna Perini, Angelo Susi, and John Mylopoulos. A Meta-Model for Modelling Law-Compliant Requirements. In *Requirements Engineering and Law (RELAW)*, pages 45–51.

[69] Alberto Siena and Angelo Susi. *Engineering Law-Compliant Requirements - The Nomos Framework.* PhD thesis, University Of Trento, 2010.

[70] Daniel J. Solove. A Taxonomy of Privacy. 2005.

[71] Daniel J. Solove. Introduction: Privacy self-management and the consent dilemma. *Harv. L. Rev.*, 126(7):1880, 2012.

[72] S. Spiekermann and L.F. Cranor. Engineering Privacy. *IEEE Transactions on Software Engineering*, 35(1):67–82, jan 2009.

[73] Sarah Spiekermann, Jens Grossklags, and Bettina Berendt. E-privacy in 2nd generation e-commerce: privacy preferences versus actual behavior. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 38–47. ACM, 2001.

[74] Myra Spiliopoulou. Web usage mining for web site evaluation. *Communications of the ACM*, 43(8):127–134, 2000.

[75] Latanya Sweeney. k-Anonymity: a Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[76] Maximilian Teltzrow and Alfred Kobsa. Impacts of user privacy preferences on personalized systems. In *Designing personalized user experiences in eCommerce*, pages 315–332. Springer, 2004.

[77] the 104th United States Congress. The Health Insurance Portability and Accountability Act of 1996. *Pub. L. 104-191. Stat. 1936.*, 1996.

[78] the 105th United States Congress. Children's Online Privacy Protection Act of 1998. *Pub. L. 105-277*, 1998.

[79] the 106th United States Congress. Gramm–Leach–Bliley Act of 1999. *Pub. L. 106–102. Stat. 1338.*, 1970.

[80] the 91st United States Congress. Fair Credit Reporting Act of 1970. *Pub. L. 91-508. Stat. 1114-2.*, 1970.

[81] Axel Van Lamsweerde, Robert Darimont, and Emmanuel Letier. Managing conflicts in goal-driven requirements engineering. *IEEE transactions on Software engineering*, 24(11):908–926, 1998.

[82] Axel Van Lamsweerde and Emmanuel Letier. Handling obstacles in goal-oriented requirements engineering. *IEEE Transactions on software engineering*, 26(10):978–1005, 2000.

[83] Chris Welty, Richard Fikes, and Selene Makarios. A reusable ontology for fluents in owl. In *FOIS*, volume 150, pages 226–236, 2006.

[84] Eric Yu. Modelling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering*, 11:2011, 2011.

[85] Nicola Zeni, Elias A.Seid, SIlvia Ingolfo, and John Mylopoulos. Building Large Models of Law with NòmosT. *Conceptual Modeling ER*, pages 1–17, 2017.

Appendices

Appendix A

DLV implementation of ProPrivacy

A.0.1 DLV representation of a BPMN 2.0 Business Process

In the DLV implementation of the formal framework, business processes are described using predicates `participant`, `activity`, `dataObject`, `event` for atomic elements, `controlFlow` for sequence flow associations, `writeDO` and `readDO` for data associations, and `executed` for executor associations.

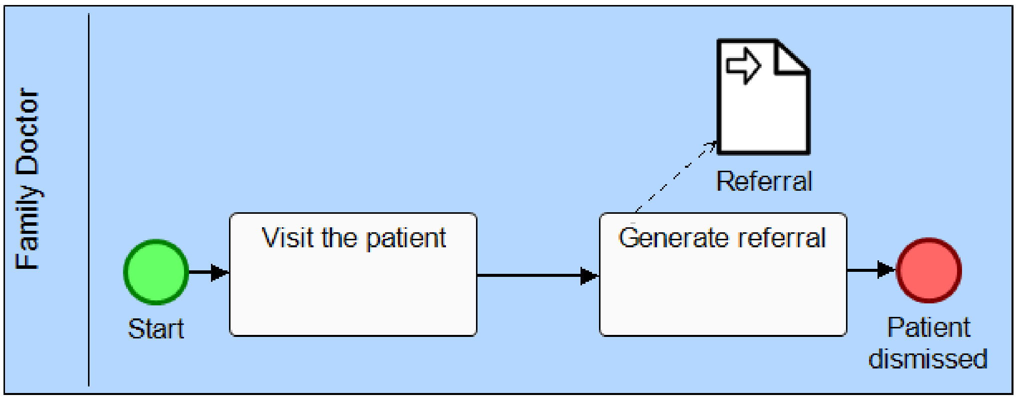


Figure A.1: Excerpt of a BPMN 2.0 diagram from the case study

Figure A.1 shows a BPMN 2.0 example diagram from the case study. In this small sub-process, the family doctor visits a patient and generate a referral before dismissing the patient.

Listing A.1: Example of representation of a business process in DLV

```
participant(FamilyDoctor).
activity(GenerateReferral).
dataObject(Referral).
```

```
4 event(AppointmentStart).
5 flow(AppointmentStart,VisitThePatient).
6 flow(VisitThePatient,GenerateReferral).
7 writeDO(GenerateReferral,Referral).
8 executed(VisitThePatient,FamilyDoctor).
```

Listing A.1 shows a sub-part of the DLV code that represent the process in Figure A.1. Facts in lines from 1 to 5 declare the elements of the process, lines 6 and 7 represent the sequence flow associations from *AppointmentStart* to *VisitThePatient* activities and from *VisitThePatient* to *GenerateReferral* activities, line 8 represents the data association from *GenerateReferral* activity to *Referral* data object, and line 9 represents the executor association from activity *VisitThePatient* to participant *FamilyDoctor*.

A.0.2 DLV representation of authorizations of consent

In DLV, the predicate `consent` is used to declare a consent, while predicate `auth` is used to declare an authorization part of a consent, where attributes represent respectively the consent element, the data object, the activity, and the authorized privacy operation.

Listing A.2: Example of privacy policies in DLV

```
1 consent(docC).
2 auth(docC,referral,uploadRef,dissemination).
3 consent(apssC).
4 auth(apssC,referral,medicalExam,collection).
5 auth(apssC,results,uploadRes,processing).
```

Listings A.2 shows the DLV code for the authorizations of the two consents from Tables 6.2 and 6.3. For example, authorization `auth( docC, referral, uploadRef, dissemination )` from line 2, expresses that the family doctor consent *docC* authorizes the dissemination of the *referral* data object in the execution of the activity *uploadRef*.

A.0.3 DLV inference rules for business process walks

In DLV, walks of a business process are inferred starting from a start event and following control flow associations. The predicate `walk` is used to represent the existence of a walk between two elements of the process.

Listing A.3: DLV walk inference rules based on a two-attributes predicate

```
walk(X,X) :- start(X).
walk(X,K) :- walk(X,Y), flow(Y,K).
```

Listing A.3 shows DLV rules used to infer business process walks. Line 1 initializes a new walk by start navigating a business process from a start event. Line 2 creates an extended versions of existing walks by following control flows.

For example, given the process of Figure A.1, `walk( AppointmentStart, GenerateReferral)` represents the existence of a walk from the event *AppointmentStart* to the activity *GenerateReferral*.

Another use of the predicate `walk` is in the case of three parameters, where the additional parameter is an element that is included in the walk.

Listing A.4: DLV walk inference rules based on a three-attributes predicate

```
1 walk(X,X,X) :- start(X).
2 walk(X,Y,K), walk(X,Z,K) :- walk(X,Y,Z), flow(Z,K).
```

Listing A.4 shows DLV rules used to infer business process walks that goes through some specific element. Similarly to rules in Listing A.3, line 1 initializes a new walk, while line 2 creates an extended versions of existing walks.

For example, given the process of Figure A.1, `walk( AppointmentStart, VisitThePatient, GenerateReferral)` represents the existence of a walk from the event *AppointmentStart* to the activity *GenerateReferral*, through the element *VisitThePatient*.

A.0.4 DLV implementation of the minimization requirement

Listing A.5: DLV implementation of minimization property

```
1 excessiveAuth(A) :- auth(C,M,A,collection), activity(A1), not
     messageFlow(A1,A,M).
2 excessiveAuth(A) :- auth(C,M,A,dissemination), activity(A1), not
     messageFlow(A,A1,M).
3 excessiveAuth(A) :- dataObject(DO), auth(_,DO,A,processing), not
     readDO(A,DO), not writeDO(A,DO).
4 excessiveAuth(A) :- message(M), auth(C,M,A,processing), activity(
     A1), not messageFlow(A1,A,M).
```

Listing A.5 shows the DLV implementation for the verification of the data minimization property. **Line 1** implements minimization of collection. An excessive authorization exception is thrown in the case that collection is authorized but no message is received. **Line 2** implements minimization of dissemination. An excessive authorization exception is thrown in the case that dissemination is authorized but no message is sent. **Line 3 and 4** implement minimization of processing. In the case of a data object, an excessive authorization exception is thrown in the case that processing is authorized but no data object is read or produced. In the case of a message flow, an excessive authorization exception is thrown in the case that processing is authorized but no message is sent.

A.0.5 DLV implementation of the freely given consent requirement

Listing A.6: DLV rules to verify the existence of some process execution trace that does not require any authorization

```
1 -free(A) :- element(A), auth(_,_,A,_).
2 free(A) :- element(A), not -free(A).
3 freeExec(FROM,FROM) :- start(FROM),        free(FROM).
4 freeExec(FROM,TO) :- freeExec(FROM,X),      flow(X,TO), free(TO),
                      not -freeExec(FROM,TO).
5 -freeExec(FROM,TO) :- start(FROM),         not freeExec(FROM,X),
     flow(X,TO), parallelJoin(TO).
6 freeExec(FROM) :- start(FROM), end(TO), freeExec(FROM,TO).
7 freelyGivenException :- start(FROM),       not freeExec(FROM).
```

Listing A.6 shows the DLV implementation for the verification of the freely-given consent property. **Lines 1 and 2** infer elements that does not require any consent, identified with the predicate `free`, and those who does require some consent, −`free`. **Line 3** initialize BPMN 2.0 execution rules considering only elements free from any consent. **Line 4** propagate execution on following elements **Line 5** implements BPMN 2.0 execution rules in the case of join elements. **Line 6** terminates the execution of the process when an and event is reached. **Line 7** require a free process execution for each start event.

A.0.6 DLV implementation of the consent specificity requirement

Listing A.7: DLV implementation to verify the consent specificity property

```
 1 nonConsentSpecific(A) :- auth(C,_,A,_), auth(C2,_,A,_), C2!=C.
 2 freeOrConsentSpecific(C,A) :- auth(C,_,A,_), not
      nonConsentSpecific(A).
 3 freeOrConsentSpecific(C,A) :- consent(C), free(A).
 4 forward(X,C) :- start(X), freeOrConsentSpecific(C,X).
 5 forward(Y,C) :- forward(X,C), flow(X,Y), freeOrConsentSpecific(C,Y
      ), not -forward(Y,C).
 6 -forward(Y,C) :- freeOrConsentSpecific(C,X), flow(X,Y), not
      forward(X,C), parallelJoin(Y).
 7 backward(X,C) :- forward(X,C), end(E).
 8 backward(X,C) :- backward(Y,C), flow(X,Y), forward(X,C).
 9 specific(X) :- forward(X,C), backward(X,C).
10 specificityException(A) :- auth(_,_,A,_), not specific(A).
```

Listing A.7 reports DLV rules used to verify the existence of a consent-specific process-execution-trace for each activity. **Lines 1, 2, and 3** infer activities that either do not require any consent or that require only one consent. Following lines apply BPMN 2.0 execution rules to explore the process by considering only one consent at a time. **Line 4** initializes execution from start node. **Line 5** propagates execution to the following nodes. **Line 6** implements join gateways. **Line 7** finalizes the execution when an end node is reached. **Line 8** propagates the finalized execution on previous nodes. **Lines 9 and 10** throw a `specificityException` for activities that do not satisfy the consent specificity property, meaning that are not part of any free or consent-specific execution trace.

A.0.7 DLV implementation of the ProPrivacy custom requirements framework

Listing A.8: Disjunctive rules implementing BPMN 2.0 execution rules

```
1  step(1,X) v -step(1,X) :- start(X).          % Step 1
2  :- step(1,X), step(1,Y), X!=Y.                % At most one start
3  started :- step(1,X). :- not started.        % At least one start
4  ended :- end(X), step(N,X). :- not ended.    % At least one end
5  step(M,Y) :- step(N,X), #succ(N,M), flow(X,Y), not alternativeFork
      (X), not parallelJoin(X).                              % All if no
      alternatives
6  step(M,Y) v -step(M,Y) :- step(N,X), #succ(N,M), flow(X,Y),
      alternativeFork(X).      % Alternatives
7  step(X) :- flow(X,Y), step(N,Y).
8  :- flow(X,Y), not step(X).                    % At least one
      alternative
9  :- exclusiveGateway(G), flow(G,X), flow(G,Y), step(N,X), step(N,Y)
      , X!=Y.            % At most one in case of exclusive
10 -joined(G) :- parallelJoin(G), flow(X,G), not step(N,X).
11 step(M,G) :- parallelJoin(G), not -joined(G), flow(X,G), step(N,X)
      , #succ(N,M).      % Join all in case of parallel
```

Listing A.8 shows DLV rules used to generate process execution traces, which are based on BPMN 2.0 execution rules. Here a description:

- Create a trace for each start event in which this is executed as the first step.

  DLV rules: (line 1) For each start event, create two traces, one in which it is executed as the first step, the other one in which it is not executed; (line 2) Allow at most one start to be executed at step 1; (line 3) Require at least one start to be executed at step 1.

- Continue until an end event is executed.

  DLV rule: (line 4) Require at least an end event to be executed at some step.

- In the case of an element followed by a single control flow or in the case of a parallel gateway, execute all the next elements.

  DLV rule: (line 5) In a new step execute all the activities that follow an element executed at current step, that are not alternative fork gateways or parallel join gateways.

- In the case of alternative gateways duplicate the execution trace for each exiting control flow or a combination of those.

  DLV rules: (line 6) In the case of alternative gateways two execution traces are created for each exiting control flow; (lines 7 and 8) Require at least one control flow to be taken; (line 9) Allow at most one control flow to be taken in the case of exclusive alternative gateways.

- In the case of a parallel join gateway, execute next element only in the case that all previous elements are executed at some previous step.

  DLV rules: lines 10 and 11.

Listing A.9: Example of custom constraint that requires the existence of a process execution trace that includes the activity `a1`

```
:- not step(a1).
```

Listing A.9 shows an example of a custom requirements to force the execution of a specific activity. The implementation consists of a DLV constraint that invalidate all traces that does not include the execution of the the activity `a1`.