Softw Syst Model (2011) 10:369–394
DOI 10.1007/s10270-010-0154-z

A framework to support alignment of secure software engineering with legal regulations

Shareeful Islam · Haralambos Mouratidis · Jan Jürjens

Received: 31 January 2009 / Revised: 7 December 2009 / Accepted: 15 February 2010 / Published online: 9 March 2010

Abstract Regulation compliance is getting more and more important for software systems that process and manage sensitive information. Therefore, identifying and analysing relevant legal regulations and aligning them with security requirements become necessary for the effective development of secure software systems. Nevertheless, Secure Software Engineering Modelling Languages (SSEML) use different concepts and terminology from those used in the legal domain for the description of legal regulations. This situation, together with the lack of appropriate background and knowledge of laws and regulations, introduces a challenge for software developers. In particular, it makes difficult to perform (i) the elicitation of appropriate security requirements from the relevant laws and regulations; and (ii) the correct tracing of the security requirements throughout the development stages. This paper presents a framework to support the consideration of laws and regulations during the development of secure software systems. In particular, the framework enables software developers (i) to correctly elicit security requirements from the appropriate laws and regulations; and (ii) to trace these requirements throughout the development stages in order to ensure that the design indeed supports the required laws and regulations. Our framework is based on existing work from the area of secure software engineering, and it complements this work with a novel and structured process and a well-defined method. A practical case study is employed to demonstrate the applicability of our work.

Keywords Secure software engineering · Non-functional properties · Security requirements · Secure Tropos · UMLsec · Modelling regulations · Legal constraints

1 Introduction

Important and sensitive information is stored in software systems, and as a result securing such systems has become a necessity rather than an option. Naturally, a large body of research has been dedicated to investigating the various challenges related to securing software systems and developing approaches to support the development of secure software systems (see [14] for an overview). As a result of such research, it has been argued that in order to develop more secure software systems, security needs to be considered from the early stages of the development process and software systems developers should analyse not only the system but also its environment [14,23]. This is important

Communicated by Marko Boškovic, Bernhard Schätz, Claus Pahl, and Dragan Gasevic.

The work is partly supported by the German Academic Exchange Service (DAAD) and by the EU project Secure Change (ICT-FET-231101).

S. Islam (✉)
Institut für Informatik, Technische Universität München,
Munich, Germany
e-mail: islam@in.tum.de

H. Mouratidis
School of Computing, IT and Engineering,
University of East London, London, UK
e-mail: haris@uel.ac.uk

J. Jürjens
Software Engineering (LS 14),
Department of Computer Science, TU Dortmund,
Dortmund, Germany
URL: http://jurjens.de/jan

J. Jürjens
Fraunhofer ISST, Dortmund, Germany

since all software systems operate within an environment and the various elements—stakeholders, users, relevant laws and regulations—of the environment might influence the security aspects of the system. As an example, consider a software system for the health-care sector. The system does not operate in isolation but within the environment of the health-care domain (in fact it might operate across a number of different domains). As a result, the security of the system might be influenced by its stakeholders (e.g. health-care service), its users (e.g. doctors, nurses, patients) and relevant laws and regulations (e.g. patient privacy laws). The latter is important due to the sensitivity of the information usually stored in various software systems and the need to ensure that software systems that contain sensitive and private information comply with the appropriate laws and regulations. As such, there is a need to develop software systems that, apart from meeting their security requirements, need to comply with relevant regulations and laws. This demand is not only driven by the relevant research, but most importantly by the current industrial and commercial needs. For example, it is estimated that in the health-care domain, organisations will spend over $17.6 billion over the next few years to align their systems and procedures with the Health Insurance Portability and Accountability Act (HIPAA) [28]. Similarly, in the business information systems domain, it is estimated that organisations will spend $5.8 billion in one year to ensure compliance with acts such as the Sarbanes–Oxley Act [30]. In fact, it is anticipated that almost all the domains where software systems are used will have to deal with a similar situation, where organisations must ensure compliance with relevant regulations (including relevant privacy directives, such as the EU directive for the EU member states and appropriate national privacy laws). As a result, the impact will be substantial for organisations that employ software to support their critical business process as well as manage sensitive information or for organisations that develop software systems. A number of those organisations are developing procedures and methodologies to support the development of secure software systems and the alignment of the system's security requirements with relevant regulations and laws. For example, the software company Microsoft has released an extensive set of privacy guidelines and incorporated them into their Microsoft Security Development Lifecycle (SDL) [35].

However, eliciting security requirements from relevant laws and regulations is a difficult task for a number of reasons. First of all, legal text is hard to interpret [34,41] by people without appropriate knowledge, such as software developers. This is due to issues such as ambiguity (when articles in regulations can be interpreted in several ways), cross reference (when one article from one section is related to another article from another section), and incompleteness (when regulations do not provide guidance for specific circumstances and/or situations). Secondly, concepts and terminology used to express laws and regulations are fundamentally different from those used to express software systems security requirements [15]. On one hand, laws are expressed in terms of rights, using concepts such as privilege, duty, liability and obligation, which control stakeholder behaviour within a specified context [43]. On the other hand, security requirements are expressed using concepts, such as stakeholder, goal, security constraint, threat, which aim to model system behaviour alongside a number of security related restrictions that ensure correctness, reliability and robustness of the software system [9]. Thirdly, the lack of maturity of the relevant laws and of the area of security requirements engineering introduces an extra challenge. Information security and data privacy laws are relatively new, unstable, and incomplete [34]. External events such as new security threats and vulnerabilities, as well as technological progress can trigger changes to the relevant laws. Similarly, the area of security requirements engineering is relatively new and there are still no widely agreed definitions for a number of concepts used, nor widely accepted methods and methodologies for security requirements elicitation and development [14].

Within the above context, the novel contribution of this paper is a framework that supports (i) the elicitation and analysis of security requirements from relevant regulations and laws; and (ii) the development of a design that satisfies these requirements. The presented framework is based on previous work from the areas of security requirements engineering [19,20] and model-based security engineering [23]. In particular, the framework supports the integration of the Secure Tropos methodology, to support the elicitation and analysis of security requirements, and UMLsec, to support the development of a design to satisfy the elicited security requirements. The selection of these two approaches is based on a number of reasons. First of all, the Secure Tropos methodology uses the same concepts and notations throughout the development process [19]. This is important in our work since it allows us to track specific security solutions to the various elicited security requirements. Moreover, the modelling language of the Secure Tropos methodology is easily extensible, so it provides us with an ideal vehicle to extend it, in order to support the modelling of the legal concepts necessary for our work. On the other hand, UMLsec is supported by a number of secure design diagrams that allow the analysis of appropriate security properties during the design stage. This is important in our work since it enable us to evaluate the developed solutions against a number of security properties identified on the relevant laws and regulations. Our framework has four main components: (i) a modelling component to support the interpretation of legal texts and the derivation of legal rights, correlatives, and legal constraints; (ii) an elicitation component to support elicitation of security-constraints and analysis of security requirements and related legal constraints; (iii) an analysis component to support threat and non-compliance analysis; and (iv) a design component to support the design of a system that fulfils the elicited and analysed security requirements. To demonstrate the applicability of our work and validate it within the legal domain, we consider a case-study based on the directives that focus on the information security law from the European Union (EU) Information Society Legislation [21] and the associate German national law that implements the directives [5]. We have selected the above since their details are publicly available and since they are the relevant directives and laws for the case studies we work on. However, we expect that our framework is suitable for the analysis of other national laws.

The paper is structured as follows. Section 2 provides a detailed description of related work for secure software system development considering regulations. Section 3 describes the components of our framework. Section 4 describes the development process supported by the presented framework, while Sect. 5 demonstrates the applicability of our proposed approach through a case-study based on an industrial context for service based software development. As mentioned above, in the case study we combine legal text from directives 95/46/EC and 2002/58/EC of the European Commission (EC) Information Society legislation [21] and the German Federal Data protection Act [5]. Finally Sect. 6 concludes the paper and presents directions for future work.

2 Related work

A number of researchers [3,12,14,16,23,29,31–33] argue for the need to consider security as early as possible during the development process. Towards this direction a number of research results have been presented in the literature. Chung et al. [26] apply a process-oriented approach to represent security requirements as potentially conflicting or harmonious goals. The proposed framework, which is called the NFR (Non-Functional Requirements) framework, represents and uses security requirements as a class of non-functional requirements and it allows developers to consider design decisions and relate these decisions to the represented non-functional requirements. MOQUARE (Misuse-Oriented Quality Requirement Engineering) [2] provides a framework to analyse Non-Functional Requirements, including security. The framework accepts as an input a functional description together with a system's business and quality goals (in which security is included). With the aid of misuse cases and NFR trees, the framework allows developers to analyse relevant NFRs. Similarly, IESE—NFR [1] is a fully prescribed method used to specify NFRs. Inputs are Functional Requirements, use cases, and the NFR tree, while the output of the analysis using the method is a list of needed NFRs, including security requirements. Mead et al. [29] propose the Security Quality Requirements (SQUARE) method for eliciting and documenting security requirements, whereas Haley et al. [6] provide an approach for security requirements elicitation, specification and analysis. Mellado et al. [10] present a Common Criteria based security requirements engineering process (SREP), where several Common Criteria constructs have been employed (e.g. security functional components, protection profile, and security assurance components) for eliciting, categorising, and prioritising security requirements. SREP is different from SQUARE in that it integrates the Common Criteria and Information Security standards such as ISO/IEC 27001 into the steps when eliciting security requirements. Mouratidis et al. [17] present Secure Tropos for eliciting security requirements in terms of security constraints. Islam et al. [37] present an approach related to security risk and its impact to software quality attributes and how eliciting security requirements at an early stage can address such attributes. Moreover, a framework for representation and analysis of security requirements is presented by Haley et al. in [7]. It allows the security engineer to represent and analyse security requirement. Mouratidis et al. [18] further extended Secure Tropos with the notion of security attack scenarios, where possible attackers, their attacks and system resources that can be attacked are modelled. A different line of work is based on the extension of the Unified Modelling Language (UML). Jürjens proposes UMLsec [23,24], an extension of the Unified Modelling Language (UML), to include the modelling of security related features, such as confidentiality and access control. Although all of these works are relevant to our work, they do not consider the analysis of legal texts when specifying security requirements.

On the other hand, a number of researchers have analysed privacy regulations to derive rights and obligations of actors to perform some actions [11,34,40–42]. Breaux et al. [40] consider activity, purpose, and rules to extract rights, obligations and constraints from the legal texts. Rights and obligations are the actions that stakeholder are permitted and required to perform. In this work, extracted rights and obligations are stated into restricted natural language statements to depict discrete activities, in an application context, based on a semantic parameterization process [40,42]. The approach was further extended to identify "allow" and "deny" rules and exceptions on these rules, which prioritize access rights and obligations relevant to access sensitive information of a system [41]. Islam et al. [38] employ the pattern based approach of Breaux et al. to extract constraints and to combine them with the policies that derive from the Information Security standard ISO/IEC 27001:2005. This approach contributed to the integration of these constraints into UMLsec models, along with relevant stereotypes, so that a design is developed that supports appropriate legal constraints. Darimont et al. [36] propose regulation modelling based on the Goal Oriented Requirement Engineering

(GORE) methodology by KAOS. In this work, regulation documents are incrementally transformed into goal, object, agent, and operation model. However, the short representation of the approach makes it difficult to understand how the transformation takes place in order to validate the approach. Ghanavati et al. [39] proposed a compliance framework based on a user requirements notation, which models business processes and links them to regulations. In this work, the *i** goal oriented requirement language is used to model goals and actions extracted from regulations. Moreover, the framework facilitates the establishment of a traceability link between law and requirements. Siena et al. [4] focused on Hohfeld's legal taxonomy as a basis for a systematic process to support decision making from regulations. May et al. [27] employed access control techniques to extract privacy related elements from legal texts.

These approaches discussed above are important but they also demonstrate a number of limitations. Most of these approaches consider just the permitted and required actions of a stakeholder. Nevertheless, according to the legal taxonomy by Hohfeld [43], considering only permitted and required actions is not sufficient to represent stakeholder legal rights. In contrast, our framework supports the analysis of legal text and the extraction of a number of concepts based on Hohfeld's legal taxonomy, such as legal privilege, claim, power, immunity and their correlatives duty, no-right, liability and disability. Moreover, the above mentioned approaches fail to specify a process where: (i) the interpretation of legal texts is aligned with the specified (security) requirements; (ii) requirements can be traced in the developed design; (iii) an analysis takes place to identify potential risks for non-compliance to a specific application context.

3 Components of the proposed framework

The proposed framework allows us to treat elicitation and analysis of requirements from regulations similarly to the goal analysis in requirements engineering. It also supports the development of a design that fulfils the requirements that are elicited. It contains four components:

1. A *Modelling Regulation* component, to support the interpretation of legal texts and the derivation of legal rights. This component supports the analysis and modelling of regulations based on the elementary concepts classified by Hohfeld [43]. It also supports the extraction of possible legal rights (e.g. claim, privilege, power, immunity) and their correlatives (e.g. duty, no-right, liability, disability), and the distinction of legal constraints, from the extracted legal rights and correlatives, with the help of a pattern based approach (described in next section), a common set of rules (also described in next section), and the secure Tropos language.
2. An *Elicit Security Requirements* component based on the Secure Tropos analysis techniques to support elicitation of security-constraints, and analysis of security requirements and legal constraints. In particular, based on the models derived from the previous component and a security analysis of the system under development, this component supports the alignment of the security requirements with relevant legal constraints.
3. A *Security Requirements Analysis* component based on a scenario-based analysis method, which supports the analysis of threat environment and non-compliance issues. In particular, potential security attacks, which might introduce non-compliance issues in the system under development, are identified by analysing malicious goals, task, and related threats of potential attackers. Such analysis enables software developers to further refine the system's security requirements and align them to relevant regulations in order to countermeasure potential regulation-based threats.
4. A *Secure System Design* component to support the design of a system that fulfils the identified security requirements and legal constraints. Based on the principles of UMLsec and Model Based Security Engineering (MBSE), this component enables software system developers to design the system in a way that its regulation-related security requirements can be traced back to the appropriate regulations.

Figure 1 graphically illustrates the components of the proposed framework along with required inputs and artefacts produced from the framework components. Moreover, the figure illustrates the foundational approach that we have used to support the specific component; and the development stage where each component is applied. In the following sections, we further describe each one of the framework's components.

3.1 Modelling regulation component

The first component aims to elicit legal constraints from relevant legal texts. The fundamental legal conceptions proposed by Hohfeld [43] classified legal rights in several elementary concepts including privilege, claim, power, immunity, duty, no-right, liability, and disability, which are organised in opposites and correlatives. These elementary concepts are adopted in our work to support the elicitation of legal constraints. The word legal right is often used broadly to cover legal relations. However legal thinking can never be truly accurate unless we distinguish different kinds of rights. Therefore, we have based our work on this classification of rights, because this provides an opportunity to precisely think about law. The taxonomy is grounded on the concept of *legal right*, which is one's assenting claim against another or a claim for an action. It can be defined broadly as entitlement to perform

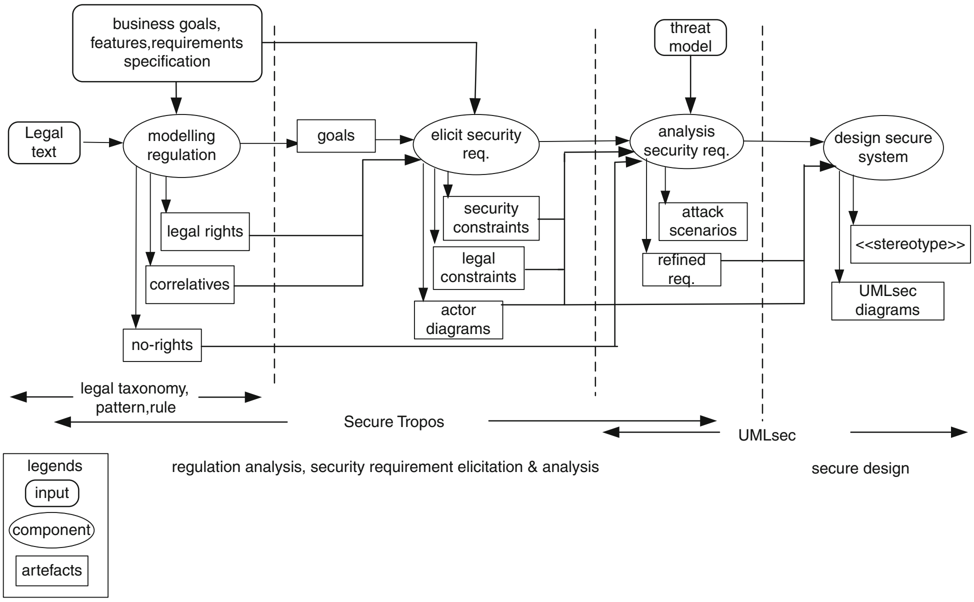

Fig. 1 Overview of the proposed framework

certain actions or be in certain states, or entitlement that others perform certain actions or be in certain states. The more perceptive mean of right is *claim*, which is the entitlement for a person to have something done by another person, who has therefore a *duty* of doing it. For instance, if Tom is holding a bank account and he raises a claim to view his account balance, the Bank has a duty to provide the data related to his bank account. *Power* is one's affirmative control over a given legal relation against another or over an action on an object. An actor holding power has the legal ability to perform certain actions or to alter legal relations within a context. *Privilege* is the entitlement of one's freedom from the right to discretionally perform an action or to claim for another [4,13,43]. Two rights are *correlatives* [43], if one right cannot exist without the other. Dealing with correlative duty, liability, and disability, requires looking at the situation from the involved actors' point of view. *Liability* is one's responsibility for carrying out a specific action. It differs from *duty* in the sense that duty does not require any legal power. For instance, when Tom provides personal data for opening a bank account, the bank has the liability to protect the data. On the other hand, *disability*, in the context of legal regulations, is an actor's lack of legal power to accomplish an action that another actor with legal power can accomplish. Dealing with correlative no-right, requires looking at two different situations from the point of view of the same person or action in all cases, irrespective if the person has the right or no-right. The term *no-right* denotes absence of right on the part of an actor to perform certain action. Finally, *opposition* means that the existence of a right excludes its opposite (in the same way as "claim" opposes "no-right" and "power" opposes "disability") [13]. The object of right specifies possible activities of an actor controlled from regulations [4]. Such activities describe behaviour of an actor or identify the purpose for an action. In requirements engineering, this identified purpose can be treated as a goal of an actor. The conceptual view of the legal rights to model the laws and regulations by modelling regulation is depicted in Fig. 6 (Sect. 4.1).

The fundamental legal concepts provide a high-level abstraction. However, a comprehensive relationship needs to be established among the involved stakeholders and their high level legal rights, so that regulations can control the stakeholders' behaviour. To accomplish this task, our framework correlates the above high level legal abstractions with abstractions derived from the security requirements engineering area, and in particular the Secure Tropos modelling language. This allows us to identify an actor's legal rights and correlatives by modelling regulations and align them with the security requirements of the system under development. To assist us in this process, we employ, as discussed above, natural language patterns and a set of rule sets so that regulations are analysed in terms of requirements engineering concepts such as goal, actor, task, and resource. Then, we derive an actor's legal rights and correlatives based on the appropriate requirements engineering concepts. The idea of natural language patterns has been extensively used to analyse regulations [40–42]. In our work we make use of the *activity* and *purpose* patterns presented in previous work [40] and introduce the *object* pattern to extract possible legal rights and their correlatives from legal text. This enables us to consider beyond permitted and required actions of the existing contributions and therefore derive a precise analysis of legal text within any application context. Below, we briefly describe these patterns:

- *Activity pattern:* The Activity pattern specifies possible actions, defined by legal text, for a subject who performs the action on an object within a specific context. The pattern identifies the actors (**who**) and possible actions (**what**) on an object (**whom**). These actions are the objects of actor's legal rights and correlatives. Therefore, the main focus of the pattern is to identify the behavioural and if possible, productive action of an actor so that the actor's necessary legal rights and correlatives can be distinguished. A behavioural action states the type of behaviour that an actor performs, while a productive action states the results produced by the behaviour performed by actor. For instance the sentence "John is studying hard" states John's behaviour and therefore it is treated as a behavioural action, while "John obtains good grades" is the result of the previous behaviour (i.e. the result of studying hard) and therefore it is considered a productive action. Nevertheless, depending on the situation the interpretation of the legal rights from the same legal text can be diverse. As such, this pattern facilitates the determination of the depending strategies among actors with possible actions from legal rights and correlatives on an object.
- *Purpose pattern:* Every action has an objective that an actor intends to achieve. This objective is the goal of the actor. This pattern mainly identifies the high level goals for an action performed by the actor. Furthermore it also identifies the goals derived from relevant regulation. Therefore the pattern provides the strategic rationale behind each action performed by an actor and the legal text that controls this action.
- *Object pattern:* The pattern describes the properties of an object by identifying object type, location and measuring parameters. Generally objects are the critical system assets such as data, physical device, and services of the system. An Actor depends upon this object to perform an action. Every object has a specific location from where it can be accessed or processed. For example, some personal data might be stored on a centralised server. Measuring parameters are the constraints from a legal text used to measure the safeguards of an object, such as risk level, security level, and technical measure, to assure sufficient protection of the object from any undesirable situations.

These patterns are combined with a common set of rules [40] to analyse and model regulations. Generally, these rules ease capturing behavioural action of an actor. The rules are:

- Subjects on legal texts are generally represented by nouns and indicate who performs an action;
- Verbs on legal texts usually represent actions such as access, disclose, process, transmit;
- Objects, usually found towards the end of a legal text, indicate different attributes such as object location and measuring parameters;
- Words such as *who*, *that* and *which* followed by verb distinguish nouns from the rest of the phrase;
- Normative phrases of legal text such as *shall, shall be, may*, *must*, and so on generally determine legal rights such as claim, power, duty, and obligation, which may be employed in strict sense depending on the application context. On the other hand, negation of the normative phrases, such as *shall not*, *may not*, *may not require, must not*, and so on, generally determine no-right and disability of actions;
- High level goals can be obtained by rationalising the reasons why the regulation is introduced as well as why an actor performed a specified action. Based on the object and its related attributes, an actor's dependency assumption on the object can be employed to identify the actor goal;
- Conditional keywords such as *if*, *except*, *unless,* and so on specify preconditions and post-conditions among legal rights and correlatives.

3.2 Elicit security requirements component

The second component of our framework supports the elicitation of security requirements and their alignment with appropriate regulations. Our analysis is based on the Secure Tropos analysis techniques, an extension of Tropos methodology. Tropos adopts the i* modelling framework [14,20], which uses the concepts of actors, goals, tasks, resources, and social dependencies for defining the obligations of actors (*depender*) to other actors (*dependee*). Secure Tropos introduces security related concepts (e.g. security constraint, secure dependency, secure goal) to the Tropos methodology, to enable developers to consider security issues throughout the development life cycle. A *security constraint* is defined in Secure Tropos as a restriction related to security issues, such as privacy or integrity, which influences the analysis and design of the software system under development by restricting some alternative design solutions, by conflicting with some of the requirements of the system, or by refining some of the system's objectives [20]. Effectively, security constraints represent in Secure Tropos the initial high level security requirements of a system and they are elicited from a number of sources including the stakeholders and users of the system as well as domain and security experts.

Secure dependencies introduce security constraint(s) that must be fulfilled for the dependency to be satisfied. Secure Tropos uses the term *secure entity* to describe any goals, tasks and resources related to the security of the system. A *secure goal* represents the strategic interests of an actor with respect to security. Secure goals are mainly introduced in order to achieve possible security constraints that are imposed to an actor or exist in the system. However, a secure goal does not particularly define how the security constraints can be achieved, since alternatives can be considered. The precise definition of how the secure goal can be achieved is given by a secure task. A *secure task* (also known as secure plan) is defined as a task that represents a particular way for satisfying a secure goal. A resource that is related to a security entity or a security constraint is defined as a *security resource*. As an example, consider Fig. 2, where three actors are modelled along with their dependencies. In Secure Tropos the presented model is known as *security enhanced actor model* and it is used in the early stages of the process to model the environment and the security requirements of the system under development. In particular, in this example, the *Patient* actor depends on the *Doctor* actor to satisfy the goal *Receive Appropriate Health Care*. However, for the *Doctor* actor to support the patient in that goal, the *Doctor* needs to *Obtain Patient Information*. However, relevant health care laws restrict the *Doctor* on sharing the information obtained. This is modelled using a security constraint *Share Information Only if Consent is Obtained*. Moreover, according to the methodology each of the actors is further analysed. In this example, we provide a very short analysis of the *Doctor* actor, according to which the main goal of the *Doctor* is to *Develop a Care Plan* for the patient. In doing so the *Doctor* needs to consult various members of the medical team. However, as derived from the previous analysis, consent should be obtained to share any patient information. Therefore a secure goal is imposed to the *Doctor* actor to *Obtain Patient Consent* and an example of a secure task is provided on how this can be achieved. Later in the process the results of this analysis is input to the analysis of the system to determine security constraint from the users and stakeholders of the system. Detailed description about Secure Tropos and further examples—based on real-life case studies—can be found in [17].

In the framework presented here, we propose to map legal constraints (restricted legal rights and correlatives) with the Secure Tropos security constraints so that security requirements that are elicited can also address legal concepts. This facilitates to identify and analysis security and legal dependencies in the system environment. This is because; in some occasions the actor (and/or the system) cannot satisfy their goals and/or security constraints without the assistance from another actor (and/or system). For example, a data processing operator actor cannot process data without the data being in the system. For this to happen, the data processing operator depends on the data controller to acquire the data for the system. In turn, the data controller depends on the customer to provide that data. Thus the elicitation in fact takes place at two different levels. At the first level, the initial elicitation of the requirements is based on high level security and legal constraints. At the second level, the requirements that are elicited are furthered analysed within a threat environment by identifying how requirements are endangered by security attacks. This takes place in the component of our framework which is described in the following section.

3.3 Security requirements analysis component

The main goal of this component is to confirm whether elicited security requirements address the possible security threats. We employ *security attack scenarios*, an enhanced Secure Tropos model, which aims to analyse how the system copes with different kinds of security attacks [18]. We have opted for a scenario-based approach because scenarios can be integrated with our framework and can be adapted to the framework's notation and concepts. Attacker goals and tasks are identified and analysed to understand why and how an attacker might attack the system. This leads to a better understanding of how possible attacks can be prevented. In a *Security Attack Scenario* an attacker is depicted as an actor who aims to break the security of the system. The attacker intentions are modelled as goals and tasks and their analysis follows the same reasoning techniques that the Secure Tropos methodology employs for goal and task analysis. Attacks are depicted as dash-lined links (called attack links) that contain an "attacks" tag, starting from one of the attackers goals and ending to an attacked resource. The scenario differentiates between internal and external actors. A security attack scenario is graphically represented as shown in Fig. 3, taken from [18]. A circle (with parallel line on the top) represents an actor, a square represents a resource, a rounded square represents a goal and a polygon represents a plan. More information can be found in [18].

Note that, no-right and disability of the extract legal rights may influence how an attacker might employ illegal activities to attack the system. When an actor without any right performs any action, then non-compliance issues might arise within the system environment. Therefore, our framework considers non-compliance issues within the security attack scenario approach. We further use the concept of *secure capability* [14] that enables us to refine the initial high level security requirement related to that attack and make it more specific. This in turn allows us to address the attack including non-compliance issues. To support a detailed design of the system to fulfil the identified requirements, we employ Model-Based Security Engineering (MBSE) and in particular UMLsec, as explained in the next section.

3.4 Secure system design

The final component of the framework enables the development of a secure design aligned with the legal and security requirements identified and analysed by the previous

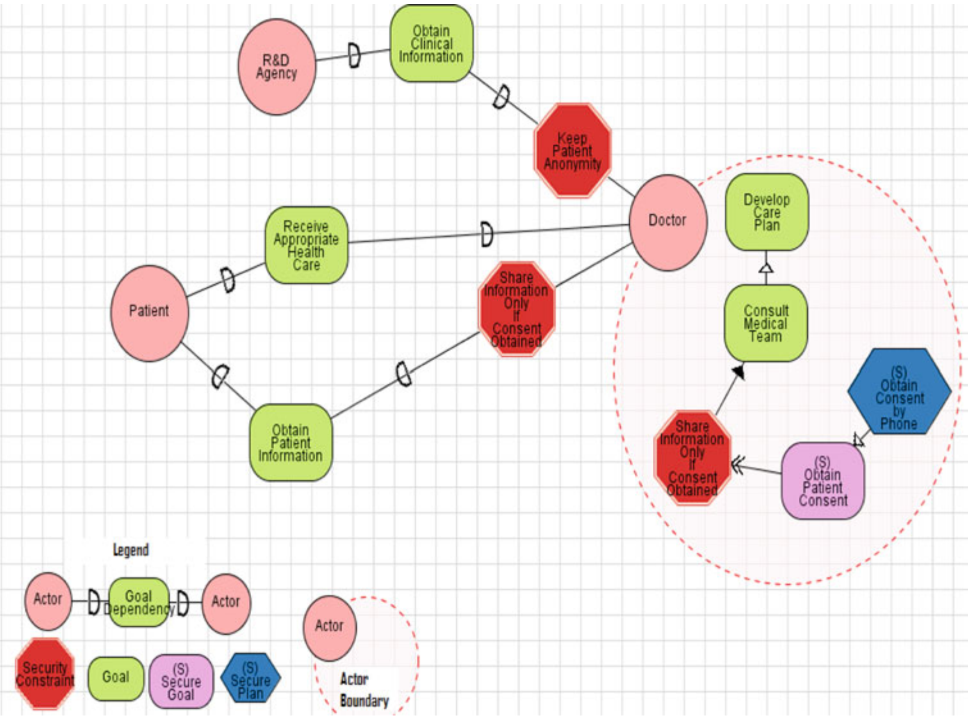
R&D Agency
Obtain Clinical Information
Keep Patient Anonymity
Doctor
Develop Care Plan
Receive Appropriate Health Care
Patient
Share Information Only If Consent Obtained
Consult Medical Team
(S) Obtain Consent by Phone
Obtain Patient Information
Share Information Only If Consent Obtained
(S) Obtain Patient Consent
Legend
Actor
Goal Dependency
Actor
Actor
Security Constraint
Goal
(S) Secure Goal
(S) Secure Plan
Actor Boundary

Fig. 2 Example of a Secure Tropos model

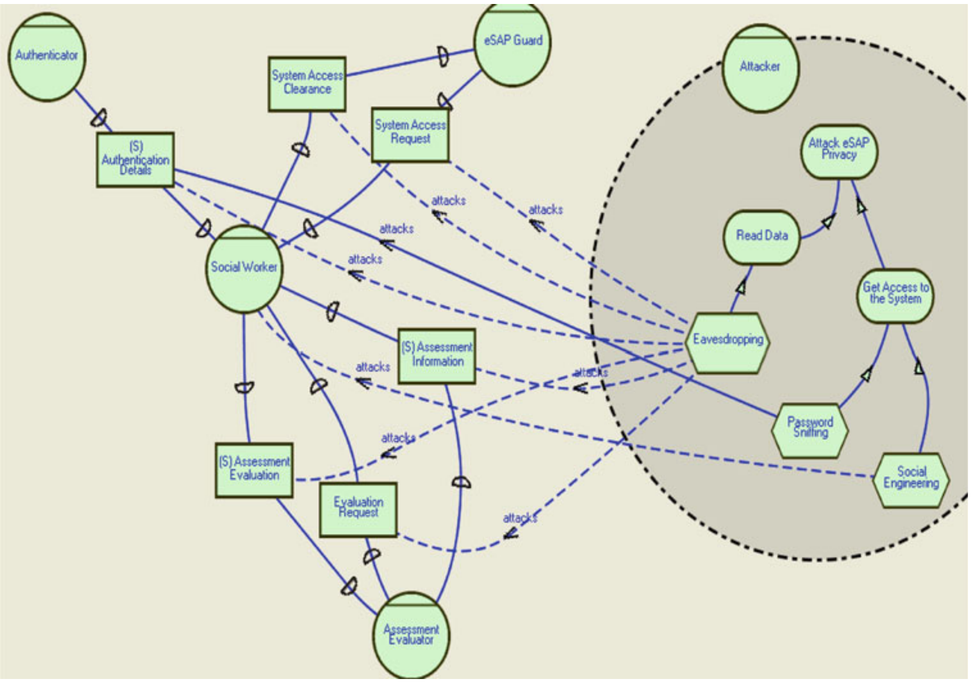

Fig. 3 Example of the graphical representation of a security attack scenario

components. In UMLsec, recurring security requirements, such as secrecy, integrity, and authenticity are offered as specification elements by the UMLsec extension [23]. These properties and the associated semantics are used to evaluate UML diagrams of various kinds that contribute to the architectural and detailed design of the system and indicate the possible security vulnerabilities. One can thus verify that the desired security requirements, if fulfilled, enforce a given security policy. One can also ensure that the requirements are actually met by the given UML specification of the system. The extension proposed by UMLsec, is given in form of a UML profile using the standard UML extension mechanisms.

Table 1 UMLsec stereotypes and associated tags

Stereotype	Base Class	Associate stereotypes	Tags	Tag value	Constraint	Description
Legitimate process	Subsystem	«provable», «integrity» «authenticity», «secure links»	Action, right, adversary	{collect, process, use}{privilege, claim, power, duty, liability}{no-right, disability}	Only permitted legal activities	Enforce legitimate processing in application context
Data filter	Node, link	«guarded>	Action, resource, adversary	{monitor, filter}, {data, service}, {deplete, flood}	monitor data & filter (if require)	Enforce monitoring & filtering data in communication & receipt

Stereotypes are used together with tags to formulate security requirements and assumptions on the system environment. For instance, existing UMLsec stereotypes support security related information including security assumptions (such as the «Internet» stereotype considering security properties of the physical layer of the system), security requirements (such as the «secrecy» stereotype considering security restrictions on the logical structure of the system), and security policies (such as the «secure links» stereotype supposed to be obeyed on different parts of the system) [23]. The tags defined in UMLsec represent a set of desired security properties under the stereotypes. For instance, adversary tag under the «secure links» stereotype specifies possible types of adversary that may threat the links based on some logical condition from previous knowledge of the adversary. Constraints give criteria that determine whether the requirements are met by the system design, by referring to precise semantics (discussed below).

3.4.1 UMLsec stereotypes

The definition of UMLsec stereotypes allows the integration of security requirements into design diagrams and it supports model checking and automated tool implementation. A detailed explanation of the stereotypes defined in UMLsec is outside the scope of this paper and it can be found in [23]. The framework presented in this paper extends the existing set of stereotypes and it enhances the verification with stereotypes related to legal concepts. This enables us to verify both static features and simple dynamic features combined with security and legal stereotypes within a system environment. In particular, two new stereotypes («legitimate process» and «data filter») are defined to support legal concepts. These stereotypes, developed as UML model elements, add security and legal concepts within UMLsec design diagrams. Table 1 provides a summary of these newly defined stereotypes and their tag values and constraints.

Explanation of «legitimate process» stereotype

This stereotype mainly considers the legal rights and it is used to associate correlatives to regulate the actions performed by the actor in a subsystem. Actors, their possible actions, objects, and security and legal constraints identified from the previous components are modelled by the UMLsec diagrams along with the stereotypes. For instance, within a subsystem, one can define the behaviour of the system components by defining behaviour of each object. If the object is an actor, then actions performed by the actor are regulated by the extracted legal rights. We can model these behaviours within UMLsec diagrams such as class and deployment diagram. The stereotype has an associate {adversary} tag, which may have values of the form (T, C), where T denotes adversary type and C denotes logical condition, to support the non-compliance issues due to no-right and disability, in addition to the existing threats.

Explanation of «data filter» stereotype

This stereotype mainly ensures availability of objects, particularly data and service, required to perform an action. The main focus is to monitor and, if required, filter any malicious traffic within a communication link and any unnecessary request received by the targeted node. Two new abstract threat elements “deplete” and “flood” along with the existing threats of the UMLsec attacker model are employed to model a specific type of adversary in a given application context. *Deplete* represents the possibility that an adversary may exhaust the resources of the targeted node, whereas *flood* represents the possibility that an adversary may overflow the targeted communication link. Thus, we extend the security analysis of UML diagrams defined in [23] such that Threats $A(s)$ is a function that is given an adversary type A and a stereotype type s and that returns a subset of {*delete, read, insert, access, deplete, flood*}. Threats $A(s)$ specifies the threat scenario associated with an adversary type A against a

Table 2 Threat from the *default* attacker

Stereotype	$\mathrm{Threats}_{\mathrm{default}}$
Internet	{delete, read, insert, flood}
Server node	{access, deplete}
LAN	{delete, read, insert, flood}

component or link stereotype s. This allows us to derive basic concrete threats from the abstract threats and model and analyse the possible adversary behaviour and issues related to non-compliance. Table 2 shows the threats that arise from the default attacker against various kinds of communication links and system nodes. The main difference between the two newly defined stereotypes is that «legitimate process» mainly focuses on the actor's behaviour of actions regulated by the laws and regulations and « data filter» mainly focuses on the prevention of attacks in particular attacks relating to the Denial of Service (DoS) within the system environment. However, these newly defined stereotypes support the fulfilment of both security goals and legal goals.

3.4.2 Architectural and detailed design

As mentioned, in UMLsec it is possible to evaluate various kinds of UML diagrams considering the security properties and associate semantics. In our framework, we employ architectural and detailed design of the system to support the secure system design. Architectural design mainly deals with the definition of the system's global architecture in terms of subsystems along with security and legal requirements, actors, and their responsibilities to fulfil the corresponding security and legal goals. Detailed design aims to specify each architectural component in further detail, in terms of inputs, outputs, control, and other relevant information. Therefore diagrams under the architectural and detailed design support modelling of the system within the model-based development. The goal is to increase the quality of the software system while keeping the implementation cost and the time-to-market bounded. In MBSE [23–25], recurring security requirements (such as secrecy, integrity, authenticity and others) and security assumptions on the system environment, can be specified either within a UML specification or within the source code (Java or C) as annotations. The associated tools [24] generate logical formulas formalizing the execution semantics and the annotated security requirements. These tools support verification of the most important security requirements, which can be directly used in the model, together with their formal definitions. Automated theorem provers and model checkers automatically establish whether the security requirements hold. If not, a Prolog-based tool automatically generates an attack sequence violating the security requirement, which can be examined to determine and remove the weakness. This way we encapsulate knowledge on prudent security engineering as annotations in models or code and make it available to developers who may not be security experts. Since the analysis that is performed is too sophisticated to be done manually, it is also valuable to security experts.

To provide a concrete illustration of the above ideas and how they are utilised in our framework, we consider deployment diagrams, as a part of architectural design, which are used to describe the physical layer of a system. We use them to check whether the security requirements on the logical level of the system are enforced by the level of physical security, or whether additional security mechanisms (such as encryption) have to be employed. Assume for example that we have a business application, part of an e-commerce system, which is supposed to be realized as a web application. The payment transaction involves transmission of data to be kept secret (such as credit card numbers) over Internet links. This information on the physical layer and the security requirement is reflected in the UML model shown in Fig. 4. We then use the stereotype «secure links» to express the demand that security requirements on the communication are met by the physical layer. More precisely, for each dependency stereotyped *secrecy* between subsystems or classes on different nodes n, m, and any communication link between n and m

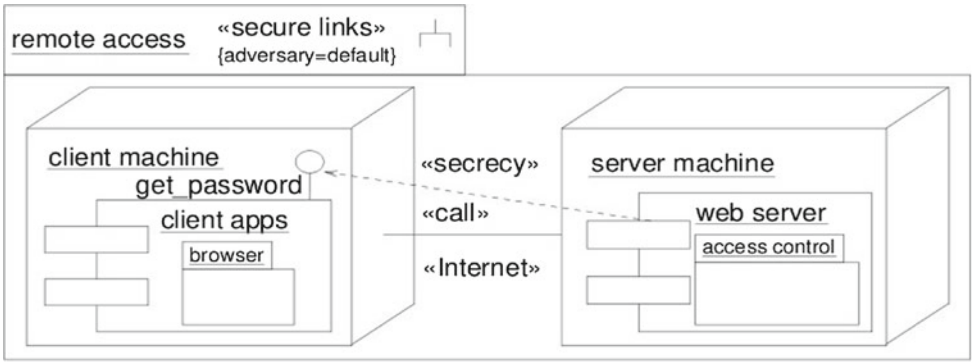

Fig. 4 UMLsec deployment diagram

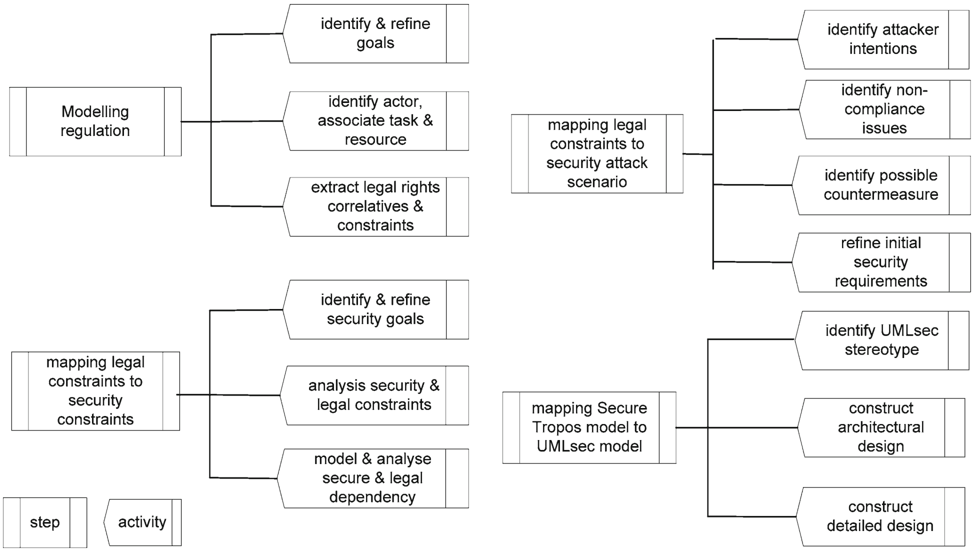
Modelling regulation
identify & refine goals
identify actor, associate task & resource
extract legal rights correlatives & constraints
mapping legal constraints to security constraints
identify & refine security goals
analysis security & legal constraints
model & analyse secure & legal dependency
step
activity
mapping legal constraints to security attack scenario
identify attacker intentions
identify non-compliance issues
identify possible countermeasure
refine initial security requirements
mapping Secure Tropos model to UMLsec model
identify UMLsec stereotype
construct architectural design
construct detailed design

Fig. 5 Steps and activities of the proposed framework

with some stereotype *s*, the threat scenario arising from the stereotype *s* with regard to an adversary of a given strength should not violate the secrecy requirement on the communicated data. We note that in the given diagram, this constraint associated with the stereotype secure links is already violated when considering standard adversaries, because plain Internet connections can be eavesdropped easily, and thus the data that is communicated does not remain secret. For this adversary type, the stereotype «secure links» is thus applied wrongly to the subsystem, which is pointed out automatically by the UMLsec tools.

4 Development process

The previous section described the components of our framework. In this section, we describe the process followed in order to arrive at a secure software system design that complies with appropriate legal regulations. Figure 5 depicts the steps and underlying activities involved in the development process. The process is initiated by identifying goals and legal rights from the relevant regulation and by modelling the system context. Therefore, before analysing the regulation, relevant laws need to be selected based on the initial system under development artefacts such as business specification, application scope/context, stakeholder expectations, and requirements. In particular, regulations are initially mapped into different Secure Tropos elements and appropriate legal constraints are identified with the aid of Secure Tropos models. Further analysis, including non-compliance issues, takes place using the security attack scenario models. Finally a secure design is developed and validated with the aid of UMLsec models. The process is divided into four main steps, each one discussed in the following sections.

4.1 Modelling regulation by Secure Tropos

The first step of the process aims to analyse the laws and regulations that were identified and to map legal requirements to Secure Tropos concepts, in order to support analysis of these requirements. In doing so, the first component of our framework is employed to support reasoning of compliance, when initial security requirements are elicited, and to assist the analysis of the requirements, based on non-compliance assumptions posed by security threats. This type of analysis goes beyond the usual ideas of only considering obligation and permission. An overview of the activities involved at this step is given below:

- *Identify and refine goals:* This activity describes both regulation related goals, by stating why the regulation is introduced based on the relevant legal source document, and actor related goals, by stating why an actor performs an action within the application context. Purpose pattern, described in the previous section, along with the business specification and application scope assist to identify the goals. Goals are initially identified from high level objectives and they are subsequently refined to more precise goals and sub-goals. Furthermore, links between regulations and actor goals are established to indicate how an actor's goals support the satisfaction of the regulation goals (in case they do). In our framework, and consistent with the Secure Tropos definition, an actor's goals represent the strategic interests of the actor within the system environment. It is worth mentioning that the purpose of the specific information security law is usually treated as the main goal, and goals from different sections, articles or related annex are usually refined into several sub-goals. However, consideration should be given only to the

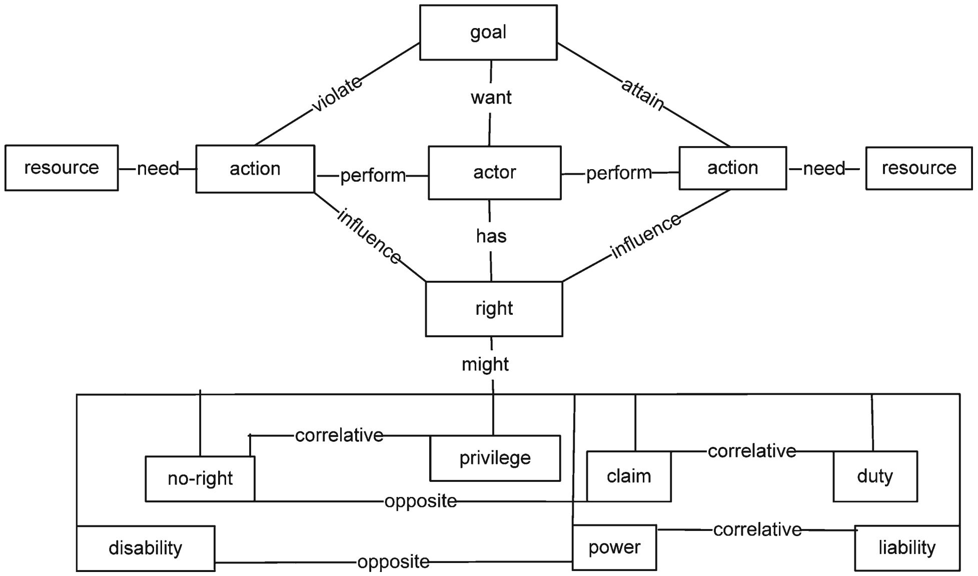

Fig. 6 Conceptual view of the modelling regulation

sections, articles or annexes that are relevant within the application context, so that scope of the goals and their elaboration confine with the application.

- *Identify actors, associate tasks and resources:* This activity identifies the actors, their performed tasks, and required resources in the system environment. Generally actors perform specific tasks and they require certain resource to accomplish the task in order to fulfil the goal. Note that we use tasks to represent the actions performed by actors regulated by the legal rights. Therefore, as mentioned, legal rights are concerned with the categorisation of actions into behavioural and productive actions. Note that usually, it specifies more the behavioural actions of the actor. These rights might be actor freedom or obligation to perform the task or certain restriction not to perform the task. We employ the activity pattern and the common rule set, described in previous section, to identify actors and their associate legal rights to perform specific tasks. Complementary, the object pattern is employed to identify possible resources, its type, resource owner and location based on the application context. The main concern when dealing with resources is whether the resource is available, and the actor has adequate rights to handle the resource. Whenever actors, their associated roles, and tasks are identified, then legal rights should support the roles. To support such modelling, our framework extends the Secure Tropos modelling language. The original Secure Tropos language does not provide support for modelling legal dependencies. We extend the language and define a legal dependency as a dependency that introduces a legal goal and in which an actor has legal rights to perform an action to attain the goal. This dependency also supports the correlatives among the rights.
- *Extract legal rights, correlatives and constraints:* Finally the legal right, correlatives, and legal constraints are extracted once the goals, actors, tasks, and resources are identified. Every right of an actor may be correlative as well as opposite of another right. Therefore care should be taken when the rights and correlatives are extracted for the various actions. The rights (as mentioned Sect. 3.1) sometimes imply freedom of an actor to perform an action, such privilege, or imply restricted freedom to perform an action, such as duty and liability, or even imply certain restrictions that forbid an actor to perform an action, such as no-right and disability. Therefore every extracted legal right does not imply restriction and we need to distinguish the rights which are stringent to apply. In our framework, when legal rights and correlatives restrict, limit or control certain actions in a situation, then these rights are defined as legal constraints. In particular, legal constraints are used to model restrictions, imposed by relevant regulations, of an actor's rights to perform an action or to prevent the actor from performing an action. Figure 6 depicts a conceptual view of the rights and associate correlatives and opposites that belong to the right holder and its contribution to attain or violate the goal based on the required resource and action performed by the actor.

To enhance understanding, we provide a small example to demonstrate the modelling regulation step of the framework. Note that a detailed demonstration of this step is presented in Sect. 5, with the aid of a case-study. In particular, we employ the underlying activities within the modelling regulation to extract the legal and correlatives from the legal text. We consider article 16 of directive 95/46/EC to demonstrate the example.

Example

Article 16 (confidentiality of data processing), directive 95/46/EC
Any person acting under the authority of the controller or of the processor, including the processor himself, who has access to personal data, must not process them except on instructions from the controller, unless he is required to do so by law.

Input scenario
John is working as a data processing operator in the IT department of a bank. Several private companies are the main customers of the bank. Therefore the bank maintains the salary information of the employees working in the companies. John's main task is to update the account balances once the salary sheet has been arrived from the companies at the end of every month.

Modelling regulation
Initially we need to identify the goals of the actor and those resulting from the relevant regulation. Using the "identify goal" activity of the proposed framework, the main goal of the directive is to protect personal data. Then, using the refinement goal activity the main goal is further analysed and refined into a number of sub-goals such as confidentiality when data is processed (Article 16 above). Moreover, goals and sub-goals are identified for the relevant actors. For example, John as an actor has a goal to perform his job responsibly and a sub-goal to accurately process the relevant data. By employing the "identify actors" activity of the framework, two actors are identified from the legal text, i.e. the data processing operator and the data controller. To keep this example simple, for the sake of demonstration, we consider John as the data processing operator who operates on behalf of the controller and a customer actor as the owner of the individual account information. The main task is the process of the data, i.e. the update of the account balance by John once a salary sheet is received. The main resource is the personal (customer) data. By following the "extract legal rights, correlatives and constraints" activity of the framework and by considering only normative phrases (i.e. must not) and conditional keywords (i.e. unless)—again for the sake of simplicity for this example—a number of legal rights are extracted as shown below. It is worth noting that all extracted rights imply restrictions and therefore in our framework we treat them as legal constraints.

- John has the liability to process only the customer account balance once the salary sheet has been arrived.
- When an authorised customer requests the account balance, then John has the duty to provide the account balance information.
- John has no-right to process other information of the customer beyond the account balance (e.g. due to the normative phrase *must not* in the legal text)
- John has no-right to use customer account information for any other purpose. For example John must not disclose salary information to another company's employees.

4.2 Mapping legal constraints to security constraint and security attack scenario

In the previous step, all legal rights, correlatives, and legal constraints were identified based on the legal text and the actors of the system under development. The next step includes two sub-steps. The first sub-step is used to identify relevant security requirements of the system under development, model them as security constraints, and align them with relevant legal constraints. The second sub-step is used to map legal constraints to security attack scenarios. This enables us to analyse, based on possible attacks, the satisfaction of the legal and security constraints, and if necessary refine the initial security requirements. Similar to the previous step, the Secure Tropos goal modelling language is employed to elicit security constraints and analyse them along with legal constraints of the system under development and the Security Attack Scenarios approach is used to analyse the satisfaction of legal constraints based on potential security attacks to the system.

4.2.1 Mapping legal constraints to security constraints

In principle, the conceptual approaches of this sub-step and the previous step are the same. The main difference lies in the fact that in the previous step, the various actors of the system were analysed, while in this step, the system itself is analysed. In particular, the various activities of this sub-step are:

- *Identify and refine security goals.* Security goals are identified by analysing the business specification, the system environment, regulation related goals, and stakeholders' expectations. Similar to the analysis described in the previous step, initial goals might be high level, which if required are refined to sub-goals that directly support the main goal. From the security point of view, the main focus is to ensure critical security properties such as confidentiality, integrity, availability, authenticity, and non-repudiation of the system under development.
- *Analyse security and legal constraints.* Similar to the previous step, relevant security constraints and legal constraints are identified and analysed. Again the main difference being that in this step the focus is on the constraints related to the system rather than the actors. It is worth noting however, that a number of security constraints and

in fact legal constraints can be the same for the system as for the actors. This is particularly true when specific goals of the actors are satisfied through the system. In this case, the security and legal constraints related to these goals are also introduced to the system.

- *Model and analyse secure and legal dependencies.* During this step, we identify and analyse the potential dependencies related to the security constraints and goals as well as the legal constraints. Since this type of dependencies might affect the satisfaction of the various security constraints and legal constraints, it is important to model and analyse these dependencies. It is also important for our work to differentiate between two different types of dependencies (i.e. security and legal) and model and analyse these dependencies related to the security and legal constraints respectively. Our framework extends the Secure Tropos modelling language, which already considers the concept of secure dependency, with the notion of legal dependency. Legal dependencies, in our framework, are modelled along with secure dependencies and they enable developers to model how an actor, with some legal rights and constraints, executes some behaviour action and depends upon another actor's correlative action to attain some goal as productive action by using some resource (if it is required). Moreover, legal constraints are correlated with security constraints so that identified security requirements can be aligned with the relevant regulations. To achieve this, our framework supports the integration of legal constraints to secure dependencies (and security constraints to legal dependencies) to support the mapping of security constraints with the relevant legal constraints within the context of the secure/legal dependency. From an analysis perspective, this means that the modelled security/legal constraints need to be satisfied by the relevant actors for the dependencies to be valid from a legal point of view.

4.2.2 Mapping legal constraints to security attack scenarios

Research [1] has shown that the earlier in the development process security requirements are verified the less the cost is for the project. We believe the same is true for the analysis and verification of legal requirements. Therefore, during this step we analyse the identified security and legal requirements (from the previous steps) based on possible security attacks and threats to the system. This is important, since some of these attacks, apart from introducing security issues, might introduce non-compliance issues to the system under development. It is therefore necessary to identify these issues and refine the security requirements so both security and non-compliance issues can be addressed early in the development process. To support this, our framework integrates the process of security attack scenarios [18] of Secure Tropos. We extend the current scenario representation by mapping certain legal constraints (e.g. no-rights and disability) within security attack scenarios to specify the non-compliance issues. There are four activities in this step:

- *Identify attacker intentions.* This activity enables developers to understand the various motives of the attacker, as attacker's goals, that lead him/her to perform the attacks. By understanding these motives, we might be able to identify some non-technical protections for the system. We are also able to identify potential resources of the system that might be attacked. In our framework, we model the goals of an attacker along with possible resources of the system that might be attacked with the aid of attack links. An attack link connects an attacker's goal and the corresponding system resource that might be attacked due to the specific attacker goal. Graphically, these links are represented as dash-lines with an «attack» tag.
- *Identify non-compliance issues.* The aim of this task is to identify non-compliance issues when an attacker performs an action beyond his right and disability or does not perform an action despite of duty and obligation. We model this situation within security attack scenarios to specify the non-compliance issue within the system environment. To support this, we extend the security attack scenario notation and meta-model by introducing a new nonconformity link. This link is used to distinguish the case of a non-compliance issue from a case of security attack in a given scenario. Graphically, this link is represented as a dash-link with a «nonconformity» tag.
- *Identify possible countermeasures.* Here we consider how the system under development "reacts" to the potential attacks and non-compliance issues identified in the previous steps. For this reason, the concept of secure capabilities is used, from the Secure Tropos methodology, to identify capabilities of the system that countermeasure the potential attacks so that the initially identified legal and security goals can be attained. Secure capabilities can prevent these attacks in the sense that an actor with such capabilities can react to these attacks. The secure capabilities, of each actor, that help to prevent the identified attacks are modelled as dashed-links, incorporating a «help» tag, which indicates the capability and the attack they help to prevent. Attacks that cannot be prevented are notated as solid attack links (as opposed to dashed attack links).
- *Refine initial security requirements.* Based on the previous steps of the analysis, the initial security requirements are refined (if needed) to accommodate provisions for the countermeasure of attacks that cannot be prevented

with the existing set of requirements. Simultaneously, the newly identified requirements are aligned with the appropriate legal requirements using the activities and steps described previously. Once the developers are confident that the secure capabilities of the system can countermeasure possible attacks, the design of the system is developed.

4.3 Mapping secure tropos models to UMLsec models aligning with a given regulation

The final step of the framework focuses on the construction of the architectural and detailed designs of the system. During this step, our framework maps the Secure Tropos model to relevant UMLsec models and it assists the developer in aligning the security and legal artefacts to the UMLsec design diagrams. The underlying activities involved within this step focus on how to construct the design diagrams so that design can support both the security and legal concepts. Note that, Mouratidis et al. already integrated the Secure Tropos notation and the UMLsec notation for the development of the secure system [15] and we take this work as our foundation. However, that work did not address the alignment of legal concepts or security attack scenarios to the design solution. Therefore, our framework extends beyond their contribution by considering the legal dimension, including non-compliance issues, as well as security attack scenarios. An overview of the activities involved in this step is given below:

- *Identify UMLsec stereotypes.* As stated, UMLsec stereotypes, when attached to the UMLsec model add security information to the model elements. However, the existing UMLsec notation does not directly support legal information. Therefore, our framework includes the new stereotypes «legitimate process» and «data filter», as stated earlier section, to support modelling and analysis of legal concepts within the UMLsec models. The focus of this activity is to identify the appropriate stereotypes so that both the legal and security information is appropriately modelled in the corresponding UMLsec model. To assist the identification of the appropriate stereotypes, developers need to look at the models defined in the previous stages, in particular the identified security requirements and legal constraints. For each one of these, stereotypes need to be employed to model the issue raised by the requirements and the constraints.
- *Construct architectural design diagrams.* This activity mainly constructs the architecture of the overall system by UMLsec class and deployment diagram. As mentioned above, existing work has mapped Secure Tropos models to UMLsec models and in particular UMLsec class and deployment diagrams within architectural design [15]. Therefore we follow the guidelines to support the mapping of Secure Tropos analysis model to UMLsec class and deployment diagram based on the current work. In addition, we include legal issues and refined security requirements from the security attack scenarios within the diagrams, so that the architectural design enables developers to trace not only the security requirements but also align the design with the legal concepts identified in the previous analysis.
- *Construct detailed design diagrams.* In this activity, details of the system components are specified with the aid of UMLsec sequence diagrams. UMLsec sequence diagrams model secure interactions of the system's components. Constraints associated with the UMLsec stereotypes considering security and legal concepts are integrated within the diagram. Therefore our framework maps Secure Tropos actor model with UMLsec sequence diagrams to precisely specify different actor interactions considering derived legal rights from legal text. The following steps were identified for the mapping:

  Step 1: *Identify object for specific interaction.* Actors in the Secure Tropos security enhanced actor model who participate in a specific interaction (i.e. application scenario) are treated as objects for the UMLsec sequence diagram.

  Step 2: *Identify possible interactions among the objects.* Each dependency (both legal and secure) among actors might provide interaction within the objects in the corresponding sequence diagram. However this is not a strict rule, depending on the type of dependency and the specific sequence of interaction.

  Step 3: *Identify possible messages and arguments within interactions.* Capabilities of each of the actors' operations are mapped, within UMLsec sequence diagrams, into the message exchange from one object to another object. Resources related to each of the actors are mapped to arguments of the message on the UMLsec sequence diagram.

  Step 4: *Identify the necessary UMLsec stereotypes.* UMLsec stereotypes are identified through the secure and legal dependencies from the Secure Tropos security enhanced actor model. The type of the secure and legal dependency indicates whether an object is critical for the security of the system or not. Any specific resource (related to security and legal constraints) required by the stereotype is also considered within the interaction. If required, these constraints are directly mapped as arguments or messages as security and legal rules on the sequence diagram.

5 Case study

To demonstrate the validity of our approach, we present an application from an industrial context based on the secure service-based software (SBS) system. The goal is to elicit security requirements, to align them with relevant regulations, and to design a secure system, in which the design diagrams support both the security requirements and the legal constraints. We consider two different directives from the EU information society legislation (2002/58/EC and 95/46/EC) [21] and the corresponding national law from the German Federal Data Protection Act (FDPA) [5]. We have chosen these because they are relevant to the application context of the system under development (the case study is in Germany). In particular, the purpose of directive 2002/58/EC is on privacy and electronic communication and directive 95/46/EC is on the protection of personal data. Partial text from Article 4 and 5 of 2002/58/EC are considered relevant with the application context, and partial text from Article 17 of 95/46/EC is considered as a cross reference of the 2002/58/EC. It is worth mentioning, for the sake of clarity, that EU directives are not directly executed within the EU member states but they define a set of regulations that must be achieved by an EU member state's national laws. Member states must adapt their national laws to meet the goals from the related specific directive, and they are free to decide how to do that. Therefore we have included the national law German Federal Data Protection Act (FDPA), because it relates to directive 95/46/EC. The reasons for choosing FDPA are that the case-study context lies in Germany, and the act ensures privacy of personal data and legitimate collection, processing or use of personal data, and therefore it supports the implementation of the directive 95/46/EC. Moreover, the selected Section 5, 9 and Annex from the FDPA address the articles of the 95/46/EC.

We consider several hypotheses for the case study to validate the framework. They are:

- Regulation modelling extracts the possible legal rights and constraints from the relevant legal text (H1).
- The identified security requirements are influenced by corresponding legal constraints (H2).
- The refined security requirements address non-compliance issues in addition to security threats (H3).
- The secure design diagrams trace the requirements and the legal constraints (H4).
- The framework should support both forward and backward tracing (H5).

To make the case-study consistent and easy to understand, we consider the following business goals and related features from the application context.

Business goals

- The system shall allow the legitimate user to request any resource.
- The system shall provide response to any legitimate user request.

Major features

- The system should support single-sign-on service with a centralized authentication mechanism for any service request.
- User requests for protected resources and manipulation of relevant data should be available through the Internet.

Scenarios
A user should be able to request, process, and restore specific data from anywhere by establishing a connection with the service provider. A single customer or group of customers under a specific subscription are treated as users. In our scenarios, we distinguish the participating entities. *A user* represents a customer who has signed up for access to protected content; *Browser* is the web application entity responsible for handling user requests and responses; *User Identity Provider* is the authority responsible to validate the user identity and support login procedures; *Content Manager* is the web service entity responsible for delivery content (mostly data) and for user data restoration; *Certificate Authority* is the authority responsible for issuing appropriate security certificates for legitimate users; and *Service Provider* is the authority responsible for the overall infrastructure.

5.1 Modelling regulation

Following the steps described in the previous section, we analyse the legal text from the directives and the act.

Relevant legal text

Directive 2002/58/EC (Directive on privacy and electronic communications)

Article 4 (partial), Security
1. The provider of a publicly available electronic communications service **must** *take* ***appropriate technical and organisational measures*** to safeguard security of its **services**, **if** necessary in conjunction with the provider of the public communications network with respect to network security. Having regard to the state of the art and the cost of their implementation, these ***measures*** **shall** ensure a **level of security appropriate to the risk** presented.

Article 5 (partial), Confidentiality of the communications

Member States **shall** ensure that the *use* of electronic communications networks to *store* **information or** to gain *access* to **information** *stored* in the **terminal equipment** of a subscriber or user is **only** allowed on condition that the subscriber **or** user concerned is provided with clear **and** comprehensive **information** in accordance with Directive 95/46/EC, inter alia about the purposes of the *processing*, **and** is offered the right to *refuse* such *processing* by the data controller. This **shall not** *prevent* any technical storage **or** *access* for the sole purpose of carrying out **or** facilitating the transmission of a communication over an electronic communications network, **or** as strictly necessary in order to provide an information society service explicitly requested by the subscriber **or** user.

Directive 95/46/EC, (Directive on protection personal data)

Article 17 (partial), Security of processing

1. Member States shall provide that the controller **must** implement ***appropriate technical and organizational measures*** to *protect* **personal data against** accidental or unlawful destruction **or** accidental loss, alteration, unauthorized disclosure **or** access, in particular where the processing involves the transmission of ***data*** over a network, **and** against all other unlawful forms of processing. Having regard to the state of the art and the ***cost*** of their implementation, such ***measures*** **shall** ensure a ***level of security appropriate to the risks*** represented by the processing and the nature of the **data** to be protected.

German Federal Data Protection Act

§ 5 (Confidentiality)

Persons employed in data processing **shall not** *collect, process* **or** *use* **personal data** without authorisation (confidentiality). On taking up their duties such persons, in so far as they work for private bodies, **shall be** required to give an undertaking to maintain such confidentiality. This undertaking **shall** continue to be valid after termination of their activity.

§ 9 (Technical and organisational measures)

Public and private bodies *processing* **personal data either** on their own behalf **or** on behalf of others **shall** *take* the ***technical and organisational measures*** necessary to ensure the implementation of the provisions of this Act, in particular the requirements set out in the annex to this Act. Measures shall be required only if the effort involved is reasonable in relation to the ***desired level*** of protection.

Annex (partial) (to the first sentence of Section 9 of this Act)

Where **personal data** are *processed* **or** *used* automatically, the internal organisation of authorities **or** enterprises are to be arranged in such a way that it meets the specific requirements of data protection. In particular, ***measures*** suited to the type of **personal data or** data categories to be *protected* **shall be** taken,

1. to prevent unauthorised persons from gaining *access* to data processing systems with which ***personal data*** are *processed* **or** *used* (access control),
3. to ensure that persons entitled to *use* a data processing system have *access* only to the **data** to which they have a right of *access*, and that **personal data cannot** be *read, copied, modified* **or** *removed* without authorisation in the course of *processing* **or** *use* **and** after storage (access control),
4. to ensure that **personal data cannot** be *read, copied, modified* **or** *removed* without authorisation during electronic transmission **or** transport, **and** that it is possible to check **and** establish to which bodies the transfer of ***personal data*** by means of data transmission facilities is envisaged (transmission control),
7. to ensure that **personal data** are *protected* from accidental destruction **or** loss (availability control).

In the above text, normative phrases (such as must, shall) and conditional phrases (such as and, or) are in bold; a subject for an action is underlined; an action is italicized; an object is in bold and underlined; a measurement parameter is in bold, italicized and underlined. Following the steps of the framework, the main focus is now to identify basic Secure Tropos elements based on the legal taxonomy.

Analysing the regulation

Identify and refine goals. The main goal of 2002/58/EC is to *protect privacy in electronic communication* and *accurate processing of the personal data*. Article 4 of the directive refines this high level goal to *secure service* by taking appropriate technical and organisational measures and article 5 refers to *confidentiality in electronic communication*. The main goal of 95/46/EC is to *ensure protection of personal data*. Article 17 refines the main goal to *security in processing* and also to *adequate technical and organisational measure*. The FDPA has similar goals as 95/46/EC because it directly implements that directive. Section 5 of FDPA refines the main goal to *confidentiality in data processing* by processor or processing operator and section 9 refines the main goal to *proper access, transmission and availability control* under *technical and organisational measure*. Note that, our framework facilitates the establishment of a link between the EU directive and the relevant national law by using the concepts of goal and sub-goal.

Identify actors, associate tasks and resources. Our analysis has resulted in the following for the main actors of the system. A user's main goal is to use and process data through the tasks request, process, own, and restore. The actors under the providers are *identity manager*, *certificate authority*, *content manager*, and the *service provider* itself. The identity manager's main goal is to identify the user before the user is allowed to perform any action. The certificate authority delivers the identity information once a user subscription has been approved. The data controller or data processing operator under the data controller serves as the content manager and is responsible for the delivery of the requested data, restoration of data and if require further process of the data. Finally, the service provider is responsible for the overall system infrastructure including ensuring adequate technical and organisational measures, secure services, and protection of personal data. It is also possible for some of these actors to play several roles. For example, the data controller may play the role of service provider, certificate authority, or identity manager. The main tasks from the provider perspective are to manage user identity information and sensitive data, and to respond any legitimate user request. Therefore, these tasks should support the satisfaction of the goals (and refined sub-goals) identified in the previous step, such as *protect personal data, secure service, provide adequate technical*, and *organisational measures*.

Extract legal rights, correlatives, and legal constraints. The final activity in this step of the framework is to extract appropriate legal rights, correlatives, and legal constraints for the application context.

Claim/duty, power/ liability, and privilege

- When a legitimate user claims to access, process, and use any specific data or claims to restore data, then the controller has the duty to perform the appropriate action based on the nature of the claim.
- A legitimate user has the privilege to access, process, and use data through public communication networks.
- The content manager has the liability to access, process, and use the data if required up to his/her specific limit.
- The service provider has the liability to take appropriate technical and organisational measures to support security of service, protection of personal data, access, transmission, and availability control.
- The content manager has the liability to take appropriate technical and organizational measures to support lawful processing, to protect against accidental loss or destruction of the data.
- The service provider, the content manager, and the resource owner have the power to refuse any user access, process, and storage of specific data, information, and/or service.

No-right and disability

- A user has no-right to unauthorised access, disclose, and unlawful process of data.
- The service provider, the content manager, and/or the owner have no right to unauthorised access, process, and use of the data.
- Without consent from the service provider, the content manager, or the owner, the user should not (disability) access, process, and store data.
- The system has no-right to accidental loss and destruction of data.

Note that except of privilege and power, all the remaining legal rights should be treated in strict sense within the context. Therefore, we consider all of them as legal constraints. Moreover, during our analysis, we have identified some differences between the EU directives and the German FDPA. The normative phrases of the directives generally are represented using *must, shall, shall not,* and so on, but FDPA represents the same meaning without using *must*. Moreover, measuring parameters such as appropriate technical and organizational measure are usually refined to several constraints such as access, transmission, availability control by the Annex of §9. From this, we can conclude that the national law facilitates more the extraction of legal rights compared to the directives. It is also worth noting that not all ambiguous terms of the legal text can be refined by our framework. An example is the level of security appropriate to the risk from article 17 of 95/46/EC, and desired level of protection from § 9 of FDPA. This is mainly because these terms are mainly related to several measuring elements including cost and nature of the object, risk and its impact on the overall business, cost of control action, but also non-technical issues such as underlying security policies, procedure and effectiveness of their implementation, organisational culture, and so on. Therefore, it is difficult to combine all these technical and non-technical issues into a single instance stated in the legal text.

5.2 Mapping legal constraints to security constraints

The artefacts from the previous step are employed in this step to assist mapping legal constraints to security constraints. Initially, we further refine some of the goals. For instance, the goal *access control* is refined to *user identification* and *required level of authorization*. After refining all the required goals, the next activity aims to analyse security and legal constraints and to develop the associate security and legal

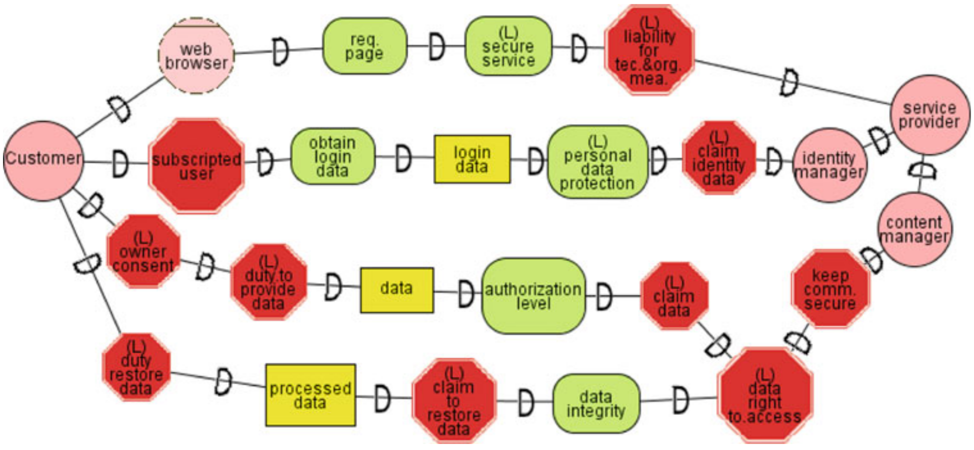

Fig. 7 Secure Tropos model with security and legal dependency

dependencies. In doing so, we need to identify the security constraints from the application context and align them with the identified legal constraints through the dependencies of the security enhanced actor model. Initially, the user submits a subscription request through a protected web browser to the service provider. The user expects that the service should be secured within the communication link. To support the single-sign-on web-service feature along with legal and security goals, every user after approval of the subscription, *claim identity data* to attain the user goal *obtain login info*. The Content manager provides the login data to subscripted user. A legal dependency is created from the user to the identity manager by integrating the legal goal *protection of personal data* along with the relevant rights and constraints. Similarly, a secure dependency is created by adding the secure goals *service availability*, and *data processing integrity* along with tasks, resources, and constraints.

Once the user claims data then the content manager has the duty to provide data when having the owner consent. Note that the content manager accesses only the data for which he/she has the right to access, process, and deliver. No option should be provided for the content manager to access data for which he/she does not have the appropriate rights. The content manager also expects that data processing should confirm integrity so that the purpose of security in data processing must not be violated. User might also claim to restore data and the content manager has the duty to restore it. Data communications through the public network requires to keep communication secure as security constraints so that unauthorised user cannot access, process, and use any data. The security enhanced actor model, shown in Fig. 7, provides the mapping between security and legal constraints in a manner that elicited security requirements, modelled as security constraints, can be aligned with regulation through several security and legal dependencies.

Based on the above analysis, we have derived a set of security requirements that align with legal constraints as shown below. Note that, for reasons of simplicity, we do not consider all the identified aligned security requirements here.

I. The user shall be identified and authenticated before allowed to perform any action relating to data collect, process, and use.
II. The system shall only provide the related data to the authorised user that has right to collect, process, and use.
III. The service provider shall support sufficient organisational and technical measure to ensure privacy of the personal data.
IV. The system shall keep the established communication link secure to protect any unauthorised access and process.

5.3 Mapping legal constraints to security attack scenario

To perform this step, we need to identify the attacker intentions, their tasks, and related threats within the system environment. We follow the Common Attack Pattern Enumeration and Classification (CAPEC) [8] to identify the threats and derive possible attacker goals on the application context. Note that there are several other sources to identify the threat environment but for simplicity in this paper we consider only CAPEC for the case-study. We consider that the main intentions of the attacker are *unavailable single-sign-on service* and *obtain sensitive information*. Figure 8 shows the attacker scenario for *unavailable single-sign-on service* and how it can be decomposed to *band-width consumption* and *resource depletion* to obstruct the goal service availability. To accomplish these sub-goals, an attacker can spread malicious data by flooding the established communication link so that it can consume the bandwidth of the link or by sending unnecessary numerous requests to the target system to deplete the system resources. Note that the same attack can contribute to violate legal and security goals. On the other

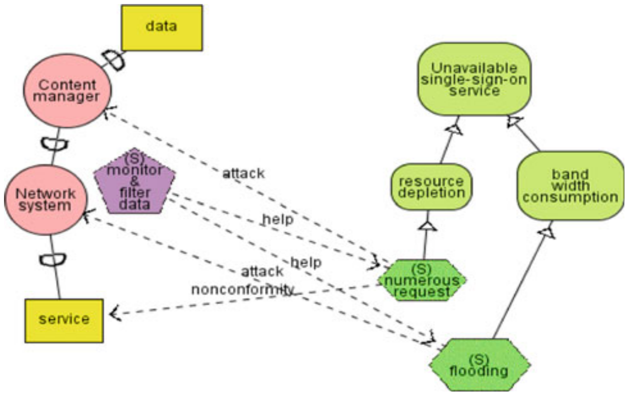

Fig. 8 Security attack scenario for service unavailability

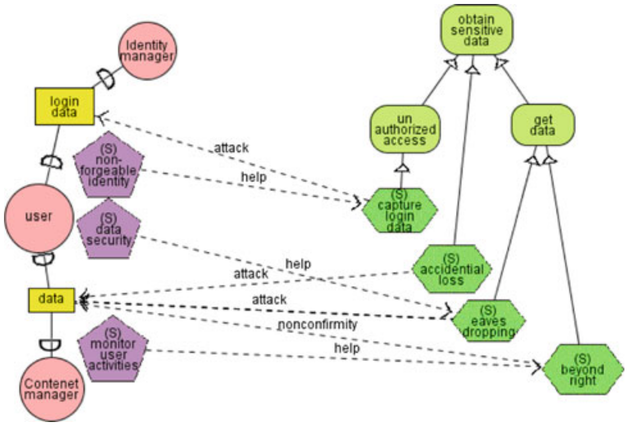

Fig. 9 Security attack scenario for obtain sensitive data

hand, a legitimate user intentionally might send numerous requests to the provider to consume the system processing resource. Therefore, these attacks contribute to the non-compliance issues along with security threats within the system environment.

Another goal of an attacker identified by our analysis is *Obtain sensitive data.* This can be decomposed to *get data* and *unauthorized access* to the system as shown in Fig. 9. To accomplish these sub-goals, an attacker might try to read the transmitted data by eavesdropping or in the case of a legitimate user (internal attacker, such as the data controller for example) might try to abuse his right to obtain data (e.g. no-right), or try to obtain login information. Once the attacker (or legitimate user) manages to obtain unauthorised data then this data may be processed unlawfully in a manner that it violates the purpose of data collection, or disclose the data in unauthorised fashion. There are also cases where incidents are accidentally caused by the system itself, such as accidental damage or loss of the data.

As discussed in the previous section, the next step of the process is to identify suitable countermeasures that the system might use to prevent the above identified attacks. To protect denial of service attacks, the system requires adequate technical measures including monitoring capability of the network traffic and filtering of unnecessary requests, therefore *monitor and filter data* as secure capabilities are essential to satisfy the service availability and secure service goals. To protect sensitive data, it is required that the system has secure capabilities such as *decrypt incoming data and encrypt outgoing data* as *data security* to control eavesdropping; and *monitor authorised user activities* to ensure that legitimate users do not abuse their rights. Moreover, the secure capability *non-forgeable identity* is required to protect towards duplication of login data. However, this capability on its own is not sufficient to protect against the attacker capturing login information, since an attacker may obtain certificate key by other means such as social engineering or theft of hardware devices.

Therefore, the above analysis assists developers in refining the initial security requirements or introducing new ones to ensure that the above secure capabilities are supported by the system.

Examples of refined security requirements are:

- The system shall monitor and if required filter incoming malicious traffic.
- Every data processing action by the data processing operator shall be tracked and controlled.
- The system shall prevent authorised users to process data that violates the purpose of the data collection, process, and use.
- The system shall encrypt all outgoing data through public communication network and decrypt all incoming data so that unauthorised users cannot access the data.

The above example also indicates the need for a new requirement:

- The system shall ensure that every user should have non-forgeable identity-related information to prove his/her identity.

5.4 Mapping Secure Tropos model to UMLsec model

The final step of our framework is the development of the secure design by mapping Secure Tropos models to UMLsec models. In doing so, initially we need to identify the UMLsec stereotypes, based on the security requirements along with the legal constraints. These stereotypes allow us to specify the security and legal constraints linked to the information flow and the processes carried out by the UMLsec model components. For the presented case study, we identified several stereotypes such as «fair service delivery», «legitimate process», « data filter», «secure links», «Internet», and «secrecy» to support the security requirements and the legal constraints. We then constructed relevant UMLsec class and deployment diagrams as a part of the architectural design. Figure 10 shows the «fair service delivery» and the « legitimate process» stereotypes integrated into a class diagram to support the alignment of security requirements and legal constraints. The stereotype «fair service delivery» has three associate tags {start}, {continue}, and {stop} to support the action related to services such as request data, deliver data, and close session. Therefore, this stereotype captures the requirements relating to the transaction of the work flow in the application context. To support the actor's behaviour of action regulated by legal rights and constraints, the «legitimate process» stereotype is included. It has three tags: {action}, {right}, and {resource}. The values of the {action} influence possible tasks performed by the actors within the system environment. The tag {right} is initiated from the extracted legal rights and constraints to regulate the action. Finally {resource} identifies the possible required resources within the system environment. Therefore, «fair service delivery» focuses on the security part, while « legitimate process » focuses on the legal part.

The four main classes shown in the Fig. 10 specify the participating entities within the context:

- *User*. User of the system with the following attributes: the IP address *Ip*, used to identify the user's machine; the data set *userdata*, which contains the name and the subscription information; the public key *key_e* and the associated private key *inv(key_e)* (used to encrypt and sign the relevant data); a session key *key_s* for establishing a session.
- *ServiceProvider*. Central administrator with the following attributes: the public key *key_l*, to verify the login token.
- *IdentityManager*. Login service provider with the following attributes: the key pair *key_l* and *inv(key_l)*, to serve the login service. It also contains the public key *key_ca* of the certification authority to decrypt and verify data.
- *ContentManager*. Data controller or data processing operator responsible for the delivery and restoration of the protected data. This class contains a privilege list *LP* from the owner and service provider to authorised user to access, process, and use data. A session key *key_s* is maintained during the established session between the user and the content manager. It also contains the public key *key_i* of the identity manager.

At this stage of the architectural design, it is important, from the security point of view, to understand the physical distribution of the nodes, in order to specify in more detail the security attributes of the system's communication links. To support this, our framework uses deployment diagrams as shown in Fig. 11. To satisfy the security constraints such as *keep communication secure*, legal goals such as *personal data protection*, or security goals such as *data processing integrity*, the UMLsec stereotypes «secure links» and «secrecy» have been employed to ensure that the security requirements and legal constraints on the communication are met by the physical layer. To ensure service availability, « data filter» stereotype is employed for both communication network as link and providers as nodes. Communication among the provider nodes and user can be done through the «Internet» or «LAN» stereotype. The deployment diagram also specifies how user equipment can initiate the login process with the signature key obtained from a smart card.

Finally, UMLsec sequence diagrams are constructed as part of the detailed design. Figure 12 shows a sequence diagram that describes the single-sign-on service feature along with identity and authenticity. The purpose is to establish a secure session to obtain data and restore processed data within the same or different session. Note that we assume that the user is already approved by the service provider.

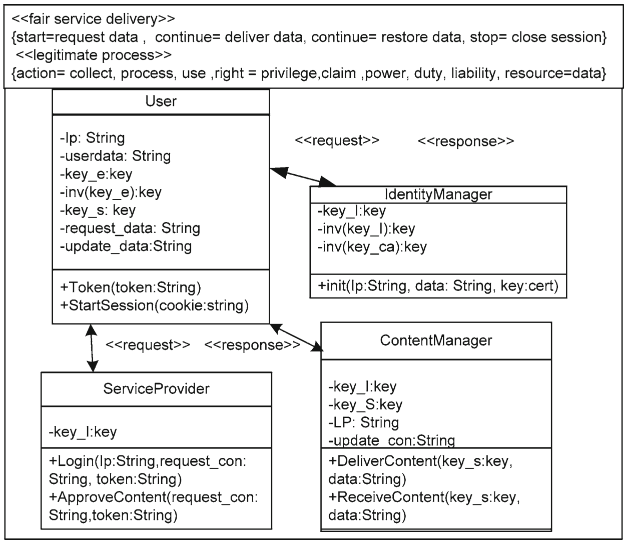
<<fair service delivery>>
{start=request data , continue= deliver data, continue= restore data, stop= close session}
<<legitimate process>>
{action= collect, process, use ,right = privilege,claim ,power, duty, liability, resource=data}
User
-Ip: String
-userdata: String
-key_e:key
-inv(key_e):key
-key_s: key
-request_data: String
-update_data:String
+Token(token:String)
+StartSession(cookie:string)
<<request>>
<<response>>
IdentityManager
-key_I:key
-inv(key_I):key
-inv(key_ca):key
+init(Ip:String, data: String, key:cert)
<<request>>
<<response>>
ServiceProvider
-key_I:key
+Login(Ip:String,request_con: String, token:String)
+ApproveContent(request_con: String,token:String)
ContentManager
-key_I:key
-key_S:key
-LP: String
-update_con:String
+DeliverContent(key_s:key, data:String)
+ReceiveContent(key_s:key, data:String)

Fig. 10 Class diagram

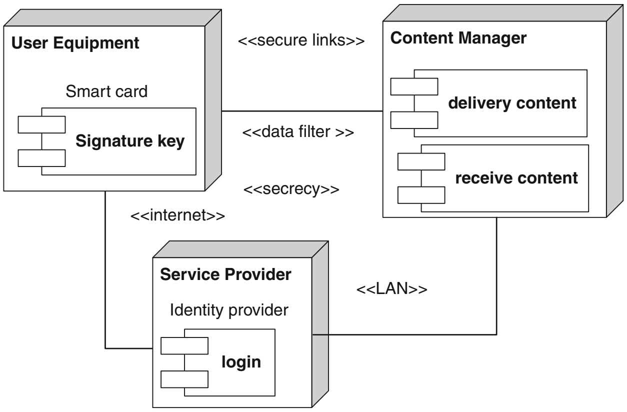

Fig. 11 Deployment diagram

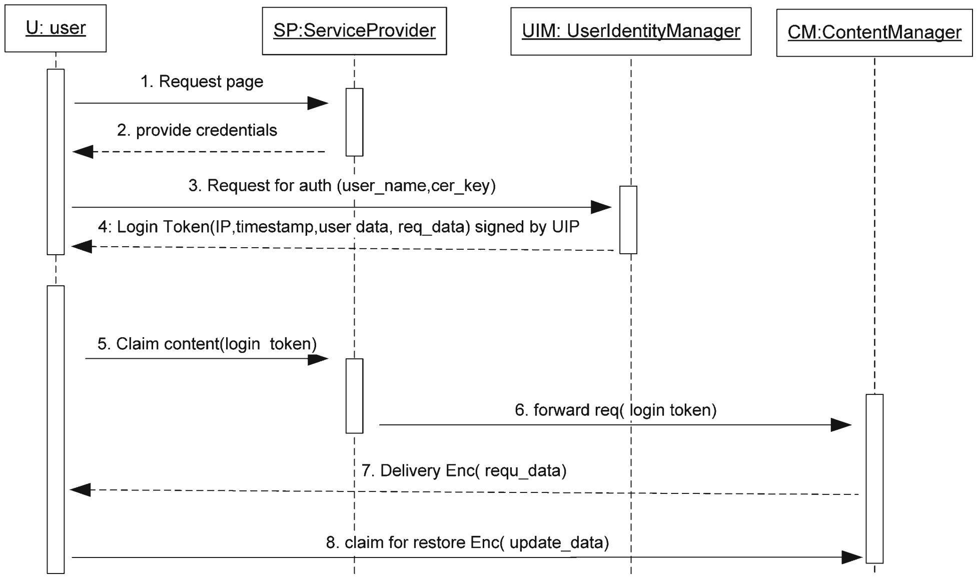
U: user
SP:ServiceProvider
UIM: UserIdentityManager
CM:ContentManager
1. Request page
2. provide credentials
3. Request for auth (user_name,cer_key)
4: Login Token(IP,timestamp,user data, req_data) signed by UIP
5. Claim content(login token)
6. forward req(login token)
7. Delivery Enc(requ_data)
8. claim for restore Enc(update_data)

Fig. 12 UMLsec sequence diagram

Initially, a user submits a request for a protected web-service page (a part of content (data)) to the Service Provider (SP).

The SP returns an html form to the user for identity information. The user further requests a login token by sending a certificate key, generated from a smart card, to the user identity manager (UIM). The UIM verifies the user identity and generates a login token (IP, time stamp, user data (name, subscription info)) and sends the signed token back to the identified user. The user with this signed login token claims data from the service provider. The service provider verifies the login token and forwards it to the Content Manager (CM). The CM initially verifies the owner consent to grant the data to the user and the associate level of authorization as part to fulfil the legal dependency. Finally the CM duly delivers the data. The user may further claim to restore the processed data within same session and/or in a different session. The CM then restores the data. All data transfers between the CM and the user are encrypted with the session key shared between the entities. The single-sign-on authentication mechanism addresses the unresolved attack *obtain login data* analysed in the corresponding security attack scenario. Because the login token consists of a user machine IP and a time stamp (current date and maximum time limit to use the token), if an attacker captures the token, they will still be unable to login due to different IP or exceeded time limit. The Https protocol is used to encrypt the data using the session key exchange between the user and the CM. This supports the *keep communication secure* constraint identified earlier in the analysis and the relevant stereotype *«data security»* with the capability for securely decrypting incoming data and encrypting outgoing data.

6 Conclusions

Compliance with regulations is becoming an important concern for software systems supporting critical organizational purposes. Our approach addresses one of the issues raised in recent research [34,41] regarding modelling and analysis of legal regulations during a software system development process. Laws change very often by amending or repealing old legislation or by introducing new legislation. This evolving situation is especially important and frequent for the information security laws. However, the current state of the art in the development of software systems fails to provide solutions for this issue. Our framework is a first step towards this direction.

Our framework provides several benefits. It supports elicitation and analysis of security requirements that align with legal issues from the requirements stage of the software development process. As it has been argued in recent research [22], this is very important. The framework also supports the development of designs that enable developers to trace back specific solutions to the relevant security and legal requirements, and therefore to the appropriate legal text. When an amendment (new or update article or section) of law is introduced, developers need to establish what goals, legal constraints, or relating artefacts the new (or updated) law introduces or modifies. For instance, when an article under a directive or a section under the FDPA is amended, then initially we should look at the goals due to the amendment then continue to review the artefacts in particular legal rights, constraints, and security requirements that can be affected by the change. If require new goals or constraints can be included

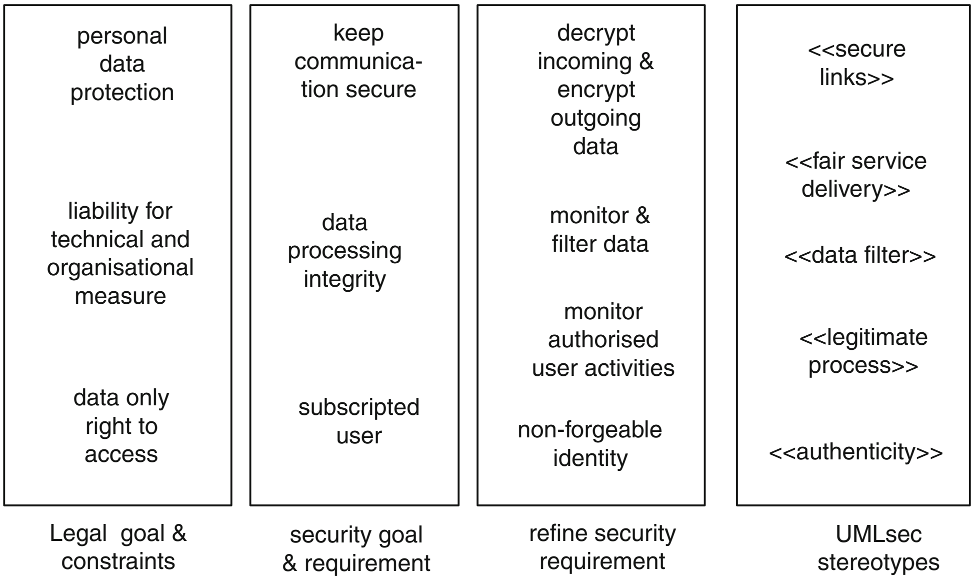
personal data protection
liability for technical and organisational measure
data only right to access
keep communica-tion secure
data processing integrity
subscripted user
decrypt incoming & encrypt outgoing data
monitor & filter data
monitor authorised user activities
non-forgeable identity
<<secure links>>
<<fair service delivery>>
<<data filter>>
<<legitimate process>>
<<authenticity>>
Legal goal & constraints
security goal & requirement
refine security requirement
UMLsec stereotypes

Fig. 13 Tracing from regulation modelling to security requirements and design

depending on the context of change. Thus modelling regulation by following goal-driven Secure Tropos allows us to adapt the change from the legal concept to the software that compliance with law. Moreover, the presented framework supports the analysis of legal text and the extraction of a number of concepts based on Hohfeld's legal taxonomy, such as legal privilege, claim, power, immunity, and their correlatives duty, no-right, liability, and disability. This is an important improvement to the existing works (see related work section), where only permitted and required actions are used to represent legal rights. As research has shown [4,43], this is not sufficient to represent stakeholder legal rights. It is also worth noting that the presented framework provides support for the unification and better understanding of concepts and terminology from two fundamentally different areas, i.e. law and software engineering, since it uses concepts and terms from the legal domain such as privilege, duty, and liability, together with concepts from the software engineering domain such as actor, goal, and task. Last but not least, our framework supports the identification of potential attacks for non-compliance to a specific software system application context and provides activities and steps to refine a system's requirements to overcome such risks.

To demonstrate the applicability of our framework and to test it against a number of hypotheses, we applied it to a case study. The case study showed that the framework sufficiently supports the extraction of legal constraints from legal texts. The elicited security requirements are aligned with legal constraints and can be further refined to address the security threats and non-compliance issues. Therefore, our framework supports hypotheses H1, H2, and H3 specified at the beginning of the case study. The case study also demonstrates that our framework supports tracing of legal constraints to security requirements and further to secure design. Figure 13 depicts the artefacts involved to support tracing from regulation to requirements and to design in the case study. The left part of the figure represents legal goals and extracted legal constraints. These legal artefacts were addressed by appropriate security goals and constraints and further refined by considering threat and non-compliance issues (shown in the middle part of the figure). Finally the right part of the figure shows the relevant stereotypes that support both security and legal aspects. As an example, consider a legal goal such as *personal data protection*, and legal constraints such as *liability for technical and organisational measure*, and *data only right to access*. These are addressed by a number of security goals and constraints identified during our analysis such as *keep communication secure*, *data processing integrity*, and *subscribed user*. These were further refined to security requirements such as *decrypt incoming data and encrypt outgoing data*, *monitor and filter data*, *monitor authorised user activities*, and *non-forgeable identity*. Finally, these requirements were supported during design by the stereotypes *«secure links»*, *<<fair service delivery»*, *<<data filter»*, *«legitimate process»*, and *«authenticity»*. The precise semantics of the stereotype enables developers to trace back the need to introduce them to the security requirements and further to the relevant legal documents. Therefore, our framework also supports hypotheses H4 and H5.

However, the framework does not address every ambiguity from legal text, especially when the legal text concerns with measuring some parameters (such as security level) that depend upon several other issues. To address this issue, we are planning to integrate risk management activities into the framework. Moreover, during the course of the framework development and its application to the case study, we have identified the need to develop appropriate tools. Currently, we are using tools that belong to some of the components of our framework such as the Secure Tropos tool and the UMLsec tool. However, neither of these tools directly supports to elicit legal rights, correlatives, and legal constraints, therefore we had to manually work to analyse the legal text during the case study. But once legal constraints are identified, then the Secure Tropos tool supports the analysis of the legal constraints along with security constraints through the security and legal dependency of the actor model. Further analysis for non-compliance issues of the system under development is also supported by the tool during the security attack scenarios analysis. Finally, the UMLsec tool can verify the security and legal property of the stereotypes. We are currently working to further automate the support provided by these tools. Moreover, we would like to analyse regulations of other domains such as the health care and financial domains.

References

1. Herrmann, A., Kerkow, D., Doerr, J.: Exploring the Characteristics of NFR Methods—a Dialogue about two Approaches, REFSQ—Workshop on Requirements Engineering for Software Quality (2007), Foundations of Software Quality (2007)
2. Herrmann, A., Paech B.: MOQARE: misuse-oriented quality requirements engineering. Requir. Eng. J. **13**(1), 73–86 (2008)
3. van Lamsweerde, A., Letier, E.: Handling obstacles in goal-oriented requirements engineering. IEEE Trans. Softw. Eng. Special Issue on Exception Handling **26**(10), 978–1005 (2000)
4. Siena, A., Mylopoulos, J., Perini, A., Susi, A.: From laws to requirements. In: 1st International Workshop on Requirements Engineering and Law (Relaw'08)
5. Bundesdatenschutzgesetz - Federal Data Protection Act (as of 15 November 2006), http://www.bfdi.bund.de.
6. Haley, C.B., Laney, R., Moffett, J.D., Nuseibeh, B.: Arguing satisfaction of security requirements. In: Integrating Security and Software Engineering: Advances and Future Visions, pp. 16–43. Idea Publishing Group, Miami (2006)
7. Haley, C.B., Laney, R.C., Moffett, J.D., Nuseibeh, B.: Security requirements engineering: a framework for representation and analysis. IEEE Trans. Softw. Eng. **34**(1), 133–153 (2008)
8. Common attack pattern enumeration and classification (CAPEC). http://capec.mitre.org/
9. Firesmith, D.: Engineering security requirements. J. Obj. Technol. **2**(1) http://www.jot.fm/issues/issues_2003_01/column6 (2003)
10. Mellado, D., Medina, E., Piattini, M.: A common criterion based security requirements engineering process for the development of secure information system. Comput. Stand. Interfaces **29**, 244–253 (2007)
11. Massacci, F., Prest, M., Zannone, N.: Using a security requirements engineering methodology in practice: the compliance with the Italian data protection legislation, Technical Report DIT-04-103 (2004)
12. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. Requir. Eng. **10**(1), 34–44 (2005)
13. Sartor, G.: Fundamental legal concepts: a formal and teleological characterisation, EUI working paper LAW No. 2006/11
14. Mouratidis, H., Giorgini, P.: Integrating Security and Software Engineering: Advances and Future Visions. Idea Group Publishing, Miami (2006)
15. Mouratidis, H., Jürjens, J., Fox, J.: Towards a comprehensive framework for secure systems development, CAiSE 2006. Lecture Notes in Computer Science, vol. 4001, pp. 48–62. Springer, Berlin (2006)
16. Mouratidis, H., Giorgini, P., Manson, G.: When security meets software engineering: a case of modelling secure information systems. Inf. Syst. Elsevier **30**(8), 609–629 (2005)
17. Mouratidis, H.: A security oriented approach in the development of multiagent systems: applied to the management of the health and social care needs of older people in England. PhD thesis, University of Sheffield, UK (2004)
18. Mouratidis, H., Giorgini, P.: Security Attack Testing (SAT)—testing the security of information systems at design time. Inf. Syst. **32**(8), 1166–1183 (2007)
19. Mouratidis, H., Giorgini, P.: Integrating security and software engineering: an introduction. In: Integrating Security and Software Engineering: Advances and Future Actions, pp. 1–14. Idea Publishing Group, Miami (2006)
20. Mouratidis, H., Giorgini, P.: Secure tropos: a security-oriented extension of the tropos methodology. Int. J. Softw. Eng. Knowl. Eng. (IJSEKE) **17**(2), 285–309 (2007)
21. Information society, Summary of legislation, European Commission, http://europa.eu/legislation_summaries/information_society/index_en.htm
22. Saltzer, J., Schroeder, M.: The protection of information in computer systems. Proc. IEEE **63**(9), 1278–1308 (1975)
23. Jürjens, J.: Secure Systems Development with UML. Springer, Berlin (2004)
24. Jürjens, J., Shabalin, P.: Tools for Secure Systems Development with UML. FASE 2004/05 special issue of the International Journal on Software Tools for Technology Transfer, **9**(5–6), 527–544. Springer, Berlin (2007)
25. Jürjens, J.: Sound methods and effective tools for model-based security engineering with UML. ICSE 2005, ACM, pp. 322–331, (2005)
26. Chung L., Nixon B.A., Yu E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Dordrecht (1999)
27. May, M.J., Gunter, C.A., Lee, I.: Privacy APIs: access control techniques to analyze and verify legal privacy policies. In: Proceedings of the 19th Computer Security Foundations Workshop (2006)
28. Medical Privacy—National Standards to Protect the Privacy of Personal Health Information. Office for Civil Rights, US Department of Health and Human Services. http://www.hhs.gov/ocr/hipaa/finalreg.html (2000)
29. Mead, N.R.: Identifying security requirements using the security quality requirements engineering (SQUARE) method. In: Integrating Security and Software Engineering, pp. 44–69. Idea Publishing Group, Miami (2006)
30. Online news of November 15, 2004, http://digital.dmreview.com/dmreview
31. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: TROPOS: an agent oriented software development methodology. In: Journal of Autonomous Agents and Multi-Agent Systems, **8**(3), 203–236. Kluwer Academic Publishers, Dordrecht (2004)
32. Devanbu, P., Stubblebine, S.: Software engineering for security: a roadmap. In: Proceedings of ICSE 2000 (the Conference of the Future of Software Engineering) (2000)
33. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Modelling security requirements through ownership, permission and delegation. In: Proceedings of the 13th IEEE International Requirements

Engineering Conference (RE'05), IEEE Computer Society Press, 29 August–2 September (2005)

34. Otto, P.N., Antón, A.I.: Addressing legal requirements in requirements engineering. In: 15th IEEE International R. E. Conference (2007)
35. Privacy Guidelines for Developing Software Products and Services, Version 3.1, September, 2008, http://download.microsoft.com
36. Darimont, R., Lemoine, M.: Goal-oriented analysis of regulations. In: Proceedings of the CAISE06 Workshop on Regulations Modelling and their Validation and Verification (ReMo2V '06). Luxemburg, 5–9 June 2006. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-241
37. Islam, S., Dong, W.: Security requirements addressing security risks for improving software quality. In: Workshop-Band Software-Qualitätsmodellierung und -bewertung (SQMB '08), Technical Report TUM-I0811, Technische Universität München, 2008, Munich, Germany (2008)
38. Islam, S., Jürjens, J.: Incorporating security requirements from legal regulations into UMLsec model, Modelling Security Workshop (MODSEC08). In: Association with MODELS '08, Toulouse, France, September (2008)
39. Ghanavati, S., Amyot, D., Peyton, L.: Towards a framework for tracking legal compliance in healthcare. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) 19th International Conference on Advanced Information Systems Engineering (CAiSE'07), pp. 218–232. Springer, Berlin (2007)
40. Breaux, T.D. Vail, M.W., Antón, A.I.: Towards regulatory compliance: extracting rights and obligations to align requirements with regulations. In: Proceedings of the 13th IEEE International Conference on Requirement Engineering (2006)
41. Breaux, T.D, Antón, A.I.: Analyzing regulator rules for privacy and security requirements. IEEE Trans. Softw. Eng. **34**(1), 5–20 (2008)
42. Breaux, T.D., Antón, A.I.: Deriving semantic models from privacy policies. In: IEEE 6th Workshop on Policies for Distributed Systems and Networks, Stockholm, Sweden, pp. 67–76 (2005)
43. Hohfeld, W.N.: Fundamental legal conceptions as applied in judicial reasoning. Yale Law J. **23**(1) (1913)

Author Biographies

Shareeful Islam is currently a Ph.D. student at the research group Software and System Engineering, Institut für Informatik (I4) of the Technische Universität München, Germany. He received the M.Sc. degree in Information Communication System Security from the Royal Institute of Technology / Stockholm University in 2004. He also received M.Sc. degree in Computer Science and B.Sc. (Hon's) in applied physics and electronics from the University of Dhaka, Bangladesh in 2000 and 1998. He completed the ISO 9001:2001 lead auditor certification. Before starting his Ph.D., he worked as an Assistant Professor at the Institute of Information Technology (IIT), University of Dhaka, Bangladesh. His main research interests are in the field of software development risk management and software security. Special interests are risk management model and security requirements engineering.

Haralambos Mouratidis holds a B.Eng. (Electronics with Computing Science) from the University of Wales, Swansea, UK and M.Sc. (Data Communications) and Ph.D. (Secure Software Engineering) degrees from the University of Sheffield, U.K. Dr. Mouratidis is Principal Lecturer in Secure Systems and Software Development at the School of Computing, Information Technology and Engineering (CITE), University of East London. His research interests are related to security requirements engineering, secure information systems development and methodologies and methods for secure software systems. He has been involved in the organisation of national and international events related to his research interests as General Chair (e.g. SASEMAS), Programme Chair (e.g. AOIS, AC&T, ICGeS), and as Programme Committee member (e.g. CAiSE, ICEIS, SECRYPT, IEEE IAT). He is Editor in Chief of the International Journal of Computer Science and Security and he has reviewed for a number of journals including IEEE Transactions on Software Engineering, the International Journal of Software Engineering and Knowledge Engineering, the Software Quality Journal, the International Journal on Information and Software Technology, the International Journal of Software and Systems Modelling, and the Data & Knowledge International Journal. Dr. Mouratidis is (and has been) actively involved in various international projects and collaborations and he is member of the Special Interest Group on Secure Software Development of the Cyber Security Knowledge Transfer Network in UK (http://www.ktn.qinetiq-tim.net). He has hold a visiting research fellowship with British Telecom (BT) and he has received funding from the Engineering and Physical Sciences Research Council (EPSRC, UK), Higher Education Funding Council for England (HEFCE)/University of East London, the Royal Academy of Engineering and the National Institute of Informatics (NII, Japan). He has published more than 80 referred papers in high quality journals (such as Information Systems, Computers & Security) and international conferences (such as IEEE RE, CAiSE, and ER). He is editor of a special issue of the Requirements Engineering Journal (on Security Requirements Engineering) and of a forthcoming book on Software Engineering for Secure Systems: Industrial and Research Perspectives from IGI Publishing.

Jan Jürjens is a Professor at the Chair for Software Engineering (LS 14), Department for Computer Science, Technical University Dortmund (Germany), the Scientific Coordinator "Enterprise Engineering" and Attract research group leader at the Fraunhofer Institute for Software and Systems Engineering ISST (Dortmund), and a Senior Member of Robinson College (Univ. Cambridge, UK). He is PI of several projects financed by Microsoft Research (Cambridge), British Telecom, and EPSRC, and Research Director of an Integrated Project financed by the EU within FP7, Future and Emerging Technologies Programme. Previously, he was a Royal Society Industrial Fellow at Microsoft Research Cambridge and non-stipendiary Research Fellow at Robinson College (Univ. Cambridge). Before that, he was a

Postdoc at the chair for Software and Systems Engineering, TU Munich (Germany). He holds a Doctor of Philosophy in Computing from the University of Oxford and is author of “Secure Systems Development with UML” (Springer, 2005; Chinese translation: Tsinghua University Press, Beijing, 2009) and various publications mostly on computer security and software engineering, totalling over 1500 citations. Much of his work is done in cooperation with industrial partners including Microsoft Research (Cambridge), O2 (Germany), BMW, HypoVereinsbank, Infineon, Deutsche Telekom, Munich Re, IBM-Rational, Deutsche Bank, Allianz.