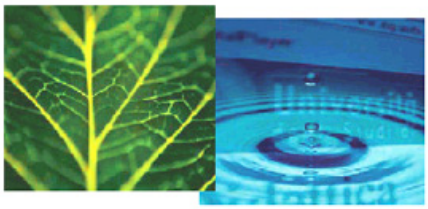
PhD Dissertation



International Doctorate School in Information and Communication Technologies

DISI - University of Trento

ENGINEERING LAW-COMPLIANT REQUIREMENTS
THE *Nòmos* FRAMEWORK

Alberto Siena

Advisor:

Prof. Angelo Susi

Fondazione Bruno Kessler - Irst-CIT

March 2010

"Thou great star! What would be thy happiness if thou hadst not those for whom thou shinest!"

(Friedrich Wilhelm Nietzsche)

Abstract

In modern societies, both business and private life are deeply pervaded by software and information systems. Using software has extended human capabilities, allowing information to cross physical and ethical barriers. To handle misuse dangers, governments are increasingly laying down new laws and introducing obligations, rights and responsibilities concerned with the use of software. As a consequence, laws are assuming a steering role in the specification of software requirements, which must be compliant to avoid users be exposed to fines and penalties.

This work proposes a model-based approach to the problem of law compliance of software requirements. It aims at extending state-of-the-art goal-oriented requirements engineering techniques with the capability to argue about compliance, through the use and analysis of models. It is based on a language for modelling legal prescriptions. Upon the language, compliance can be defined as a condition that depends on a set of properties. Such a condition is achieved through an iterative modelling process.

Specifically, we investigated the nature of legal prescription to capture their conceptual language. From juridical literature, we took a taxonomy of legal concepts, which has been elaborated and translated into a conceptual meta-model. It is then bound with the meta-model of a goal-oriented modelling language for requirements engineering, in order to provide a common legal-intentional meta-model.

Requirements models built with the proposed language consist in graphs,

which ultimately can be verified automatically. Compliance amounts then in a set of properties the graph must have.

The compliance condition gains relevance in two cases. Firstly, when a requirements model has already been developed, and it needs to be reconciled with a set of laws. Secondly, when requirements have to be modelled from scratch, and they are wanted to be kept compliant. In both cases, compliance results as the product of a modelling process.

The developed modelling language, as well as the compliance condition and the relative process, have been applied to two case studies. The obtained results confirm the validity of the approach, and point out interesting research directions for the future.

Keywords

Requirements Engineering, Law Compliance, Goal Orientation

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The history of computer science is characterised by two important milestones. The first one is the invention of modern computing machines - the *computers*. The second one is the interconnection of computers into a worldwide network - the *Internet*. If the first milestone is a clear dividing line for technology, the second milestone gets a social revolution underway. In the Internet era, computers communicate with each other to accomplish tasks that were otherwise not possible. This transformed computing machines into complex entities such as software systems, software-intensive systems, or socio-technical systems, up to that innovative yet vaguely defined concept of Future Internet as a single, global infrastructure comprised by several sub-system and components working together. Such systems support processes and business, governments and companies, as well as professional, social and private life. Living in the Internet era has become dependent on the communication capabilities of computers: a report of the U.S. Department of Labor states that for the last five consecutive years the most common disruption to business was the loss of broadband internet connection, and concludes that long-term failure could be catastrophic[1].

The real life and the digital life are nowadays interleaved. What happens in the real life can be propagated through electronic highways to the world, and

[1] `http://ezinearticles.com/?Internet-Reliability-Most-Common\-Disruption-to-Business,-HDTV-Offers-New-Threat&id=1241165`

what happens in the digital world can turn out into real events. This fact raised the attention of governments on the need to regulate by law citizens' electronic interaction. In recent years, new laws have been enacted to explicitly regulate sensitive matters, such as business and health assistance, when performed on networked systems. Also, previous laws have gained new meaning when referred to an Internet-based activity.

The social relevance of information systems impacts on the way the information system has to be conceived. If misaligned with legal prescriptions, a functionality of the system can violate the rights of users, and ultimately breaches the law. Nevertheless, it's trivial mentioning that the system itself is not responsible for the breach, and someone else - the committer? the administrator? the analyst? - will pay for it. Preventing this situation to happen is in the hand of those who are called to define the system's functionalities: the requirements analysts.

1.1 Research problem

Goal-oriented requirements engineering (GORE) rests on the idea of deriving the requirements for a software system from the analysis of the goals that the system-to-be will support once developed and deployed. However, when the stakeholders are addressed by laws, the system-to-be has to be aligned with the legal prescriptions too, and goals, *per se*, don't provide information about such alignment. This is the problem of law compliance of goals models.

Finding a solution to this problem means finding the assignment of actors responsibilities (goals) such that if every actor fulfils its goals, then law is respected. This is the condition or law-compliance of requirements, and we derive a general rule to say that a requirements set is compliant with a law:

$$R, K \models L \tag{1.1}$$

which means that, given a set of requirements R, represented as actors goals,

and a set of domain assumptions K, the requirements are compliant with a law L if, for every possible state of the world, if R holds, then L holds.

To produce a usable solution to the problem of law compliance of requirements, we adopt a modelling approach, which consists in starting from a model of legal prescriptions, and then building the model of goals in an incremental way that maintain the alignment with the prescriptions. The specific problem that we are trying to address here, is to provide a definition for the notion of "alignment" in terms of the necessity or possibility for certain strategic element to exist or not.

1.2 Contribution

The contribution of this thesis is summarised as a set of conceptual tools for modelling legal prescriptions, a method to analyse the models for compliance, and a methodology to guide analysts through the modelling and analysis phases.

Conceptual tools The first ingredient for proposing a solution to the above problem is the identification of the proper conceptual tools. Law is a discipline with its own history, philosophy and language, and if we want to address it, we have to draw from its world. A milestone of juridical literature is Hohfeld's taxonomy of legal rights, which describes the building blocks of law. The taxonomy is comprised by the 8 concepts of *duty*, *claim*, *privilege*, *no-claim*, *power*, *liability*, *immunity* and *disability*, structured in opposites and correlatives. Upon that taxonomy is built the meta-model of the Nomos framework. Such a meta-model provides the syntax for a modelling language that integrates legal concepts with intentional elements. Specifically, we integrated the Hohfeldian taxonomy with the i^* set of concepts. This results in a language for modelling legal prescriptions alongside stakeholder requirements. The language also provides a visual notation for representing models, which extends the i^* modelling

language, adding the capability to specify the rights existing between actors of the domain. Moreover, it is able to represent legal rights in the rationale of stakeholders, and to link them with actor goals and plans.

Compliance analysis The conceptual tools represent a means to argue about compliance or not compliance of a requirements model. In other words, they are used to provide a definition of the requirements compliance problem of Equation (1). We move from the idea that legal prescriptions carry legal knowledge representable with such concepts. Physically, legal texts are documents written in natural language, but they can be broken down legal into their most atomic elements, the *normative propositions*. A normative proposition is a proposition that contain the nature of the right (duty, claim, etc.), the holder of the right, its counter-party, and the action to which the right applies. The first step to define compliance becomes this way the definition of compliance with respect to a single normative proposition. Afterwards, compliance to the law as a whole is defined as the sum of compliance to their single propositions. In doing this, a special attention is paid to the problem of complexity. Specifically, is shown how a large number of alternatives may arise from the conditions and exceptions that law contains, as well as from the strategic alternatives that exist to comply with law. The models will retain properties of (i) workability, being the compliance choice feasible with respect to actor capabilities; (ii) auditability, because the compliance solution will include requirements for auditing the running system and monitor its compliance; and (iii) traceability of the audits and processes back to their originating compliance choices, and up to the source laws.

Process Finally, the use of language and the tools for compliance analysis has been coded into a methodological process, which aims at incrementally building models of compliant requirements. It illustrates the steps to be done to build

models of requirements that comply with law, or to reconcile with the law existing requirements. The process covers those aspects of law compliance that are not related with the capabilities of goal-based models. A first aspect is the reconciliation of law-level subjects with domain-level stakeholders; in other words, if the law speaks in very generic terms (such as "the citizen" or "the covered entity"), in the domain we have more specific actors, such as "the user" or "the administrator". Covering the discrepancies between the two worlds is done at a methodological level. The identification of relevant laws and law fragments is also done here, to avoid that the models to be built become too big and complex. Strictly related to both the previous activities, is the exploration of legal texts to discover applicable laws, which may lead to include new actors, and this way bring to the identification of new requirements.

Additionally, we report the application of the proposed framework to three case studies. The first case study, developed as a sandbox scenario for testing the framework, concerns the generation of law-compliant requirements to the U.S., Health Insurance Portability and Accountability Act. The second case study, developed within an industrial project, concerns the verification of law compliance of partially defined requirements with respect to the Italian privacy law. The third case study, developed within a European project, concerns the gathering of requirements from implicit knowledge, exploiting laws as an additional source of information.

1.3 Thesis structure

The thesis is organised as in Figure 1.1. Chapter 2 survey recent work on requirements engineering applied to the problem of law compliance. Chapters 3 presents a theoretical definition for the desired solution to the problem. Chapter 4 introduces the requirements engineering modelling framework for arguing about compliance of requirements models. Chapter 5 makes an analysis of the

compliance condition and make explicit how to define it in terms of models. Chapter 6 gives a set of guidelines and a structured process to use the modelling and analysis framework in practice. Chapter 7 illustrates the application of the framework to two case studies. Finally, Chapter 8 concludes our work and discusses future research directions.

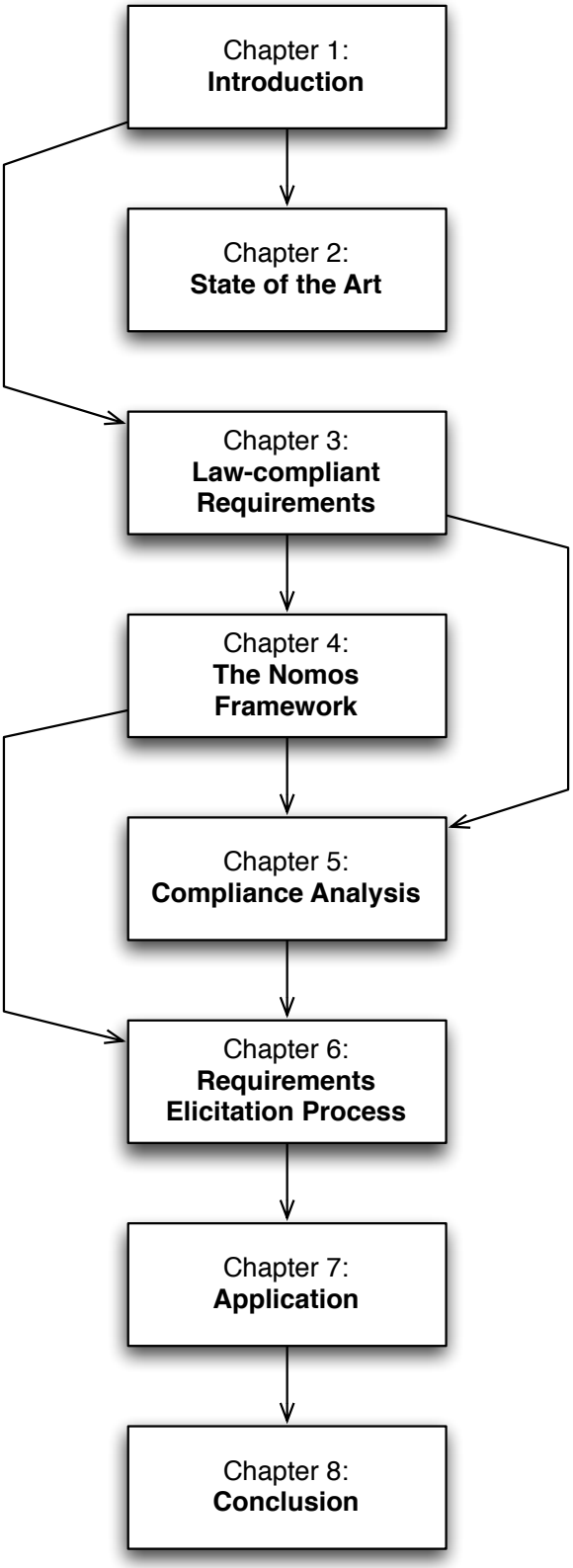


Figure 1.1: Thesis outline

Chapter 2

State of the art

An important part in the comprehension of the requirements compliance problems stays in the disambiguation with respect to others branches of information technology. The present work has required a cross-disciplinary deepening of both requirements engineering and legal literature. We will present in the following those research results that have been relevant for shaping our work. We will touch in particular four research areas: formal logics for automatic legal reasoning and artificial intelligence; knowledge representation of law-related information; normative multi-agent systems; and very recent literature on the specific theme of legal compliance in requirements engineering.

2.1 AI and Law

Law firstly came into the scene of computer science with the efforts made by logicians to represent legal prescriptions in a formal shape. The idea, pointed out by Allen (Allen, 1957) was that it is possible to formalise law as a logical theory, in order to be able to make conclusions about the consequences of the law.

Stamper proposed LEGOL (Stamper, 1980), a LEGally Oriented Language aimed at expressing a legislation by means of formal rules, suitable to be processed by computers. The idea behind LEGOL was that legislations shape the

behaviour of information systems: regardless on *how* information systems are implemented, they perform within the boundaries created by laws.

McCarty's TAXMAN system (McCarty, 1980) is an (experimental) application of artificial intelligence techniques to the study of legal reasoning and legal argumentation, using corporate tax law as problem domain. It allows to build models of facts comprising tax-related cases, such as corporations issue securities, transfer property, distribute dividends, and so on. Law (specifically: the United States Internal Revenue Code) is represented as rules, which classify transactions as taxable or nontaxable, ordinary income or capital gains, dividend distributions or stock redemptions, etc. Testing on several stock dividend and corporate reorganization cases decided by the Supreme Court in the 1920's and 1930's.

Following this experimental work, McCarty developed a Language for Legal Discourse (LLD) (McCarty, 1989), based on deontic logic, to specify actions as well as deontic operators. In LLD, one can specify the rule that any corporation that owns cash has an obligation to distribute cash to all its stockholders in a Lisp-like syntax.

A well known example of this approach is the work by Kowalski et al., who try to represent the British Nationality Act as a logic program (Sergot et al., 1986). That approach has been reported to be successful in the given context, suggesting the feasibility for its application to a larger class of cases.

As will be mentioned at the very end of the present work, although automatic reasoning hasn't achieved the ambitious objectives it was hoped to have, it can play a interesting role in the future of law-compliant requirements research. The principles of legal reasoning and argumentation cover a significant gap in support modelling choices. And since modelling and modelling choices constitute the core of the present work, we expect to be able to link the above mentioned results with our compliance modelling theory.

Nevertheless, worth saying that the legal reasoning approaches developed in

AI are hardly suitable for being directly used in requirements engineering, due to various reasons. Firstly, such approaches are not targeted to provide a representation of typical requirements engineering concepts, such as stakeholder, objective, preference and so on. Moreover, formal logic approaches are not usable by non-experts, so that they result useless during the requirements engineering phase, which generally involves a continuous interaction with stakeholders to check that the work of the analysts has capture their needs and preferences. Finally, automatic reasoning has never been completely independent from human intervention, because the need always exists, to provide an input to automatic reasoners and interpret their output; so the problem still exists, how to support analysts to perform every activity that can't be left to the machine.

2.2 Knowledge Representation

Kelsen (Kelsen, 1991) proposed a general theory of norms, in which he classifies norms in four categories: command norms, empower norms, permit norms and derogate norms. Commanding norms command to to (obligation) or not to do (prohibition) a certain action. Empowering norms associate roles with the power to posit and apply norms under certain restrictions. Permitting norms refer to what he called the positive sense of permission. Finally, derogating norms alter the validity of another norm, which still exists, but it is no longer valid.

From a different standpoint, Hart classifies norms as primary and secondary (Hart, 1961). Primary norms refers to human behaviour, whereas secondary norms, meta-level of the first, contain knowledge about primary norms. Secondary norms may further be classified as rules of adjudication (to provide support for solving conflicts), rules of recognition (to specify the limits of the legal system), and rules of change (to specify how the legal system can change in time).

Valente's FOLaw - Functional Ontology of Law - is an ontology of legal system, which adopts a legal-sociological view rather than a law-only perspec-

tive (Valente, 1995). The first purpose of FOLaw was to distinguish the various types of knowledge in legal reasoning, and in particular those types that are typical for legal reasoning. Knowledge for legal reasoning is classified in six main classes: normative knowledge, world knowledge, responsibility knowledge, reactive knowledge, creative knowledge and meta-legal knowledge.

2.3 Multi-Agent Systems

A large literature exists on Norms and Multi-Agent Systems (MAS), and we use some of these works as a foundation for our framework. A normative multi-agent system (NMAS) is defined as *"a multi-agent system together with normative systems in which agents on the one hand can decide whether to follow the explicitly represented norms, and on the other the normative systems specify how and in which extent the agents can modify the norms"* (Boella et al., 2007).

Moses and Tennenholtz (Moses and Tennenholtz, 1995) have defined the concept artificial social system as a multi-agent system in which agents behaviour is constrained by a set of rules, called social laws or social norms. Such norms are defined in terms of states of the system and abilities of the agents, and allow agents to co-exists in the same system. Although norms are defined at design-time, they are applied and monitored at run-time. However, their formalisation effectively captures the relation between agency and norms, and we took inspiration fro their works.

An Electronic Institution (Esteva, 2003), or e-Institution, is a multi-agent system designed as a variation of a finite state machine. Roles, which the agents will be able to play, are explicitly modelled and articulated with respect to their separation of duties. Agents play roles, and behave within "scenes". Scenes define the states, which the multi-agents system will exist in. A collection of scenes is a "performative structure". It defines the illocutions (messages) that agents can exchange and the transitions from one scene to another. An

e-Institution, as a whole, is the normative structure that articulates agents. E-institutions are open systems, so agents with unpredictable behaviour can join the institution and break the system. When a violation occurs, it results in a set of (punishment) messages that the enforcing agents have to send, to restore the institution into an acceptable state.

Our software engineering task of having a person check for compliance between a model of law and another of requirements is different from that of formalising law for purposes of automatic question-answering and reasoning. As in most other areas of Software Engineering, we assume that our models (of law, requirements or whatever) only capture some of the formal properties of the things being modelled, while the rest is left to humans (software engineers, lawyers etc.) who are ultimately responsible for checking for compliance and/or deriving designs. Our task is then to offer useful models of the artefacts of interest (in our case, laws and requirements), along with systematic processes for performing software engineering tasks, preferably with tool support.

2.4 Requirements

In recent years, the problem of law has begun to be considered in the earlier phases of software engineering. The results, produced in attempting to formalise law for AI, have shown that law can't be processed in a fully automated way — human intervention is still needed to interpret and disambiguate legal statements. Particularly, governments are recently laying down laws whose content hardly fits in automatised approaches. Laws, regulating the use of Information Technology in fields such as privacy, health, finance, and other human-related areas, easily miss the link with state-of-the-art technologies, and other technical aspects related to their use and misuse. For this reason, recent IT-related laws tend to state more abstract principles, instead of detailed procedural rules, leaving to the subjects responsibility to translate such principles into

concrete behaviours. This way laws are capable to embrace a larger number of behaviours. The gap introduced between legal prescriptions and concrete behaviour is matter of study for requirements engineering.

A first, relevant problem to be solved when facing legal prescriptions is the complex structure of legal documents. Anton and Breaux have developed a systematic process, called semantic parameterisation (Breaux et al., 2008), which consists of identifying in legal text restricted natural language statements (RNLSs) and then expressing them as semantic models of rights and obligations (Breaux et al., 2006) (along with auxiliary concepts such as actors and constraints).

Secure Tropos (Giorgini et al., 2004) is a framework for security-related goal-oriented requirements modelling that, in order to ensure access control, uses strategic dependencies refined with concepts such as trust, delegation and permission, to fulfil a goal, execute a task or access a resource, as well as ownership of goals or other intentional elements. We use that framework to ensure that compliance decisions, once made, are not compromised through the delegation chains in an organisational setting. The main point of departure of our work is that we use a richer ontology for modelling legal concepts, adopted from the literature on law. Models based on the law ontology allow to reason about where and how do compliance properties of requirements are generated.

Along similar lines, Darimont and Lemoine have used KAOS as a modelling language for representing objectives extracted from regulation texts (Darimont and Lemoine, 2006). Such an approach is based on the analogy between regulation documents and requirements documents. KAOS goal models express the goals of a law - i.e., why the law has been introduced by the legislator. From such high-level goals, be decomposition, concrete law articles are modelled as goals and attached to their reason-to-be. An object model defines the terminology used by the law. An agent model defines the agents that are addressed by the law. Finally, an operation model represents the behaviour of the addressed

agents.

Ghanavati et al. (Ghanavati et al., 2007) use URN (User Requirements Notation) to model goals and actions prescribed by laws. URN combines two complementary notations: the Goal-oriented Requirements Language (GRL) and Use Case Maps (UCM). This work is founded on the premise that the same modelling framework can be used for both regulations and requirements. GRL is used to create a model of law. The model of law are models of goals, create by extracting them from legal documents. At the same time, a goal model for the organisation is created, which describes the needs and objectives of the hospital. Finally, UCM diagrams describe the processes running inside the organisation. The framework adds then traceability links between GRL elements and UCM elements. Such links are used to keep track of which part of the business processes are intended to guarantee the achievement of a certain goal. On the other hand, the then-existing requirements tracking tool Telelogic DOORS was used to link goals with the place, in law source, where they were extracted from.

Likewise, Rifaut and Dubois use *i** to produce a goal model of the Basel II regulation (Rifaut and Dubois, 2008). Basel II is the document published in 2004 and produced by the second Basel Accord as a set of directives and recommendations to be used as a standard by regulators of banking systems. It indicates the allowances that banks (and therefore the whole financial system) need to maintain to be reasonably protected against various types of financial risks. In their approach, Rifaut and Dubois rely on the internal structure of Basel II to map its content into goal models. This can be done because Basel II gives a specific taxonomy of business process goals and assurance goals, which are structured into assurance aspects. A business process is described in terms of its purpose its outcome, which is evaluated through indicators. Purposes are mapped into soft-goals, whereas outcomes and indicators are mapped into hard goals, since they are unambiguously measurable. Multiple assurance aspects are modelled this way.

In summary, state-of-the-art approaches to law compliance for goal-oriented requirements engineering mainly rely on seeing goal descriptions as "embedded" into legal documents. We previously experimented this goal-only approach in the Normative *i** framework (Siena et al., 2008a; Ghanavati et al., 2010). That experience gave significant result in terms of gathering implicit knowledge from stakeholders. However, it was not possible to sistematise the goal modelling activity due to a lack in legal conceptualisation, which in turns made necessary a relevant creativity effort for modelling.

Ensuring law compliance consists then in extracting such goals from legal documents - either in a partially automated way or fully manually - and reconciling them with stakeholders objectives. Goal-only approaches however don't consider (or it as implicit) that the transformation between something legal (legal statements) and something strategic (goals) requires *decision making*. It in turns requires (i) legal knowledge: knowledge and skills on law, law interpretation and implications; (ii) domain knowledge: knowledge on stakeholders needs and expectations; and (iii) engineering knowledge: skills on how the requirements engineering process should be, and how to gather, validate and specify stakeholder goals. The actor, in charge of merging these three aspects, is the requirements engineer. The requirements engineer in its law-related decisions does not have, in our knowledge, does not have a complete and systematic support in other existing frameworks.

Chapter 3

Law-Compliant Requirements

Requirements engineering basically falls back on linking the purpose of an information system with the problem it is supposed to solve (Nuseibeh and Easterbrook, 2000). As of Jackson and Zave formulation, solving the requirements engineering problem amounts in finding a solution (the specification) such that, with the proper domain assumptions, if the specification holds, then the requirements hold: $S, K \vdash R$ (Zave and Jackson, 1997). Roughly speaking, that formula says that the purpose of the information system, represented as the specification S, is a solution for the problem represented as stakeholders requirements R. If laws and regulations exist, which have effect on the information system, then a part of the problem is to engineer a solution such that it satisfies stakeholders goals and at the same time complies with such laws (Siena, 2007). This is the problem of law compliance of requirements. In this chapter, we will provide a characterisation of the compliance problem, giving a definition of what are the properties of law relevant for requirements engineering, and how we expect a solution should be conceived.

3.1 A Primer on Law

On the essence of law there is a millenary debate involving philosophy and jurisprudence concerning what law actually is. It has the power to regulate and

give reason to human actions in a way that presupposes characteristics universal to every culture. Explaining how does it achieve this result involves both the explanation of its normative means, as well as the psychological aspects that make human being acknowledge such normative power. We will not enter such philosophical problems, trying instead, for the purpose of the present work, to adopt a more concrete notion of law. We can do this, because the universality of normative power implies a conceptual independence from specific aspects of legislations such as the language, the institutional structure of countries and so on.

Law forms the legal infrastructure of countries, and shapes their politics, economics and society in numerous ways. Law is primarily a human *artefact*, resulting from human conception and action, and so working in practice with law involves studying the physical artefacts that form law: legal sources.

There is a wide number of artefacts carrying normative power: governmental laws, regulations, rules, policies, norms, and so on; and each of them can have an impact on information systems. Out of this heterogeneity, it's of critical importance to understand the difference in the level of abstraction of legal prescriptions. This difference originates two fundamental types of legal prescriptions that ultimately distinguish the problem of law in requirements engineering from the problem of law in later phases of software development. This difference is effectively illustrated in (Logrippo, 2007), which speaks respectively of Hammurabi laws and Moses laws. The first class of laws - inspired by the biblical example of the Hammurabi code - states cause-effect rules, in a shape similar to Event-Condition-Action (ECA) rules. This type of laws is also called rule-level laws. The second class of laws - inspired by the biblical example of the Moses code - states desired states of affairs, without specifying how to reach them. This distinction is used to argue that Moses laws is mostly the type of legal prescriptions that has to be taken into consideration at requirements time (and for this reason they are also called requirements-level

laws): this is because their level of abstraction makes necessary to evaluate the requirements for their operationalisation into a concrete behaviour.

A less biblical characterisation distinguishes three level of legal prescriptions (Biagioli and Grossi, 2008): *Constitutive* laws provide a propositional definition of things being regulated. *Procedural* laws consist in collections of propositions in natural language that describe the sequences of actions forming a procedure. *Regulative* (or requirements-level) laws consist in collections of propositions in natural language that define the states of affairs (desired by the legislator). So we can represent a law as a set of propositions:

$$L = \{l_1, ..., l_n\} \tag{3.1}$$

We'll show in the next Chapter what's the structure of such propositions. Here we hypothesise that a regulative proposition of that kind simply contains the description of how the world should be. If the world is actually in one state described by the proposition, then we say that the proposition holds; otherwise, the proposition does not hold.

Propositions and states of the world Generalising, we seek for formalism to express the fact that a set of propositions in a law L entail a set of states of the world $W_L = W(L)$, so that W_L is a model for L:

$$W_L \models L \tag{3.2}$$

We have then that a state of the world w is legal it is in the set of possible worlds described by law L; i.e., if:

$$w \in W_L \tag{3.3}$$

For example, a privacy law $L = \{l_1\}$ may state that l_1 = "electronic transmission of health care data must be encrypted". Once a system has been built

and is running, according to its behaviour it can be in a finite set of states. Out of them, law makes an assertion about the correlation between qualities of the entities that form the system. Let suppose the entities are "Health care data", whose qualities can be "encrypted" and "not encrypted", and "Transmission", whose qualities can be "in progress" and "closed". The resulting states are
(w_1) "Transmission" = "closed", "Health care data" = "encrypted";
(w_2) "Transmission" = "in progress", "Health care data" = "encrypted";
(w_3) "Transmission" = "closed", "Health care data" = "not encrypted";
(w_4) "Transmission" = "in progress", "Health care data" = "not encrypted".
l_1 states that ("Transmission" = "in progress") $\rightarrow$ ("Health care data" = "encrypted"). In state w_1 and w_3 the pre-condition does not hold; in w_4 the post-condition does not hold; in w_2 both pre- and post-conditions hold. This is a pretty common approach in normative multi-agent field (see for example (Dastani et al., 2002)). We depart from such approaches in that we aim at representing the relation between the representation of a specific class of legal norms (laws) and the representation of the states of the world defined by system requirements. We model this relation by saying that $\{w_2\} \models L$ — i.e., by saying that the state w_2 is **law-compliant** with respect to the law L. The exact notion of pre/post-condition, their semantics in requirements engineering and the way to represent them in goal-oriented requirements engineering methodologies form the solution to the problem of law compliance of requirements. In the next section, we will try to characterise more such relation.

Law properties Before continuing, worth specifying two properties we assume in our representation of law, to better circumscribe the problem. As said, law is a collection of natural language sentences, which define a set of possible states of the world. The structure and semantics of the sentences influences the structure of the possible worlds. In particular, legal sentences may be (and they actually are) affected by inconsistencies of any type, due to imperfections in

their formulation. It may be the case that apparently both w and $\neg w$ hold in L. However, if this contradiction may exist in the source artefacts, in principle this is not admissible in a legal system, and should be solved upstream of the derivation of L. The assumption is then that contradictions don't exist in L:

$$(w \models L) \rightarrow \neg(\neg w \models L) \tag{3.4}$$

Law always addresses subjects, who ultimately are the only responsible for abiding the law. If the addressee can't be identified, such responsibility can't be identified and nobody must abide the law. For instance, if a law says that "killing is forbidden", it means that those *behaviours* are forbidden, which cause death. If killing is consequence of - for example - a natural disaster such a typhoon, then nobody is responsible for that — we can't say that the typhoon is responsible of murder.

From this consideration, it follows the fact that the states of the world described by law have to be matched with those that are *local* to subjects. Everything, that can't be associated to a subject, can be ignored. We express it by saying that, despite the words used in law sentences, the states of the words described by law only concern behaviours. A consequence of locality of law is that not every normative sentence is addressable to our purposes. For example, if a state of the world described by a proposition contained in the law can't be reached with human actions, then the proposition is not considered for compliance. Conversely, technical standards, which refer to properties of other artefacts, are not contained into L, but the *behaviour* of the subjects in charge adhering to the standard could be part of L . For example, if a propositions apparently does not describe a behaviour - for example, a technical norm prescribes that HTML documents must be enclosed within `<html>` and `</html>` tags - then we say that the regulated actor could the the webmaster in charge of developing a web site.

3.2 Requirements Compliance

In the previous section, we have provided our notions of law and law-compliance. Here, we will characterise the concept of *law-compliance of requirements*. Requirements come from stakeholders needs and objectives, and are synthesised, in goal-oriented approaches, as actors' goals — i.e., states of affairs desired by stakeholders. On the contrary, law describes states of affairs *desired by the subjects who create the law, but intended to be achieved by the subjects who are addressed by the law*. To do this, laws make prescriptions that have (a) different language, (b) different concepts, (c) different interests, (d) different scope than the stakeholders goals.

Laws can conflict with stakeholders goals and break their strategies. As a consequence, stakeholders face law prescriptions by trying to adapt their strategies and comply without compromising their objectives. The actual requirements come from this adaptation mechanism of objectives to laws.

The condition, in which a requirements set is not in contrast with the set of applicable laws, is defined as requirements compliance. Defining what "not in contrast" mean, implies being able to compare laws prescriptions and requirements. As said above, stakeholders' needs and laws are not comparable in terms of language, concepts, interests and scope. However, they have in common the purpose of describing desired states of affairs. Along this line it is possible to study the alignment of requirements with laws.

We hypothesize that both, law propositions and requirements statements, describe states of the world, but laws do it with underspecified information with respect to stakeholders. Specifically, laws don't provide information that are local to the domain. Requirements describe the set of states of the world characterised by the property of being *wanted* by stakeholders. Laws describe the set of states of the world characterised by the property of being *legal*. The overall set of possible world is then given by the union of these two sets of states:

$$W = W_R \cup W_L \tag{3.5}$$

We derive a general rule to define the notion of compliance for requirements. We say that, given a set of requirements R, and a set of domain assumptions K, the requirements are compliant with a law L if, for every possible state of the world, if R holds, then L holds.

$$R, K \models L \tag{3.6}$$

A state of the world entails the requirements if it belongs to the states $W_R = W(R)$. It is law compliant if it belongs to the states $W_L = W(L)$. We say that a set of requirements R is compliant with respect to a set of laws L if and only if, for every possible state of the world w that belongs to R, w also belongs to L. In this case, we say that the entailment relation $R, K \models L$ holds:

$$R, K \models L \text{ iff } \forall w \in W, (w \in W_R) \rightarrow (w \in W_L) \tag{3.7}$$

The property $(w \in R)$ refers to the strategic value of a state, and it is fully verifiable: since the information concerning R comes form the stakeholders, strategic acceptability is part of the gathered knowledge. The property $(w \in W_L)$ refers to the legal acceptability of a state - i.e., to its *compliance* - and verifying if it holds is the topic elaborated here.

3.3 Compliance Properties

In common sense, being compliant with law means behaving how law prescribes. Without formally characterising it, we call this intuitive meaning of compliance **actual compliance** because it means being actually compliant from a theoretically correct point of view, and it's therefore is the only meaning that ultimately is relevant. This meaning has the drawback that it can only be realised and verified ad run-time. For this reason, we have introduced the notion

of **intentional compliance**, defined as the design-time distribution of responsibilities such that, if every actor fulfils its goals, then actual compliance is ensured (Siena et al., 2008b). In a perfect world - in an economic sense, a world in which complete information and perfect rationality exist - intentional compliance could be a sufficient answer to the problem of compliance, since it would be possible (for everybody) to fully evaluate at run-time the compliance of the running processes, the behaviour of the system and that of people. But in our imperfect world, being actually compliant is not enough. Said with an example, if a murder occurs, if somebody can't prove his own innocence, then he can be declared guilty, even if he is actually not guilty; similarly, if somebody is guilty but can somehow prove his own innocence, then he can be declared innocent. In other words, it becomes necessary to *prove* that the due has be done. Finally, proving that a compliant process has been executed falls back in proving two different things: first, that the process has been executed in a certain way; and second, that the way it has been executed was the compliant one.

Ultimately, we derive a set of properties that a framework must be able to express, in order to be a valid framework for law-compliance of requirements: intentionality, ability, auditability, traceability.

Intentionality A compliance framework must be able to engineer a requirements set such that the system, once designed, developed and deployed correctly, will behave within the states admitted by legal sentences. As described in previous section, Specifically, we want to link to legal prescriptions the *purposes* for the system-to-be, rather than just its operations. This will allow that, one the purpose if aligned with legal prescriptions, the specification will be compliant by incremental construction.

Ability A key element that distinguishes laws from goals (as will be detailed in the next chapter), is that laws are prescribed to *classes of subjects*, while goals

are *specific* to a certain actor. In other words, laws disregard what actors can or want to do. However, an important element that has to be taken into consideration while gathering, validating and specifying requirements are actors' attitudes. For example, in (Yu, 1996) are described the concepts of *ability* and *workability*. In general, any compliance solution is acceptable only of the actor, who has to be compliant, has the capability to perform the necessary actions - either directly, or by delegating them - to achieve the desired objectives.

Auditability Only the running system is actually compliant or not compliant. And true compliance can only be established *ex post* by the judge. Formal proof of run-time compliance can't be given at requirements time: there are properties of law that makes that the compliance condition can only be stated *ex-post* by the judge - e.g., the subsequent design could be wrong, people could behave differently from what is assigned to them according to their roles, software programs could be bugged and also behave differently from what expected, as well as the intentional ambiguity. Auditability refers to the capability to provide formal evidence (proof) that a certain behaviour has been operated in alignment with a normative proposition. The concept of auditability is not intrinsic to the nature of law, although some laws may prescribe to keep some records from the running processes. Rather, it is mostly an industrial need (Cederquist et al., 2007). We add auditability as one of the requisites for a requirements compliance framework, because we look at the problem of compliance from the standpoint of stakeholders strategy.

Traceability Compliance choices have relevance in two different moments. The first moment is when the choices are actually made, in order to align requirements with law. The second moment is subsequent to the requirements elicitation, and it is when the motivation of the requirements is needed. For example, when the requirements are going to change it is necessary to know where did

the requirements come from, if they exist because of a strategic choice or due to a legal necessity. Also, when law changes, it raises the need for knowing which requirements are affected. This traceability information is also the first element of provability of compliance choices.

Chapter 4

The *Nòmos* Framework

The requirements analysis activity has the purpose of fostering a correct comprehension of stakeholders' needs, and to ensure that requirements, once specified, retain some desired properties. Goal-oriented requirements (Mylopoulos et al., 1999) engineering rests on the idea of deriving the requirements of a software system from the analysis of the goals that the system-to-be will need to achieve once developed and deployed (van Lamsweerde, 2001; Jureta et al., 2008). Several works, exploiting currently established goal-oriented modelling and analysis methods, provide evidence of the effectiveness of goal-oriented requirements engineering (van Lamsweerde and Letier, 2000; Fuxman et al., 2004).

However, no evidence is given that the compliance relation between requirements and laws, as specified in Equation (3.6), holds. For this reason, we propose the *Nòmos*[1] framework, which aims at giving a conceptual and methodological solution to the requirements compliance problem. The framework is comprised basically by 3 elements: a language for modelling requirements and the impact of legal prescriptions on requirements; an analysis guidelines, to check compliance as in Equation (3.6); and a process for systematically generating law-compliant requirements. This chapter covers the first element; the next chapters will discuss the analysis and the process.

[1] From greek $N\acute{o}\mu o\varsigma$, which means "norm".

Table 4.1: The Hohfeldian taxonomy of legal rights.

Legal relation	Opposite	Correlative
Claim	Noclaim	Duty
Privilege	Duty	Noclaim
Power	Disability	Liability
Immunity	Liability	Disability

4.1 The Language of Law

In a general sense, addressing the issue of law-compliance of software requirements means being able to understand the actual prescriptions of legal statements, and how they are reflected on stakeholders goals. To do this we firstly want to understand the language and the concepts that form legal prescriptions, in order to be able to build models of the prescriptions.

Normative propositions The normative semantics expressed by laws is carried by utterances called **normative propositions** (Alchourron and Bulygin, 1971). As in (Sartor, 2009), we will not make explicit distinction between the normative proposition, which is a meta-level assertion concerning what can or cannot be deduced from a given set of law fragments, and the law fragment itself as a norm. In other words, we consider every normative proposition as a proposition carrying the normative power it pretends to have, and don't investigate if and why it actually has such power or not. Additionally, this notion of normative proposition detaches the problem of normativity, per se, from the problem of normative texts. The written source of law are ideally structured in a hierarchical way, comprised by chapters, titles, paragraphs, and so on; however, they also make heavy use of cross-referencing, and stratification - i.e., succession of additions, deletions, modifications - occurs over time. Tool-suppoted approaches (Kiyavitskaya et al., 2008; Breaux et al., 2008) are being developed to cope with this problem. We move from where these methods end — i.e., from a

flattened view on legal documents, in which cross-referencing and stratification has been solved, so that we can see law as a set of well-formed sentences.

Normative propositions are sentences that charge somebody of a certain legal modality with respect to an object: `<NP> = <subject>, <object>, <LM>`. The physical or artificial person, who is addressed by the normative proposition is referred to as the **subject**. The object of the proposition concerns, directly or indirectly, a behaviour of the subject. The legal modality is what actually carries the normative aspect concerning the object; i.e., it is what actually creates a command or a prohibition.

Legal modality Several approaches have been proposed in the past to capture the legal language. Deontic logic is the most known formalism, used over the decades pursue objectives of automated legal reasoning, AI, multi-agent system organisation and regulation and so on.

For our purposes, we have adopted as a background the Hohfeld's fundamental legal taxonomy (Hohfeld, 1913). The Hohfeldian taxonomy is a milestone of juridical literature that proposes a widely accepted classification of legal concepts. The choice to adopt the Hohfeldian taxonomy of rights is due to several factors. First, as said above, the importance it has in the juridical literature suggests that this - not others - is actually the kind of information that we need to know about law. Second, it's range of concepts, and their level of abstraction, make their representation capabilities very close to the expressiveness of legal texts. This consideration comes mainly from experience: constructs like powers, immunities and so on do actually exist in legal texts, and the proposed taxonomy is able to successfully capture them, differently for example from deontic logic-based approaches. Finally, the Hohfeldian concepts that have a *descriptive* nature, rather than *prescriptive*, acting as the bridge between the world of the "ought", typical of legal prescriptions, and the world of domain description. Said differently, the Hohfeldian concepts do not prescribe what

stakeholders should do - rather, they describe what are the legal relations that bind them. This is of particular importance in requirements engineering, whose first step (early requirements analysis) is to describe the so-called "as-is", before specifying the "to-be". So, linking a description of stakeholders goals with a description of applicable laws may allow to reason about compliance and compliant alternatives. In the following, we try to formally characterise such a link.

Hohfeld's taxonomy is grounded on the concept of ***right***, which can be defined as "entitlement (not) to perform certain actions or be in certain states, or entitlement that others (not) perform certain actions or be in certain states"[2]. Rights are classified by Hohfeld in the 8 elementary concepts depicted in Table 4.1: *privilege*, *claim*, *power*, *immunity*, *no-claim*, *duty*, *liability*, *disability*, and organised in opposites and correlatives, as in Figure 4.1. Two rights are **correlatives** (Hohfeld, 1913) if the right of a person A implies that there exists another person B (A's counter-party), who has the correlative right. Vice versa, the concept of *opposition* means that the existence of a right excludes its opposite for the same action. The concept of *correlativeness* is particularly important because it implies that rights have a **relational nature**. In fact, they involve two subjects: the owner of the right and the one, against whom the right is held - the *counterparty*.

The Hohfeldian taxonomy As said, Hohfeld's taxonomy is based on the concept of right. In commonsense it might be counterintuitively to call *right* a duty or a liability, since the word has a different meaning, that is "having the entitlement to legally pretend". However, he points out that:

> *Latin "Ius", the German "Recht", the Italian "Diritto", and the French "Droit" [are] terms used to express not only a right, but also Law in the abstract.* (Hohfeld, 1913)

[2]From `http://plato.stanford.edu/entries/rights/`

So the word "right" has two meanings: one weaker, which refers to the whole class of legal entities; and one stronger, which refers to a particular type of right, he calls *Claim* to disambiguate.

Claim and Duty **Claim** is the entitlement for a person to have something done from another person, who has therefore a **Duty** of doing it; for example, if John has the claim to exclusively use of his land, others have a corresponding duty of non-interference.

> *[...] if X has a right against Y that he shall stay off the former's land, the correlative (and equivalent) is that Y is under a duty toward X to stay off the place.* (Hohfeld, 1913)

Privilege and No-claim **Privilege** (or liberty) is the entitlement for a person to discretionally perform an action, regardless of the will of others who may not claim him to perform that action, and have therefore a **No-claim**; for example, giving a tip at the restaurant is a liberty, and the waiter can't claim it.

> *"If the power of the State will protect him in so carrying out his wishes, and will compel such acts or forbearances on the part of other people as may be necessary in order that his wishes may be so carried out, then he has a legal right so to carry out his wishes." The first part of this passage suggests privileges, the middle part rights (or claims), and the last part privileges.* (Hohfeld, 1913)

Power and Liability **Power** is the (legal) capability to produce changes in the legal system towards another subject, who has the corresponding **Liability**; examples of legal powers include the power to contract and the power to marry.

> *X, the owner of ordinary personal property in a tangible object has the power to extinguish his own legal interest (rights, powers, immu-*

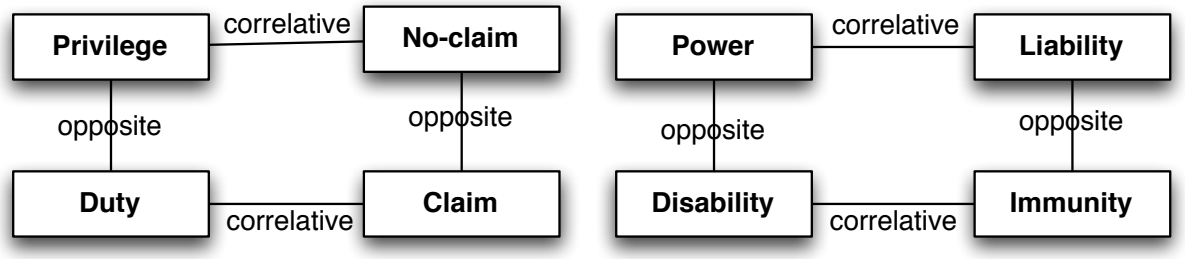


Figure 4.1: The fundamental hohfeldian legal rights.

> *nities, etc.) through that totality of operative facts known as abandonment; and-simultaneously and correlatively-to create in other persons privileges and powers relating to the abandoned object,-e. g., the power to acquire title to the later by appropriating it.* (Hohfeld, 1913)

Immunity and Disability **Immunity** is the right of being kept untouched from other performing an action, who has therefore a **Disability**; for example, one may be immune from prosecution as a result of signing a contract.

> *If, indeed, a sheriff has been duly empowered by a writ of execution to sell X's interest, that is a very different matter: correlative to such sheriff's power would be the liability of X,-the very opposite of immunity (or exemption). It is elementary, too, that as against the sheriff, X might be immune or exempt in relation to certain parcels of property, and be liable as to others. Similarly, if an agent has been duly appointed by X to sell a given piece of property, then, as to the latter, X has, in relation to such agent, a liability rather than an immunity.* (Hohfeld, 1913)

The above taxonomy of rights offers concepts that have a *descriptive* nature, rather than *prescriptive*. As such, the taxonomy acts as the bridge between the world of the "ought", typical of legal prescriptions, and the world domain

description. In the following, we try to formally characterise such a link, and explain how this allows for reasoning about compliance and compliant alternatives.

Subject The subject in a normative proposition (or of a set of them) consists in the description, in natural language, of the characteristics of the actors - a physical or artificial person - who is in charge of the need to comply with the normative proposition. The subject described by a normative proposition does not explicitly identify the physical or artificial person; using the agent metaphor, the law does not explicitly identify neither an agent nor a role. Law has no knowledge of the agents or roles that exist in a certain domain at a certain time, so it makes an abstraction and just gives the information that allows for identifying the actual agent in every specific context.

Action The object of normative propositions concern, directly or indirectly, *actions* performed by subjects (Roversi, 2005). Actions are composed by two parts: the *behaviour* of the actor and the *consequences* of the behaviour. Actions specified by normative propositions can therefore be either *behavioural actions* or *productive actions*.
A **behavioural action** consists in the specification of the behaviour to be performed by the actor, regardless of the results it produces.
A **productive action** consists in the specification of the result of a behaviour, regardless of the actual behaviour for producing the result.
The distinction between the two types is fuzzy, since the concept of causation may lead to important issues. For example:

> *A person died of thirst in the desert, after that (a) one killer put poison in the victim's bottle to poison him, and (b) another killer independently made a hole in the bottle to let the victim die of thirst. In such a case, we may wonder who brought it about that the victim died:*

> *the first killer, who was prevented from poisoning the victim, or the second, who saved the victim from being poisoned, and so postponed the victim's death.* (Roversi, 2005)

And, from another standpoint:

> *We may wonder whether any type of causation by an agent's bodily movement (for example, a movement done while sleeping [...]) can properly be said to realise an action.* (Roversi, 2005)

Despite this terminological ambiguity, when referring to the action specified by a normative proposition we will always mean a productive action.

4.2 Formalisation

In this section we outline a formal characterisation of the basic legal concepts introduced above, borrowing definitions proposed by Sartor in (Sartor, 2006). The purpose is to describe the essence of the adopted concepts in terms of their *deontic* meaning, i.e. binding their meaning to the notion of "ought" that laws contain.

The object of a right is an "action" (Sartor, 2006), i.e., an abstract characterisation of a behaviour. Two types of actions exist: *behavioural actions* are described by the actual behaviour performed by actors as "j does A"; *productive actions* are described by means of the results produced by the behaviour of the actors: "A brings it about that x" (Sartor, 2006). Notice that actions can be *positive* (do something) or negative (omit something), but we will always represent them as positive; for example, both "don't smoke" and "pay taxes" are considered actions to be performed by actors, regardless to the fact that the first one is expressed with a negative proposition.

So, both types of actions are represented using the notation

$$Does_j A$$

which means that the action A is ascribed to actor j. For example:

$Does_{John}$[Smoke]

John is smoking

$Does_{John}$[Get a degree]

John brings it about that he gets a degree

In deontic logic (von Wright, 1951), whenever an actor j is obligated to do something (A), then j has an *Obligation* to do A, and it is written as $Obl^j(A)$. If j is forbidden to do something, he has a *Prohibition*, which is represented by the predicate $Forb^j(A)$, and can be written in terms of obligation as $Forb^j(A) = Obl^j(\neg A)$. When j is permitted to do something, then he has a *Permission* (predicate $Perm^j(A)$) and can be also expressed in terms of obligation as $Perm^j(A) = \neg Obl^j(A)$. The semantics of hohfeldian rights can be expressed in terms of the above basic notions as follows.

Privilege - Noclaim. As first, we characterise the notion of privilege, which intuitively means that the rightholder (actor j) is free whether to do or not to do something (action A), with respect to any other (actor k). We can formally write it as: **Privilege**$^k Does_j A$, where $Does_j A$ is a description of the behaviour of actor j. Notice that $Does_j A$ implies that actor j has the *ability* to perform A. This notion of privilege is equivalent to say that the actor j is permitted by any actor k to not do action A, that is we can find an equivalent formalisation of the notion of privilege in terms of the deontic operators introduced by Sartor (Sartor, 2006), as follows:

$$\textbf{Privilege}^k Does_j A \equiv \textbf{Perm}^k(\mathsf{NON}\ Does_j A) \tag{4.1}$$

Consistently, a no-claim is the missing prerogative of somebody (actor k) to have something done by another (actor j) and can be written using deontic

operators through the following equivalence:

$$\textbf{NoClaim}^k Does_j A \equiv \textbf{Perm}^k(\mathsf{NON}\ Does_j A) \tag{4.2}$$

which reads as: the fact that actor k can not claim that actor j performs action A is equivalent to say that actor k permits actor j to not do action A. So it is possible to characterise the hohfeldian *correlativeness* relation along the following equivalence:

$$\textbf{Privilege}^k Does_j A \equiv \textbf{NoClaim}^k Does_j A \tag{4.3}$$

The above formula is a *normative proposition* (NP), and we represent it by means of the predicate:

$$NP = \textbf{privilegeNoclaim(j, k, A)} \tag{4.4}$$

For example, if j is "hospital", k is "patient", and A is characterised as "Disclose PHI", then the above formula means that the hospital is free (or not) to disclose to the patient its PHI.

Claim - Duty. Similarly to the equivalence between *privilege* and *noclaim*, it is possible to formalise the other correlative rights. So, the fact that an actor (j) has a duty towards another actor (k) to perform an action (A) is expressed, according to (Sartor, 2006), by the formula:

$$\textbf{Claim}_k(Does_j A) \equiv \textbf{Obl}^k(Does_j A) \tag{4.5}$$

where the $Claim$ side is a formal representation of Claim, and the Obl side is the formal representation of Duty. This is equivalent to say that actor k has the *obligative right* towards j, to have action A done.

We represent this equivalence with the predicate:

$$NP = \textbf{claimDuty(k, j, A)} \tag{4.6}$$

Power - Liability. Generally speaking, power is the right to produce legal effects, which means changing the *normative position* of somebody else. The

normative position P of an actor k, written (P^k), is the set of legal entitlements of k. For example, voting, the entitlement to marry or to own a driving license, the enrolment as university professor, but also a governmental license to trade cigarettes or to manage a casino - all of these are examples of legal position of a given actor. Normative positions have a legal existence, and as such they can be created or deleted by means of legal actions. For our purposes, we consider a particular notion of power - the *enabling power* - which is used to complement legal claims by acting as the means for the claim's accomplishment to be ensured. Powers are conferred to an actor to further its interests. If A is the action that has legal effect, and B is the interest to be protected, then we write B VIA A to mean that, doing A has the purpose of achieving B. The enabling power results:

$$\textbf{EnablingPower}_j P^k(B \text{ VIA } A) \equiv (\text{IF } Does_j A \text{ THEN } B) \tag{4.7}$$

If B is something that the actor is legally entitled to pretend (a claim), then a law can define a legal action that the actor can undertake to enforce B, if it is not fulfilled. We call that action a *sanction*, S. This complements a claim/duty in that it allows to activate the sanctioning mechanism in the case B is not fulfilled. We have then:

$$\textbf{EnablingPower}_j P^k(S) \equiv \textbf{LiabilityRight}_k(Does_j S) \tag{4.8}$$

that means that, whenever an actor has a claim, then it should also have the power to pretend the accomplishment of the claim in case the obligated actor does not fulfill its duty. Notice that this is a strong assumption and in laws it's possible to find claims without connected power; we admit this possibility, but it will not be explained here, as it is a simpler case. Similarly, liabilities can further be connected to other rights, but for sake of simplicity it will not be explained here.

We represent the above formula by means of the predicate:

$$NP = \textbf{powerLiability(j, k, S)}$$

Immunity - Disability. Finally, an immunity is a defence against the power of others and consists in the right of being kept unchanged from other's power, by means of doing something. We represent this equivalence with the predicate:

$$NP = \textbf{immunityDisability(k, j, S)}$$

As depicted in Figure 4.2(a), the described formalism allows to map the first Hohfeldian square of reights into deontic primitives. Similarly, due to Equation (4.7), the second Hohfeldian square can be mapped too, as depicted in Figure 4.2(b). In the following, we will rely on this semantics when defining the primitives of the *Nòmos* language.

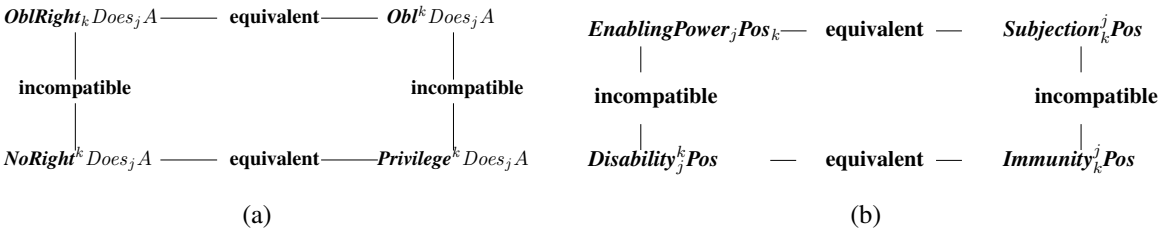


Figure 4.2: The deontic representation of the Hohfeldian squares.

4.3 The *Nòmos* modelling language

The purpose of *Nòmos* is to link requirements models with law models, allowing to argue about compliance of the first with the latter. To do this, a language is required, to model requirements and legal prescriptions.

To model requirements, *Nòmos* adopts the *i** modelling framework (Yu, 1996). Worth mentioning that this choice is not the only possible. Other frameworks could be used or adapted to be used as well, as long as they provide primitives for modelling actors, goals, and relationships between actors. The

*i** framework (Yu, 1996) models a domain along the two following perspectives: the **strategic rationale** of the actors - i.e., a description of the intentional behaviour of domain stakeholders in terms of their goals, tasks, preferences and quality aspects (represented as softgoals); and the **strategic dependencies** among actors - i.e., the system-wide strategic model based on the relationship between the depender, which is the actor, in a given organisational setting, who "wants" something and the dependee, that is the actor who has the ability to do something that contributes to the achievement of the depender's original goals. The system-to-be is modelled as a new actor of the organisational setting that enables the achievement of domain stakeholders goals, so expressing the requirements of the system. For a detailed description of the *i** framework, see (Yu, 1996). Here, we recall the important concept of actors ***ability*** to pursue a goal. An actor has the ability to achieve a goal G if (i) the actor has in the set of intentional elements that characterise it (such as sub-goals, tasks and resources) an element or a set of elements whose purpose is the achievement of G; or, (ii) if the actor can delegate the achievement of the goal to another actor - the dependee.

Upon the *i** language, *Nòmos* introduces a notation to model normative proposition. The combination of *i** and the extension for law modelling constitutes the *Nòmos* modelling language.

4.3.1 Meta-model

The *Nòmos* modelling language relies on a meta-model (Siena et al., 2009c), which has the twofold objective of providing an abstract characterisation for the language, and integrating its modelling capabilities with those of the *i** meta-model, as proposed for example in the variant used by the Tropos methodology (Susi et al., 2005). The *Nòmos* meta-model is depicted in the diagram of Figure 4.5. The dashed line represents a part of the *i** meta-model, limitedly to the `Actor` class and its `wants` association with goals. The dotted line is

a UML representation of the predicate described in previous section as a **normative proposition** - i.e., the descriptions of the right between two subjects, and the action that the right refers to. A normative proposition (NP) It is the lowest level of granularity in which it's possible to split a law statement. However, the structure of normative propositions not necessarily matches the textual description of legal sentences, as discussed above.

Right Since rights have a dual nature, the relation of "correlative" or "equivalent" means that the two rights that it connects - for example, *duty* and *claim* - are interchangeable, because they describe the same reality from two different points of view. So for example, instead of speaking about "duty" and its correlative "claim", we speak about a unique class, the $ClaimDuty$. In Figure 4.5 we have summarised the 4 resulting classes of duties, namely `PrivilegeNoclaim`, `ClaimDuty`, `PowerLiability and ImmunityDisability`. The classes are grouped as sub-classes of the abstract class `Right`.

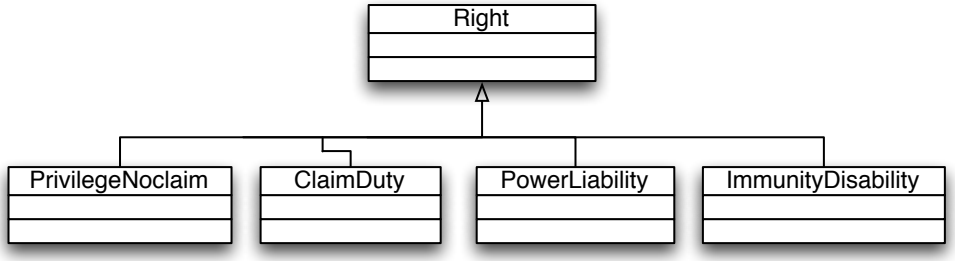


Figure 4.3: The hierarchy of rights.

Counter-parties The addressee of the legal prescription is the *Subject*. In our framework we identify the concept of subject with the concept of `Actor`. Because of the concept of correlativeness, both the right holder and its correlative are actors. If a person has a right, then another person has its complement. This is shown in Figure 4.5, where the class `Actor`, representing the subject, and the class `Right` are introduced and connected by the two relationships `holder`

and `counterparty`. When a right is created, it connects the two `Actors`. Any `Actor` can be owner of an unlimited number of rights, and a right has exactly two `Actors`.

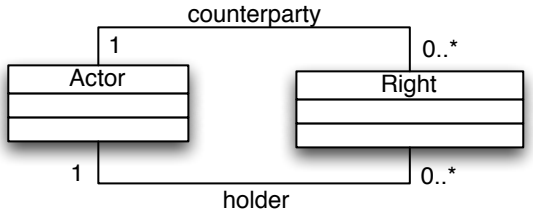


Figure 4.4: A right connects two parties: the holder and its counter-party.

Rights are not symmetric: the position of the *claim* owner is different from the position of the *duty* owner. Generally speaking, the two positions are called active (juridical) position and passive (juridical) position, and each right has exactly an active position and a passive position. To capture this characteristic in the metamodel, active `Actors` are in `holder` relation, while passive `Actors` are in `counterparty` relation with respect to the right, as shown in Figure 4.5.

Action The last component of a prescription is the **concerned action**. Here we use the word *action* in the sense of (Sartor, 2006) - i.e., a concept that groups *behavioural actions* and *productive actions* (not shown in the meta-model). To avoid confusion with a more common use of the word, we refer to it as `PrescribedAction`. Each `Right` is in `concerns` relations with exactly one `PrescribedAction`, but an `PrescribedAction` can be addressed by a number of rights, as depicted in Figure 4.5.

Dominance The concept of normative proposition allows to split the complexity of legal statements into their atomic elements. But the legal prescriptions contained in laws have more properties that have to be considered. In particular, legal prescriptions are articulated structures built with conditions, exceptions,

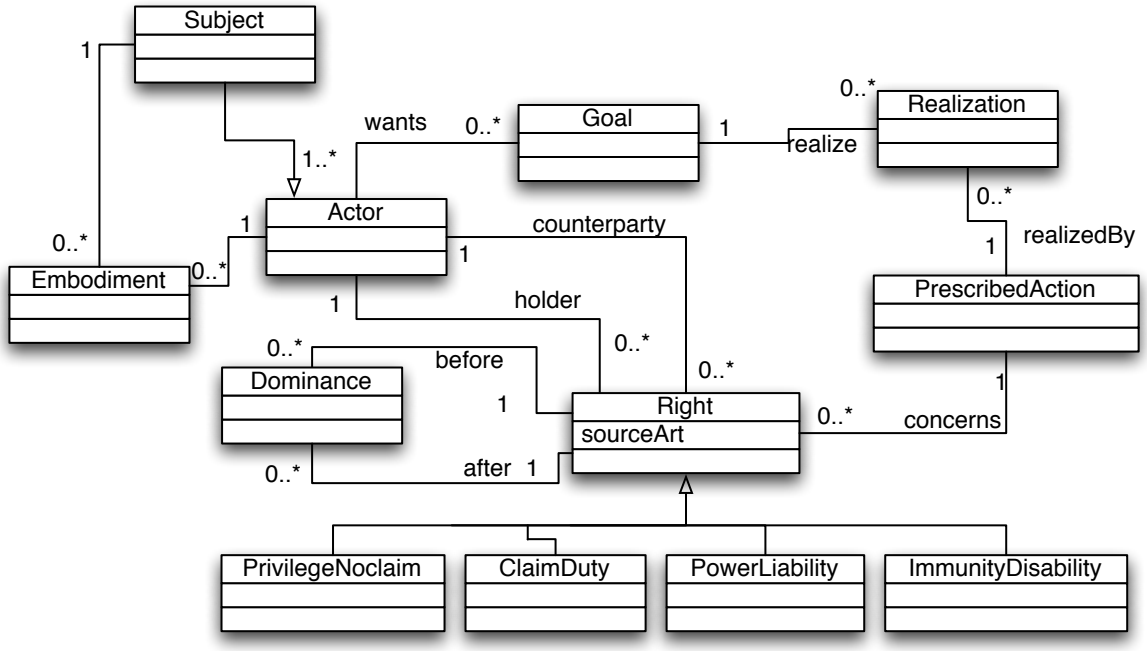


Figure 4.5: The meta-model of the *Nòmos* language.

and so on. It is important to capture the effects of these conditionals in order to obtain a meaningful requirements set. We give a uniform representation of conditional elements by establishing an order between normative propositions. Given a set of normative propositions $\{NP_1 ... NP_n\}$, $NP_k > NP_{k+1}$ - read: NP_k overcomes NP_{k+1} - means that if NP_k is satisfied, then the fulfilment of NP_{k+1} is not relevant. For example, a citizen has the duty to give his personal details to the policemen; but if the policeman does not identify itself correctly as a policeman, then the citizen whether to do it or not. In this case, we say that the privilege of giving personal data to a not properly identified person dominates the duty. This is captured in the metamodel via the definition of the concept of **dominance** (class `Dominance`), connected to the class `Right`.

4.3.2 Representing Compliance

Embodiment Subject addressed by law are abstract conceptions. In other words, in real legislation systems can't be considered as valid propositions such as

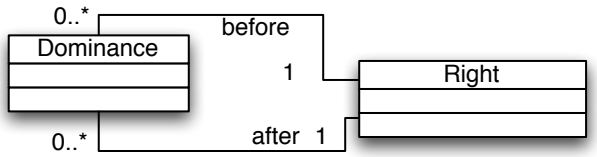


Figure 4.6: Rights are put in dominance relation with each other.

Claim$_{Mary}(Does_{John}$[Pay 100€]), because, as said before, law can't explicitly address a specific agent (here, Mary and John); rather, it has to indicate in an abstract way the characteristics that let identify the concrete agents.

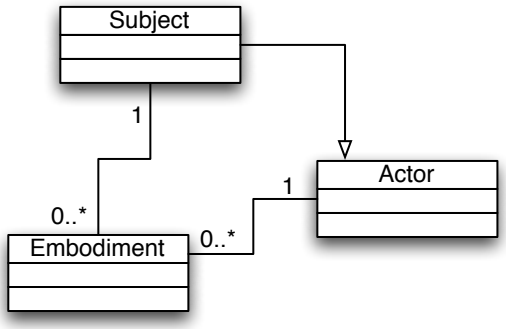


Figure 4.7: Every actor can embody a legal subject, including other legal subjects.

In the *Nòmos* meta-model, this is expressed by introducing two new classes, as in Figure 4.7. The class `Subject` is a generalisation of the `Actor` class. Semantically, a `Subject` is a rigid classification as in (Guizzardi et al., 2004): rigid means that every instance of a type is necessarily (in a modal sense) an instance of that type — i.e., if the type don't exist the instance can't cease to be an instance. For example, if a law addresses "all Italian citizens", every Italian citizen is automatically a subject of such law, unless he changes citizenship. This is different from the relation between `Agent` and `Role`, where the `is-a` relation between an agent and a role means that the behaviour of the role is intentionally assigned to the agent.

Realisation Rights concern actions, which can be behavioural or productive. Intuitively, the characterisation of these actions is a description of a behaviour of the addressee actor and can result in a goal or task of the actor. However, an action characterisation, *per se*, is not a goal neither a task, because a goal needs to be wanted and a task needs to be performed by an actor, and an action characterisation misses this. It's necessary to separate the concept of goal from the one of action characterisation to avoid misleading shortcuts. However, we need to be aware when a goal or task fit the characterisation given by law. In Figure 4.5, this is expressed with the concept of **realisation** (class `Realization`), which puts in relation something that belongs to the law with something that belongs to the intentions of actors. In the next section, we show how we use the `Realization` concept to propose a solution to the problem of requirements compliance.

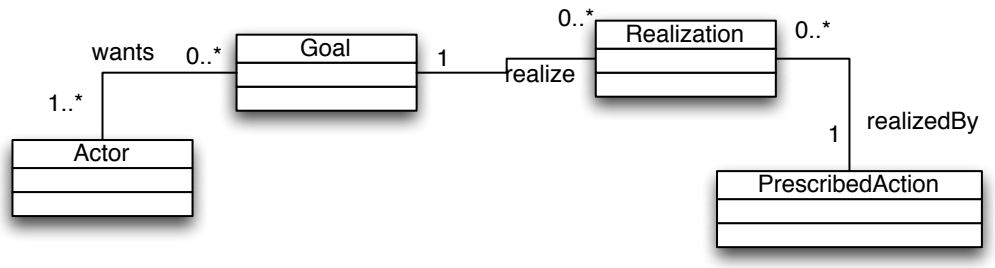


Figure 4.8: Actors' goals can realise prescriptions.

4.4 Visual notation

In order to support the requirements analyst in dealing with legal prescriptions across the subsequent phases of the requirements elicitation process, normative propositions are represented in the *Nòmos* frameworks by means of a visual notation. The notation of the *Nòmos* language, depicted in Figure 4.18, has been defined as an extension of the *i** visual notation. In fact, to argue about intentional compliance of requirements, it is necessary the language supports both

the representation of the intentional dimension of the domain - i.e., stakeholders with their goals - and of the legal dimension - i.e., the legal prescriptions.

In summary, the actors linked by a right (holder and counterparty) are modelled as circles (i.e., *i** actors). The specified action is represented as a triangle and linked with both the actors. The kind of right (privilege/noclaim, claim/duty, power/liability, immunity/disability) is distinguished via labels on both the edges of the right relationships. Optionally, it's also possible to annotate with the same labels on the left side the triangle representing the action. The language also introduces a *dominance* relationship between specified actions, represented as a link between two prescribed actions and labelled with a "$>$" symbol that goes from the dominant action to the dominated one. Finally, a *realisation* relation is used in the language to establish a relation between one element of the intentional model and one element of the legal model. In the following, we will detail the use of the language.

Legal relations Figure 4.9 shows a situation that is generated by laws on private transportation. There are subjects (the actor [Sender] in the picture) shipping goods, and subjects (actor [Carrier]) who physically do the transportation and deliver the goods. Law says that the action, which is the object of the transportation activity, i.e. [Goods be delivered], is a right for the sender. Said differently, the sender can pretend that action to be done by the carrier, and legally obtain it from a judge, if the carrier does not accomplish it. The carrier is therefore addressed by the corresponding duty.

Recalling the formalisation in previous section, the diagram in the picture is equivalent to say that the sender has a claim that the sender deliver goods:

$$\mathbf{Claim}_{\text{Sender}}(Does_{\text{Carrier}}[\text{Goods be delivered}])$$

or, equivalently, that there is an obligation for the carrier towards the sender to

deliver goods:

$$\mathbf{Obl}^{\text{Sender}}(Does_{\text{Carrier}}[\text{Goods be delivered}])$$

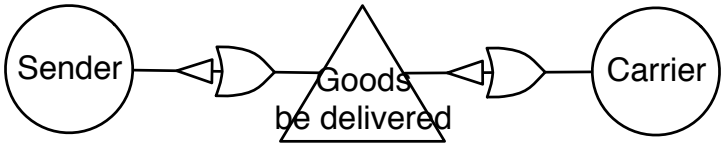


Figure 4.9: The claim/duty legal relation.

Figure 4.9 depicts another situation taken from the same law on transportation. Here, the carrier has a privilege: he can subcontract the execution of the delivery to somebody else. In this case, the sender can't do anything to prevent this to happen: he has a no-claim.

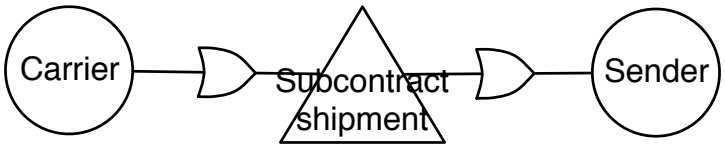


Figure 4.10: The privilege/noclaim legal relation.

In Figure 4.11 is depicted the power of the sender to withdraw the mandate. The nature of the power is apparently similar to the one of the claim, since in both cases the holder may arbitrarily use the right to produce an effect on the counter-party. The most visible difference is that in this case, the counter-party does not have to do anything: the use of the power automatically produces the effect on the counter-party, who is consequently owner of a liability. Worth noticing that this automatism is due to the fact that the power does not addresses a concrete action, but produces only a legal modification on another right.

Figure 4.12 shows an immunity of the carrier against the sender. An immunity is basically a means of self-defense from a power of somebody else. For this reason, it will always appear in relation of such a right. In the picture, the

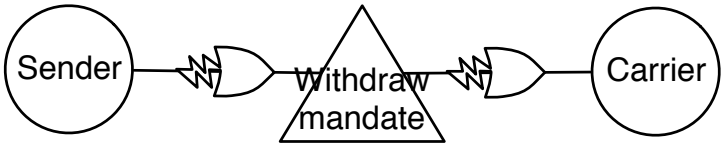


Figure 4.11: The power/liability legal relation.

carrier has the immunity from the sender's power to withdraw the mandate, if the carrier is able to demonstrate the on-time delivery.

Figure 4.13 summarises the result of rights modelling and how rights form legal relations between actors. However, the immunity case let clearly understand that legal rights rarely can be treated apart of each other. In order to put in relation legal relations, dominance relations are used.

Dominance Figure 4.14 depicts a dominance relation between rights. The pictures shows that, *for the Carrier actor*, the privilege to subcontract the shipment dominates the duty to delivery the goods. In other words, the relation shows graphically the priority between rights. The relation arrow links the two symbols of `PrescribedAction`, and this way it establishes the relation between the relative rights and, ultimately, to the normative propositions. By using the dominance relation, it is also possible to establish links between rights that come form normative propositions contained in different laws. This way, the rights that address a certain actor are abstracted from the structure of the legal documents. Worth noticing that a simple diagram like this is enough to generate a legal alternative. In fact, since the dominant right is a privilege, it ends up that the Carrier has two alternatives: either it delivers goods, or it uses its right and

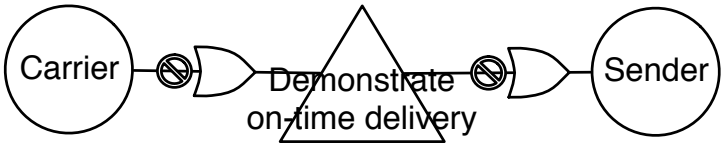


Figure 4.12: The immunity/disability legal relation.

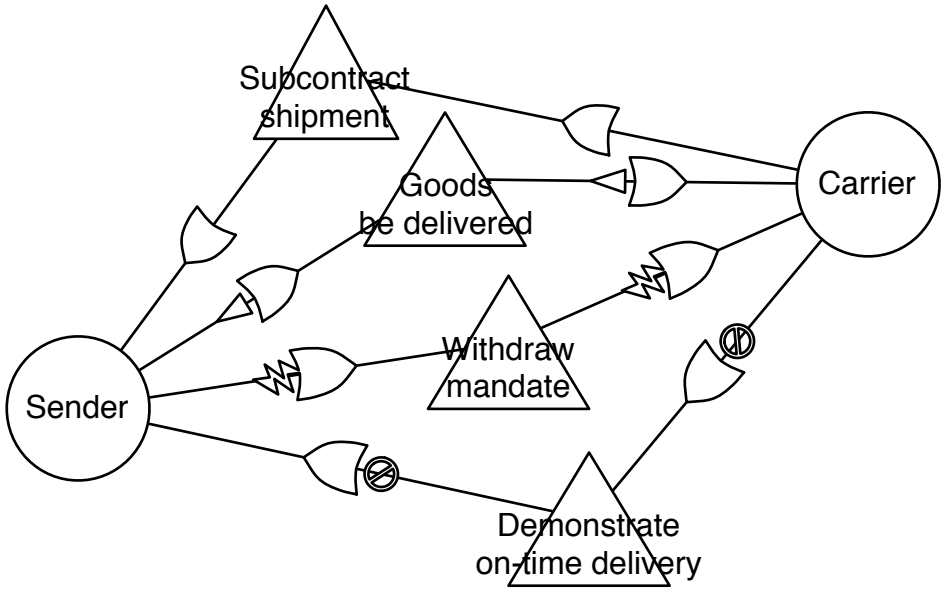


Figure 4.13: In this example, it is shown how legal relations contribute in shaping the social and organisational settings.

subcontracts the delivery.

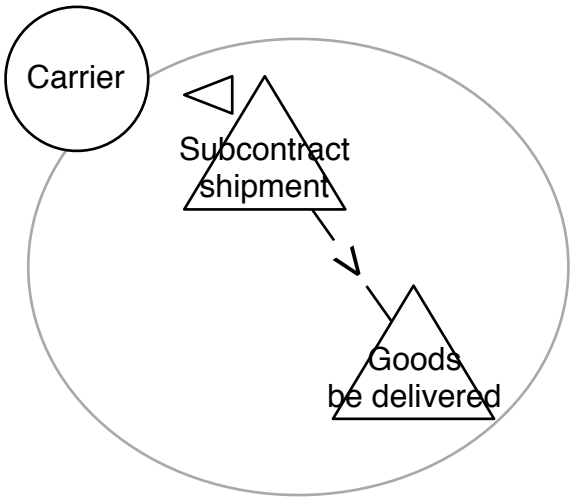


Figure 4.14: A dominance relation.

Embodiment Figure 4.15 depicts an embodiment relation between an actor of the domain and a legal subject. The picture shows that the [Carrier Company] is the subject referred to as [Carrier] in the law. [Carrier Company] is not a concrete

agent: [T.C. Inc.] is actually the agent that actually plays the role of a carrier company. Worth saying that, when not strictly necessary, in the following when an embodiment relation binds an actor of the domain and a legal subject, we will model only the domain actor, assigning to that actor all the right belonging to the corresponding legal subject.

Realisation Finally, a realisation between a goal and a right is depicted in Figure 4.16 as an arrow that links the goal with the right. The picture shows that, by achieving the goal [Delivery goods] goal, the [Carrier Company] is able to fulfil the duty [Goods be delivered].

In summary The usage of the language is depicted in Figure 4.17. The diagram in the picture shows the whole law that regulates the shipment of goods. The diagram can be read as follows: the carrier actor has the duty, towards the sender actor, to deliver the shipped goods. However, the carrier has the privilege to subcontract the delivery service to others. The carrier is free to decide whether to subcontract or not. The dominance relation says that, if the carrier subcontracts, then it has not to perform the shipment. Otherwise, if the shipment is not subcontracted, and the carrier does not perform on its own, then the sender has

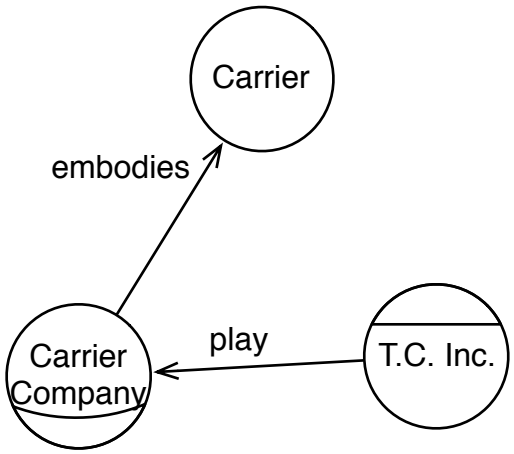


Figure 4.15: An embodiment relation.

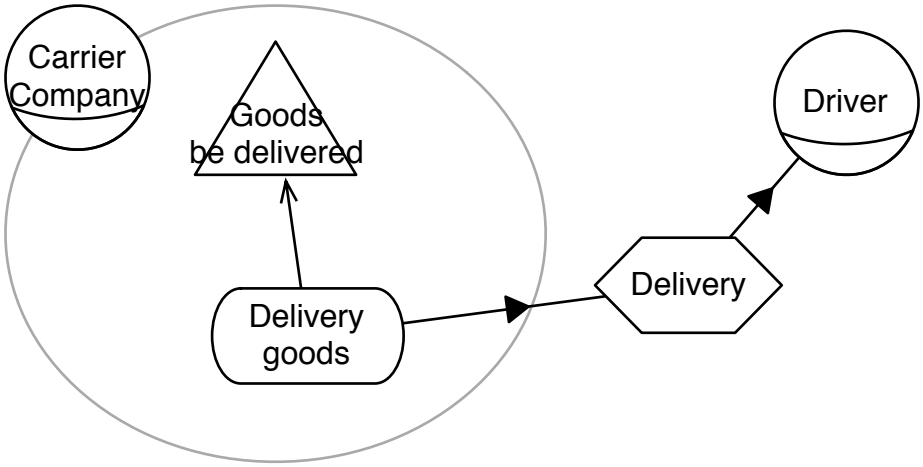


Figure 4.16: A realisation relation.

the power to withdraw the mandate to the carrier.

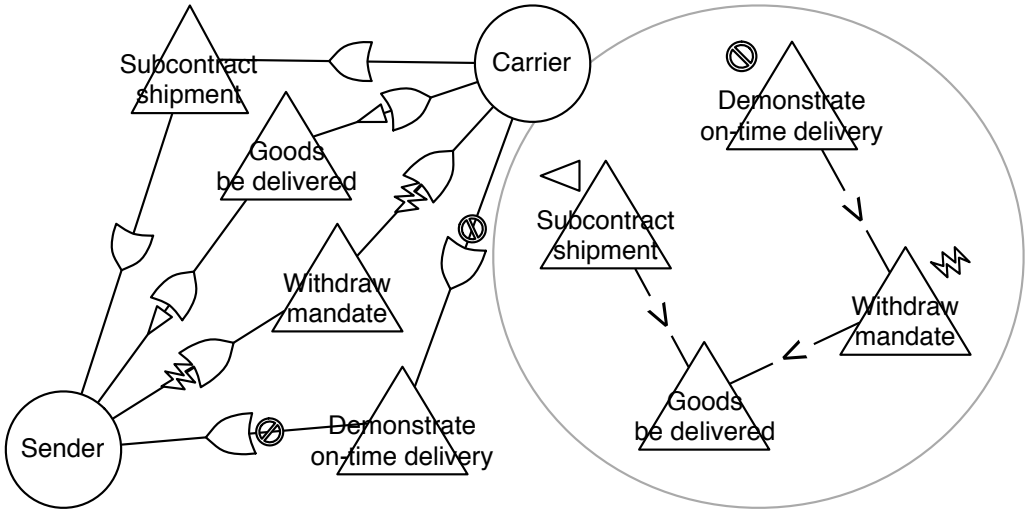


Figure 4.17: An example of use of the *Nòmos* modelling language.

Figure 4.18 summarises the notation used by the *Nòmos* language. In the next chapter, we will detail the semantics of the specific constructs introduced by the *Nòmos* framework (embodiment, dominance and realisation) and will describe how to use them to assess compliance.

The *Nòmos* language

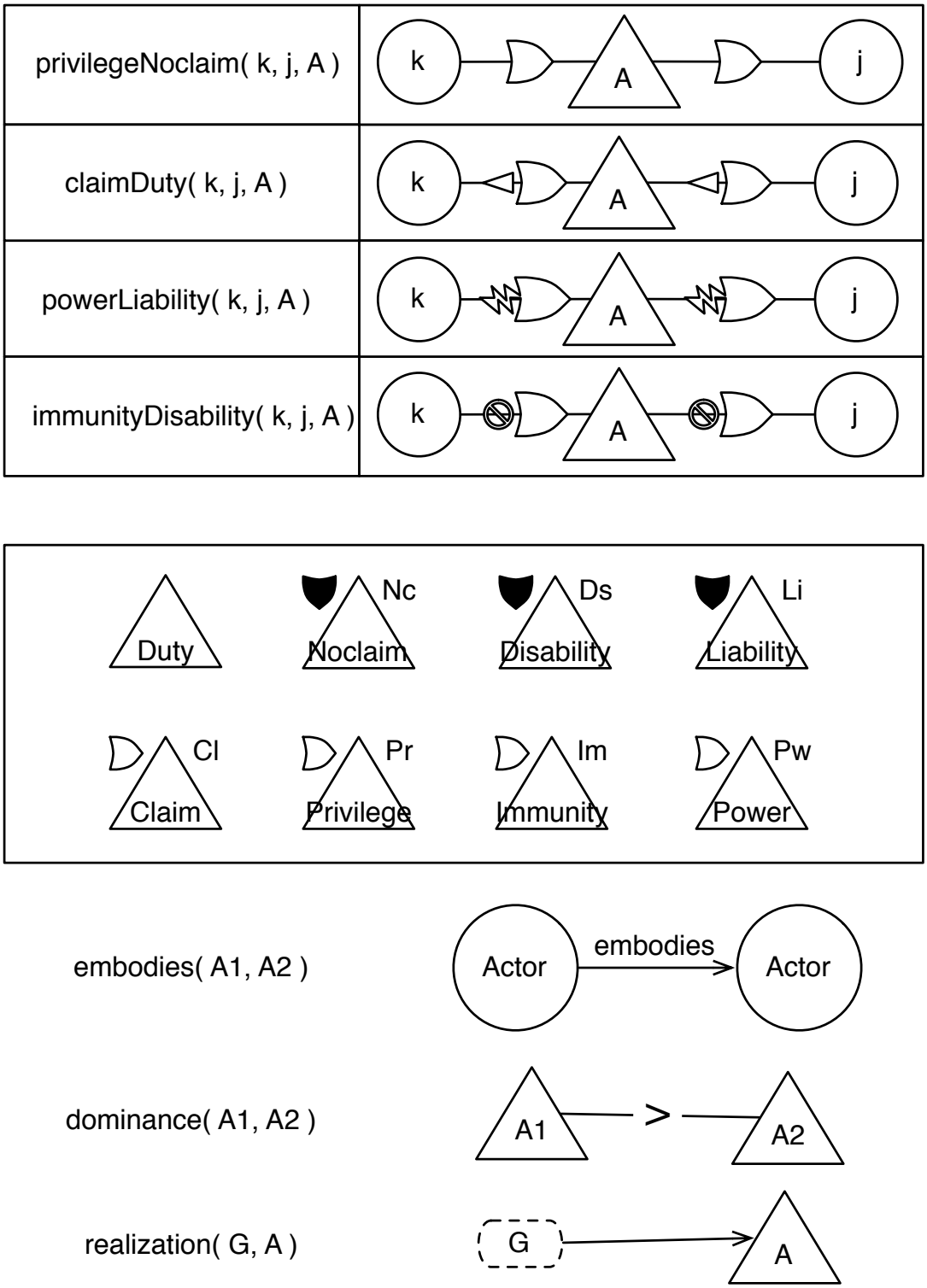


Figure 4.18: The primitives for the *Nòmos* visual languages.

Chapter 5

Compliance analysis

We stated the problem of law compliance of requirements as the condition that, under some assumptions K, whenever R holds, L holds too. In terms of models, R can be represented by means of goals, and we introduced the *Nòmos* modelling language that can be used to represent L by means of normative propositions. Given a model, comprised by elements of both R and L (i.e., goals and normative propositions), we can make explicit the entailment relation of Equation (3.6) as a property of such model.

5.1 Compliance assessment

As stated in Chapter 3, law is comprised by a collection of propositions defining states of the world. In Chapter 4, we have provided an ontological analysis of such propositions, and linked them with conceptual models for requirements. In this section, we will detail the relation between normative proposition and states of the world, which in turn is the key to provide a solution for the requirements compliance problem.

5.1.1 State-level compliance

According to (van Lamsweerde and Letier, 2000), a goal "defines a set of desired behaviors, where a behavior is a temporal sequence of states". If a system, which supports a goal g, is in one of the states defined by the goal, then the goal is satisfied (Giorgini et al., 2002). So we define $W(g)$ the set of states of the world, in which the goal proposition is true:

$$W_G(g) = W(g)|g = \top \tag{5.1}$$

Applicability As explained in previous chapter, a normative proposition is comprised by four elements: the addressee of the normative proposition, who is also the holder of the right specified in the normative proposition, and the passive subject in the generated legal relation; its counter-party, or active subject of the legal relation; the right type; and the specified action:

```
<np> := { <actor>, <counter-party>,
          <rightType>, <action> }
```

For example, let consider the sentence "Hospitals must keep patients' personal data closed". Out of this sentence, a normative proposition can be instantiated:

```
np1 = { "Hospital", "Patient", "duty", "Patient's
        data kept closed" }
```

In other words, Hospitals have the duty, towards their patients, to keep their personal data closed; or, patients have the claim, towards the Hospital they refer to, to have their personal data kept closed. Within the boundaries of a legislation - i.e., without taking into consideration the domain, its stakeholders and

their actual behaviour - this normative proposition is necessarily true: the proposition is actually creating the right by uttering it; so the normative proposition is tautologically true.

When the domain is taken into consideration, things change. Requirements express the desires of stakeholders that will be supported the system, whose behaviour will match their original wishes. According to the requirements designed for that behaviour, it will made possible a set of states rather than another. In each state defined by such behaviour - designed by means of its requirements - each of the four components of the normative proposition above can be true or false. In other words, given a state of the world w,

`<action>` is true if in the given state the result specified by the action is produced. In the example, if patient's data is kept closed the `<action>` component is true. If patient's data is disclosed `<action>` is false. Given a normative proposition *np*, we write $a(np)$ to indicate the truth value of the `<action>` element. Therefore, the set of states of the world in which $a(np)$ is true is the set of applicable states:

$$W_A(np) = W(a(np))|a(np) = \top \tag{5.2}$$

`<rightType>` is true if in the given state it is true that such right is applicable to the specified action. As explained more in detail later, in the above example if the Patient's data is kept for generic health care purposes the duty holds — i.e., it is evaluated as true. However, if it is processed for other purposes - for example, security measures - the duty does not hold. Similarly, if the data is disclosed to another institution, which in turn provides health care services, the duty does not hold.

`<holder>` and `<counter-party>` are true if the one, who is in charge of fulfilling the right, and the one, who owns the correlative right, are the actors addressed by the normative proposition.

If a certain right holds, its holder is the actor under analysis and is held against the proper counter-party, we write $h(np) = \top$; otherwise, we write $h(np) = \bot$. Said differently, we reduce the truth values of holder and counter-party to particular applicability criteria. Therefore, the set of states of the world in which $h(np)$ is true is the set of addressed states:

$$W_H(np) = W(h(np))|h(np) = \top \tag{5.3}$$

Figure 5.1 depicts the possible combination of the set of states defined by g, $a(np)$ and $h(np)$. The combination of these values define the compliance of a given state of the world. Table 5.1 reports the truth values for the components of compliance analysis and for the resulting compliance condition. The first row of the table refers to the nature of the right type carried by a normative proposition — e.g., claim, duty, privilege, power and so on. Given a set of states of the world, the "Right" row informs whether the right is applicable or not. The second row refers to the action carried by the normative proposition. The table reports true if the proposition describing the action is entailed in the specified state of the world, and reports false otherwise. The third row refers to the requirement we want to evaluate for compliance. Finally, the fourth row of the table reports a value that expresses whether a given state (numbered according to Figure 5.1) is compliant or not.

Table 5.1: Truth values for compliance analysis

		1	2	3	4	5	6	7	8
$h(np)$	(Right)	$\top$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
$a(np)$	(Action)	$\top$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\top$	$\bot$
g	(Goal)	$\top$	$\top$	$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\bot$
admissible		$\top$	$\bot$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$

In cases #5 to #8, $h(np)$ does not hold. It means that the specified right is not addressing this state of the world; so is a goal g is describing this state as

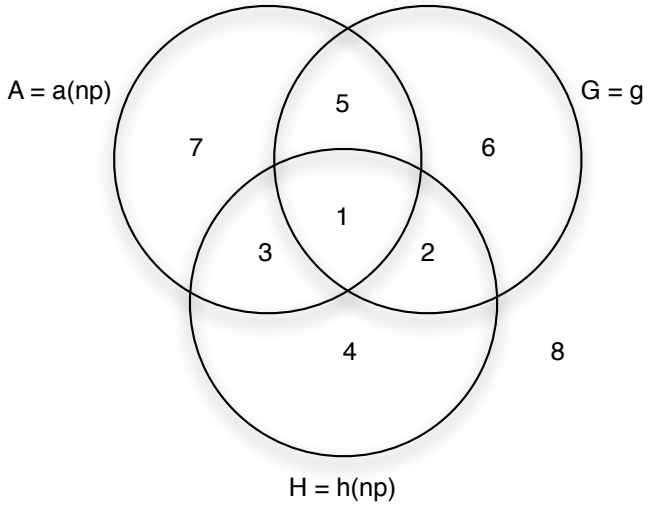


Figure 5.1: Venn diagram representing the truth table of compliance

wants by the stakeholder, a requirements specification derived by g on this state is admissible, but if this state is not expressed through a goal it is admissible as well. Since there is no possibility to be un-compliant in these state (because the law is not producing effects here) we say these states are compliant.

In case #1, the right is addressing that state, and the world is as well in a state described by the normative proposition; if a requirement is derived by a goal on this state (i.e., g holds), in the state of the world #1 compliance exists.

In case #2 the goal g hold, but the normative proposition prescribes something that in not satisfied in that state; in this case, we are out of the boundaries of the law and this state is not compliant.

In case #3, the right is applicable, and its specified action holds, but no goal exist, which will lead to that state of the world. This is not a sufficient condition to argue compliance, but since it is not possible to exclude un-compliance, we derive that this case is un-compliant.

In case #4, no goal exists as well, but no action is prescribed, so this case is admissible and we say it is compliant.

5.1.2 NP-level compliance

Despite its atomicity, a normative proposition does not univocally identify a *single* state of the world — rather, it defines a *set of* states of the world that satisfy the proposition. On the other hand, goals represent states of the world desired by actors. Atomic compliance refers to the condition, in which the states of the world defined by a requirement (or a set of requirements) are also defined by a *single* normative proposition.

If we consider a bunch of world states, more than one case, out of those represented in Table 5.1, entail one normative proposition and one goal. If $W(g)$ is the set of states entailing the goal g, and $W(np)$ is the set of states entailing the normative proposition *np* (i.e., in which both $a(np)$ and $h(np)$ are true), three cases can occur, namely: (a) Strong compliance, if $W(g) \subset W(np)$; (b) Partial compliance, if $W(g) \cap W(np) \neq W(g)$; And (c) Non-compliance $W(g) \cap W(np) = \emptyset$.

Strong compliance $W(g) \subset W(a(np))$ - Supposing that $R = \{g\}$ - i.e., that the set of requirements is comprised by exactly one goal g, the set of states of the world entailed by the goal g is a subset of the set of states of the world entailed by *np*, as in Figure 5.2. This means that there is no possibility that, having reached goal g, an actor is not compliant with the normative proposition *np*. For example, if *np* = "Crossable passage", and g = "Door kept open", in our world whenever the door is open the passage is crossable.

If this can be argued, then the goal g is the realisation of the law-specified action a belonging to *np*.

$$g \models np \tag{5.4}$$

and an actor, who wants g, is compliant with *np*.

The relation between goals, normative propositions and states of the world has a crucial role to establish compliance. As mentioned, such entailment re-

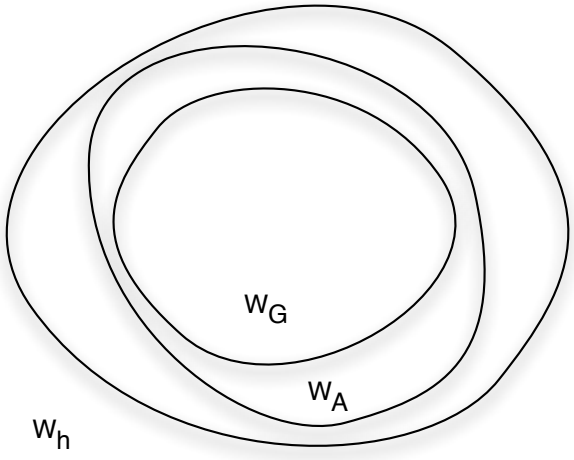


Figure 5.2: The case of a strong compliance between goal and normative proposition

lation between goals and normative propositions has to be *argued*. If it can be argued, then we take the argumentation as one of the domain assumptions K.

From this definition, ti follows easily that, if $R = \{g\}$ and $L = \{np\}$, then Equation (3.6) ($R, K \models L$) holds.

Further detailing the requirements set by adding new goals, may change that equilibrium. Supposing that in our requirements two goals exist, $R' = \{g, g_1\}$, then the relation between g and g_1 causes $W(R')$ to change with respect to $W(R)$. Two classes of relations exist: AND and OR relations.

- AND relations. A relation between two (or more) goals is defined AND relation if and only if the set of states of the world entailed by the goals is the intersection of the states of the world entail each goals.

$$g_1 \Box g_2 \leftrightarrow W(g_1, g_2) = W(g_1) \cap W(g_2) \tag{5.5}$$

  Notice that with AND relation we aren't actually referring to any specific construct of a modelling language (such as *i**), but it's possible to map

it into a specific construct. For example, as will be explained in the next chapter, we consider *i** Or-decompositions as an AND-relations. Let suppose that for an actor a goal "Eat pasta" is Or-decomposed into "Go to the restaurant" and "Go home". The two resulting alternatives will be: (1) Go to the restaurant AND eat pasta; (2) Go home AND eat pasta. It will never be the case that the actor will go to the restaurant without eating pasta, or will eat pasta without going to the restaurant or at home, respectively.

- OR relations. A relation between two (or more) goals is defined OR relation if and only if the set of states of the world entailed by the goals is the union of the states of the world entail each goals.

$$g_1 \Diamond g_2 \leftrightarrow W(g_1, g_2) = W(g_1) \cup W(g_2) \tag{5.6}$$

For example, we consider *i** contribution as an OR-relations. Let suppose that for an actor a goal "Go to the restaurant" contributes positively to the goal "Eat pasta". This means that three alternatives can occur: (1) the actor goes to the restaurant and eat pasta; (2) it goes to the restaurant but will not able to eat pasta; and (3) it will eat pasta without going to the restaurant.

If in R' a $\Box$ relation holds between g and g_1, then $R' \models R$ and strong compliance is ensured. For example, in *i** it's common practice (even if not standardised) that the sub-goals in an AND or OR decomposition are semantically a logical refinement of the decomposed goal, so that it's not possible to reach the sub-goals, or one of them respectively, without reaching their parent.

If in R' a $\Diamond$ relation holds between g and g_1, then $R' \not\models R$. This means that some of the new requirements may be not aligned with legal prescriptions. We will show in the following that such other cases are solved by reducing them to the strong compliance case.

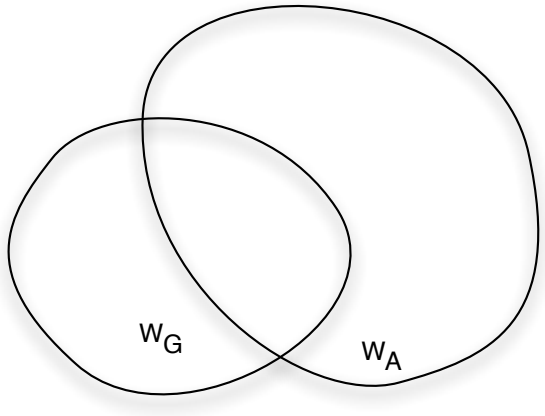


Figure 5.3: The case of partial compliance of a goal with respect to a normative proposition

Partial compliance $W(g) \cap W(np) \neq W(g)$ - Some of the states of the world entailed by the goal g also entail *np*, whereas some other don't. Figure 5.3 depicts this case. For sake of simplicity, we consider $h(np)$ to be always true, so that the set H is not shown. Later on, we will introduce $h(np)$ again. Under this assumption, we have that $W(np) \equiv W(a(np))$. This is possible because, as Table 5.1 shows, every state in which $h(np) = \bot$ is admissible (i.e., is not un-compliant).

In case of partial compliance, it is *possible* to be compliant by fulfilling g, but it is also possible that, depending on how the goal is fulfilled, a violation occurs. For example, if a law states that "patients' data should not be disclosed", and the stakeholder goal is "patients' data is processed electronically", we can't declare compliance of the goal, because processing patients' data electronically does not ensure that data is not disclosed electronically; however, we can't declare un-compliance as well, because that goal does not exclude the possibility to keep data closed.

Again, adding new goals to the requirements set may alter their compliance

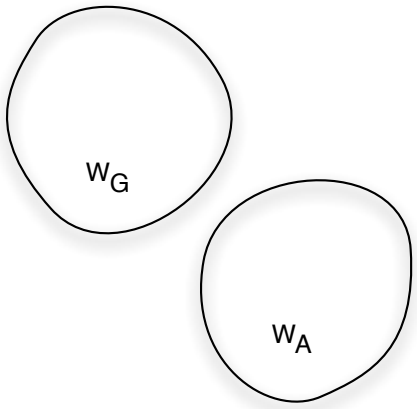


Figure 5.4: The case of mutual exclusion between a goal and a normative proposition

condition. If $R' = \{g, g_1\}$, with $g \Box g_1$, and $W(g_1) \cap W(a(np)) = \emptyset$, then it's never the case that $R' \models W(a(np))$. We fall in the non-compliance case.

Non-compliance $W(g) \cap W(np) = \emptyset$ - The set of states of the world entailed by the goal g does not intersect at all with those intersected by *np*. Under the above assumption that $h(np)$ always holds, this means that g holds as well, but $a(np)$ does not hold. This case is represented in Figure 5.4 and means (as in case #2 of Table 5.1) that we are in an un-compliance state.

Non-applicability A special case happens when the set of states of the world entailed by the goal g does not intersect at all with those intersected by $a(np)$, but $h(np)$ does not hold in such states. In this case, as in Table 5.1, the normative proposition is not applicable, and there is no need to be compliant.

5.1.3 Law-level compliance

Intuitively, if a law L is comprised by a set of normative propositions $\{np_1, ..., np_n\}$, then in order to be compliant with that law its necessary to be compliant with

each of its normative propositions. If G is a set of goals $\{g_1, ..., g_{gn}\}$ and $\square_{k=1}^{gn}$ is an $\square$-relation among its elements, we have that:

$$\forall np \in L, \exists G | wants(j, G), \bigsquare_{k=1}^{gn} g_k \models np \tag{5.7}$$

This is the general compliance condition for requirements represented as goals.

Legal alternatives In Figure 5.5, a set of possible states of the world is depicted. A law L addresses that world, and specifies, by means of two normative propositions, which properties have to retain the states of the world to be legally acceptable. The two normative propositions holding in the world are np_1 and np_2, with the latter dominating the first. So we have that $L = \{np_1, np_2\}, np_1 > np_2$. In the picture are shown the states in which the propositions hold — i.e., the states in which $h(np_1)$ and $h(np_2)$, respectively, are true. By construction, we let np_1 hold in w_1, w_2, w_3, w_5 and w_7, and write $\{w_1, w_2, w_3, w_5, w_7\} \models np_1$. Similarly, $\{w_1, w_4, w_5, w_7\} \models np_2$.

Generally speaking, the applicability of a certain law depends on the sum of the applicability of its normative propositions.

$$h(L) = \bigcup_{k=1}^{n} (h(np_k)) \tag{5.8}$$

If $np_1 > np_2$, then for every state of the world w, if np_1 does not hold but np_2 holds, w also holds. In other words, if a behaviour is not valid according to a certain normative proposition, but there is a second normative proposition that derogates the first one and admits the behaviour, then the behaviour is legally admissible. So if L is comprised by the two normative propositions in dominance relation, the admissible states of the world are those in which either np_1 holds, or np_2 holds:

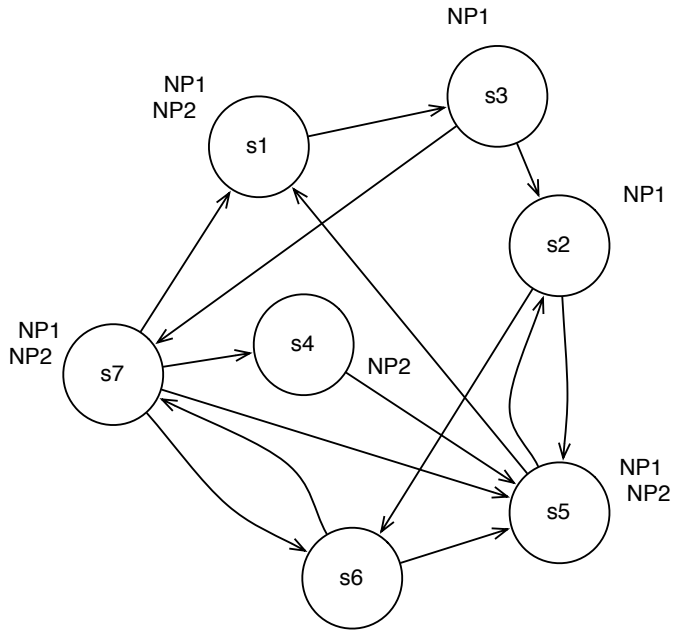


Figure 5.5: States of the world

$$L = \{np_1, np_2, np_2 > np_1\} \rightarrow h(L) = h(np_1) \vee h(np_2) \tag{5.9}$$

Conversely, if the normative propositions are disjoint (i.e., no domination relation exists), failing to comply with one of them immediately causes to be in a non-compliance state for the whole law. In fact, if a behaviour is not compliant the the normative proposition, there is no other normative proposition that admits that behaviour. We have that if L is comprised by the two normative propositions not in dominance relation, the admissible states of the world are those in which both np_1 and np_2 hold:

$$L = \{np_1, np_2\} \rightarrow L = np_1 \wedge np_2 \tag{5.10}$$

Dominance relations establish a partial order between normative propositions such that not every normative proposition has actually to be fulfilled. For example, a law $L = \{np_a, np_b, np_c\}$, with $np_b > np_a$. This means that np_b *dominates* np_a: as long as np_b holds, np_a does not, and it is quite common in law. Let

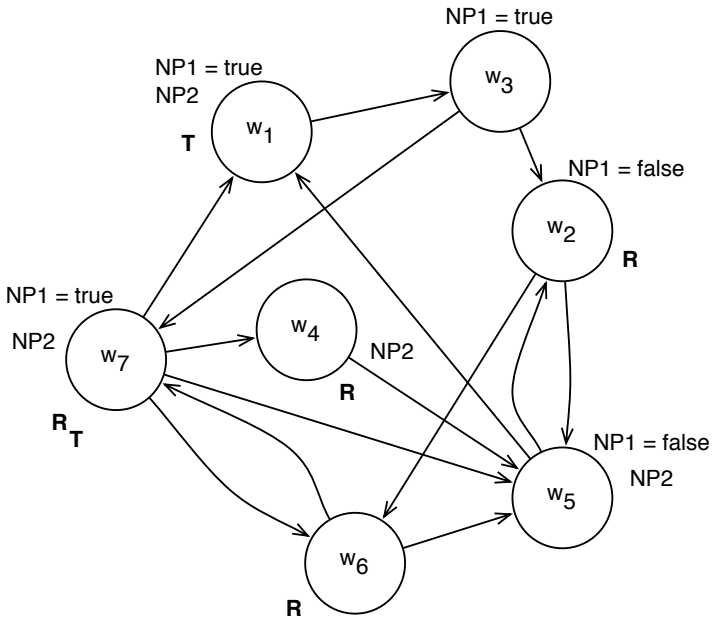


Figure 5.6: Legal states of the world vs. strategic states of the world

suppose that np_a says that it is mandatory to pay taxes, and np_b says that it is possible to use the same amount of money, due for taxes, to make investments. $np_b > np_a$ means that, if a company makes an investment, then it does not have to pay taxes for the same amount. Now, with the given *np*s and dominance relations, companies have two alternatives: $L_1 = \{np_a, np_c\}$, and $L_2 = \{np_b, np_c\}$. We call these alternative prescriptions *legal alternatives*.

A set of normative propositions L^α is a legal alternative on a law L if and only if L^α is a subset of L and $L^\alpha \models L$ — i.e., a compliance solution for L^α is also a compliance solution for L.

As long as many alternative prescriptions exist, the need arises for selecting the most appropriate one. The above example of tax payment is simple because only one normative proposition is different in the two cases, so it could easily be solved. Generally speaking, legal alternatives can be different for a large number of normative propositions, which can change, appear or disappear in a given legal alternative, together with their dominance relationships, so that the overall topology of the prescription also changes. This causes the risk that the

space of alternatives grows too much to be tractable, so the ultimate problem is how to cut it. This problem will be discussed in the next chapter.

Legal alternatives influence the notion of compliance, introducing variability in the strategic model. We said that an actor is compliant with a law L if it is compliant with every normative proposition of that law. However, aggregate law compliance needs more than this. The complexity of law comes from the fact that single prescriptions are subordinated to applicability conditions, exceptions, derogations and so on. In previous chapter we have introduced the notion of *dominance*. We use it now to model such kind of conditional elements. If an np_a dominates another np_b we write $np_a > np_b$ (and if $np_a > np_b$ and $np_b > np_c$ then $np_a > np_c$). So, for aggregate compliance, the dominance relation means that, *whenever* np_a *is in contrast with* np_b complying with np_a implies that it is not necessary to comply with np_b. For example, let suppose that according to a certain law John has the duty of paying a tax of a certain amount; but, if he invests the same amount in entrepreneurial activity, he can omit the tax payment. In this case, complying with the second sentence means no need to pay taxes. Accordingly, if np_a is only partially in contrast with np_b, $needToComply$ holds for both normative propositions.

5.2 Compliance Modellisation

In previous chapter we have introduce the *Nòmos* meta-model in its parts that concern law modelling and compliance modelling. The semantics of compliance constructs (such as embodiment and realisation) was weakly defined, so in the following we will define it in terms of the theory introduced in previous section. Using such compliance constructs, we will be able to define the notion of compliance for a goal model.

5.2.1 Intentional compliance

The first requisite for a requirements compliance framework, as stated in Chapter 3 is the intentionality of actors to be compliant. In other words, intentional compliance consists in making the top-level decisions. In terms of models, having represented requirements as a goal model, intentional compliance consists in identifying the set of goals such that, once achieved, they allow for arguing compliance with a set of addessed normative propositions.

Applicability Partial compliance is the most common case when trying to achieve compliance, moving from a situation in which compliance does not exist or has not taken into consideration. For example, when a new law is created, processes that were previously allowed to be discretionally designed, are then affected by legal prescriptions. Rarely existing processes are fully un-compliant — although this is possible, when the law has the precise intent to prohibit such existent activities. And it's uncommon as well that the processes are already fully compliant. Most likely, new laws address existing processes to force their modification for adhering to legislator's objectives.

$$applicability(\textit{np}, g) \equiv W(g) \cap W(\textit{np}) \neq W(g) \tag{5.11}$$

Realisation As of previous section, a goal g is a realisation of a normative proposition *np* if and only if, for every state of the world w, if w entails g then w entails *np*.

$$realization(g, \textit{np}) \equiv (g \models \textit{np}) \tag{5.12}$$

Intentionality Referring to the general compliance condition seen above, compliance in a model consists (in absence of legal alternatives) in being compliant

with each normative proposition contained in the law. If G is a set of goals $\{g_1, ..., g_{gn}\}$ and $\square_{k=1}^{gn}$ is an $\square$-relation among its elements, we have that:

$$compliance(j, L) \equiv \forall \mathbf{np} \in L, \exists G | wants(j, G), \square_{k=1}^{gn} g_k \models \mathbf{np} \tag{5.13}$$

As explained above, legal alternatives are subsets of a certain law, which can be chosen according to the intention of the actor who wants to comply. Being compliant with a law is equivalent to be compliant with (at least) one legal alternative within the law:

$$compliance(j, L) \equiv \exists L^{\alpha} \in L, compliance(j, L^{\alpha}) \tag{5.14}$$

5.2.2 Ability

A compliance choice that satisfies the intentional compliance condition described above, needs to be operationalised in order to have effect. To do this, we distinguish *compliance goals* from purely strategic goals. A compliance goal is a goal, whose reason-to-be within an actor's rationale is the need for the actor to be compliant with one (or more) normative proposition. A compliance goal is needed when the actor is addressed by the normative proposition, which in turn affect some of the actor's goals. In practice, a compliance goal is either the goals that has been identified as realising a certain normative proposition, or sub-goal of it. In the first case, we define it as:

$$complianceGoal(g) \equiv \exists \mathbf{np}, realization(g, \mathbf{np})$$

while in the second case, we define it as:

$$complianceGoal(g) \equiv \exists g_c, complianceGoal(g_c), relation(g, g_c)$$

The operationalisation of a compliance goal consists in providing a means to achieve the goal. In terms of modelling, in consists in finding a task that

means-end a compliance goal:

$$operationalCompliance(g) \equiv complianceGoal(g), task(t), meansEnd(t, g)$$

or, more generally, any compliance goals, whose sub-goals can be operationalised, can in turn be operationalised:

$$operationalCompliance(g) \equiv \forall subgoal(g_s, g), operationalCompliance(g_s)$$

5.2.3 Auditability

Only the running system is actually compliant or not compliant. And true compliance can only be established *ex post* by the judge. Formal proof of run-time compliance can't be given at requirements time: there are properties of law that makes that the compliance condition can only be stated *ex-post* by the judge - e.g., the subsequent design could be wrong, people could behave differently from what is assigned to them according to their roles, software programs could be bugged and also behave differently from what expected, as well as the intentional ambiguity. Auditability refers to the capability to provide formal evidence (proof) that a certain behaviour has been operated in alignment with a normative proposition. If a compliance goal is operationalised by means of a certain task, we say that the task is auditable if uses some resource:

$$auditable(t) \equiv task(t), resource(r), affects(t, r) \tag{5.15}$$

The idea behind this is that, whenever a resources is added to an i* models for requirements engineering, it does not represent an abstract mental concept, but a concrete artefact that is needed by a task to operate. When the access to the resource modifies it, the resource becomes a trace back to the task that operated. In this case, we say that the task *affects* the resource.

5.2.4 Traceability

Compliance choices have relevance in two different moments. The first moment is when the choices are actually made, in order to align requirements with law. The second moment is subsequent to the requirements elicitation, and it is when the motivation of the requirements is needed. For example, when the requirements are going to change it is necessary to know where did the requirements come from, if they exist because of a strategic choice or due to a legal necessity. Also, when law changes, it raises the need for knowing which requirements are affected. This traceability information is also the first element of provability of compliance choices.

5.2.5 Model-wise compliance

As stated in Chapter 3, four requisites are necessary to argue the design-time compliance of requirements, namely: awareness, ability, traceability, auditability.

Chapter 6

Compliance Modelling Process

The conceptual tools described in the previous chapters are intended to be used as part of the software development process, and specifically as part of the process that goes form requirements gathering to their specification. The *Nòmos* framework is integrated in particular with the principles of goal-oriented modelling processes. Laws are iteratively modelled together with requirements, in an interleaved process, and the specification is derived from the operationalisation of high-level, law-compliant requirements. Being R the requirements model built at iteration $n-1$, and assuming that it is law-compliant, performing a modelling iteration n means modifying R such that a new set of requirements R' is generated. R' must also be law-compliant, so that, at every iteration, the condition must hold, the the newly generated set of requirements match both the previous set R as well as laws L:

$$R', K \models R, L \tag{6.1}$$

This is the process equation for modelling law-compliant requirements. In the following, we will detail how to ensure this equation to hold through a set of modelling tasks.

6.1 Elicitation process

Roughly speaking, software requirements consist in information concerning what the system should do once developed and deployed. Regardless of how such information is represented - natural language, diagrams, prototypes and so on - for sake of simplicity we assume that it can be expressed by means of a set of propositions, so that $R = \{r_1, ..., r_n\}$. The requirements elicitation process consists in the sequence of activities needed to form R. The process starts when $R = \emptyset$, and ends when there is no more information to add. During the process, in the requirements set some properties must hold, such as stakeholders acceptability, security, dependability, and so on. If the requirements set does not match these properties, information can be added, removed, or changed. Information is added if it can be gathered from the domain, acquired from the stakeholders or deduced from existing knowledge. When the requirements set satisfies every property, no new information has to be added, and the process ends. At this point, the identified requirements, if well engineered, should be law-compliant by construction (Siena et al., 2009b).

The process is exemplified in Figure 6.1. In the initial state, when $R = \emptyset$ and $L = \emptyset$. Requirements are not in equilibrium: there is at least one property not satisfied: stakeholders acceptability. Stakeholders have needs, but the requirements don't capture those needs. The process starts by gathering knowledge from the domain stakeholders, and adding it to the model as a requirement. Adding a requirement - for example, adding r_1 - results in the accomplishment of an activity, represented in the figure as an arc. Each of the arcs in Figure 6.1 corresponds to performing a requirements engineering activity that goes from a state of local equilibrium to another state with local equilibrium. Such activity can consist in (i) adding or removing a requirement to the requirements set or to its model; (ii) adding or removing a law fragment to or from the set of applicable laws; and (iii) adding or removing assumptions. From a different

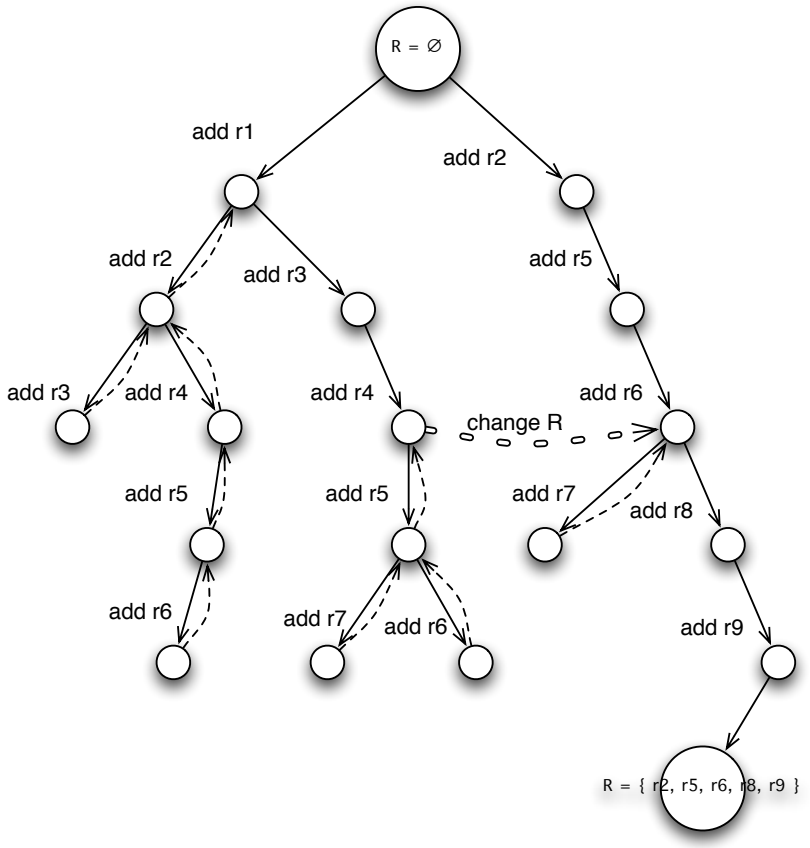


Figure 6.1: The requirements elicitation process

standpoint, we classify activities in four different classes, namely:

- Law discovery activities, which consist in the identification of the laws that at a certain time concern the actor under analysis.
- Requirements alignment activities, which consist in incrementally model requirements maintaining some compliance condition.
- Compliance verification activities, which consist in the reconciliation of an existing set of requirements with a law.
- Vulnerability analysis activities, which consist in identifying and resolving issues that arise from the need of being compliant.

The general law-compliant requirements elicitation process Algorithm 6.3 presents an algorithmic description of how such requirements engineering activities are linked, and how they can be performed in an interleaved way in order to build models of law-compliant requirements (Siena et al., 2009a). The algorithm takes as input two arguments: a set of requirements R and a set of normative propositions L. Both inputs can be empty in the first iteration, and the same algorithm can be called iteratively when new requirements are gathered form the stakeholders for reasons other than law compliance. The process ends when a full compliance solution is found for the input requirements.

Line #1 Requirements are firstly gathered from stakeholders. This activity consists in every action that is traditionally performed in requirements engineering to gather knowledge from stakeholders, such as, for example, interviewing the stakeholders, acquire document and other relevant resources, and so on. The gathered information is then processed and translated into a goal model.

If the input set of requirements R is empty, $gatherRequirements$ is equivalent to the initial requirements gathering, but if R is other than $\emptyset$, this means that some information has already been gathered. In this case, $gatherRequirements$ consists in a subsequent iteration of interviews with stakeholders to incrementally add information to R.

The returned value, R', consists in the newly acquired information about stakeholder requirements.

Line #2-#3 If the requirements gathering activity didn't produce any new information, R' is empty. In this case, the algorithm immediately ends returning R. This case is a termination condition.

Line #4 If some new requirement is gathered from stakeholders, the algorithm will from now on work on the augmented requirements set. This step is equivalent to modelling R' into the model of R.

Algorithm: elicitateCompliantRequirements
Input: $R = \{r_1, ..., r_n\}$, $L = \{np_1, ..., np_m\}$
Output: $\{r_1, ..., r_v\}$

```
1  R' = gatherRequirements(R);
2  if R' = ∅ then
3     return R;
4  R = R ∪ R';
5  L' = discorverLaws(R, L);
6  if L' = ∅ then
7     return R;
8  L = L ∪ L';
9  if checkRequirementsCompliance( R, L ) = true then
10    return R;
11 repeat
12    R' = generateNextComplianceAlternative(R, L, R');
13    if R' = ∅ then
14       return ∅;
15    if checkRequirementsAcceptance( R ∪ R' ) = true then
16       R'' = elicitateCompliantRequirements(R ∪ R', L);
17       if R'' ≠ ∅ then
18          return R'';
19    end
20 until true ;
```

Algorithm 1: Overall process for law-compliant requirements elicitation.

Line #5 Now the requirements set has changed, so the applicable laws could have probably changed too. The $discoverLaws$ function takes as input the working requirements set R and the already identified laws L. The function returns L', a set of normative propositions evaluated as applicable to some of the requirements in R. Worth saying that L' may contain normative propositions already present in L.

The behaviour of this function is detailed in Section 6.2.

Lines #6-#7 If no new applicable normative propositions have been identified

(so L' is empty) the algorithm terminates. In fact, the $gatherRequirements$ function has previously added some new requirements, but no laws address such new requirements, so there is no need to assess compliance on them.

Line #8 As previously for R, the algorithm now works on a new set of normative propositions, so L contains now the existing ones and the newly discovered.

Lines #9-#10 We have now a set of requirements and a set of normative propositions. We still don't know whether something has to be done to be compliant — it could be possible that the gathered requirements are already compliant. This happens for example when the new requirements are in $\Box$ relation with the old ones (see Chapter 3).

The behaviour of the $checkRequirementsCompliance$ function will be detailed in Section 6.4.

Line #11 At this point an iteration begins, which has the purpose of finding a valid compliance solution for R. The iteration continues until a solution is found, or it has been found that no solution exists.

Line #12 The purpose of the $generateNextComplianceAlternative$ is to refine the requirements set R until a point, in which compliance can be stated with respect to L. On requirements represented as *i** models, the function adopts a modelling approach that falls back into goal oriented methodologies, such as the Tropos methodology (Susi et al., 2005). Differently form such goal-only approaches, the $generateNextComplianceAlternative$ finds a solution that obeys to the general compliance principles as described in Chapter 5. The parameter R' is used here to highlight that, at each iteration, a different solution is returned, which is ideally the subsequent after R'.

Worth saying that this function differs from the $gatherRequirements$ because the latter consists in the addition of new requirements that come from stakeholders, whereas the first introduces new requirements as result of a modelling activity subject to compliance rules. As a consequence, the latter is necessarily consistent with stakeholder needs but suffers the risk to be un-compliant; viceversa the first, if found, is necessarily law-compliant but suffers the risk of being not acceptable by stakeholders, as will be explained in the following.

The behaviour of the function will be detailed in Section 6.3.

Lines #13-#14 If no *further* compliance solution is found, it means that it is not possible to satisfy the given stakeholder needs and at the same time be law-compliant. So the algorithm exists returning a failure (i.e., an empty set).

Line #15 If at least one compliance solution is found, the solution must be checked against stakeholders acceptability. In fact, we know that the solution found is law-compliant, but we still need to know it the modelling choices made by the analyst still satisfy stakeholders. Notice that this step only limits to evaluate the acceptability of requirements, without modify them.

Line #16 After having checked stakeholders' acceptability if the proposed solution, we have a set of requirements ($R \cup R'$) that has changed from the last compliance verification (done in step #9). So the *elicitateCompliantRequirements* function is called again, recursively. The function will return either an increased set of requirements, law-compliant and stakeholders-acceptable, or an empty set.

Lines #17-#18 If the function returned a non-empty set, this is in turn returns and the algorithm ends. If the function returned an empty set, it means

that this compliance solution can't be at the same time law-compliant and stakeholders-acceptable, and another iteration is done, searching for another compliance solution.

As said, *gatherRequirements* and *checkRequirementsAcceptance* do not depend on law-compliance theory, and any requirements engineering technique is valid for carrying out such activities. In the following, we will detail how the remaining three process fragments work, namely: the *discoverLaws*, the *checkRequirementsCompliance*, and the *generateNextComplianceAlternative*.

6.2 Law discovery

Compliance is a notion that involves two components: laws and requirements. Law exists in the domain, regardless of the acquisition of any requirement, so in principle law models can be the starting point of the modelling process. However, law is an extremely complex artefact, and providing a model of it as a whole is useless, as the difficulty in managing it overcomes its utility. Not every law has to be taken into consideration for every requirements model. In many cases, no laws at all have to be considered. Rather, only those laws or law fragments that are applicable to the given domain are relevant to be modelled. We called applicability the condition, which causes some (fragments of) laws to be taken into consideration or not, depending on a context. Applicability comes from two main factors: the nature of the addressed subject, and the nature of the addressed behaviour. Specifically, applicability refers to the match of these two factors with the nature of the stakeholders and of their behaviour. So, the process of law discovery consists in finding such applicable laws.

When performing requirements elicitation, we interleave *i** modelling of domain stakeholders and normative modelling. Figure 6.2(a) depicts a typical scenario that occurs while exploring a regulated domain. Let us suppose that, during the initial phase of requirements gathering, we have knowledge on the

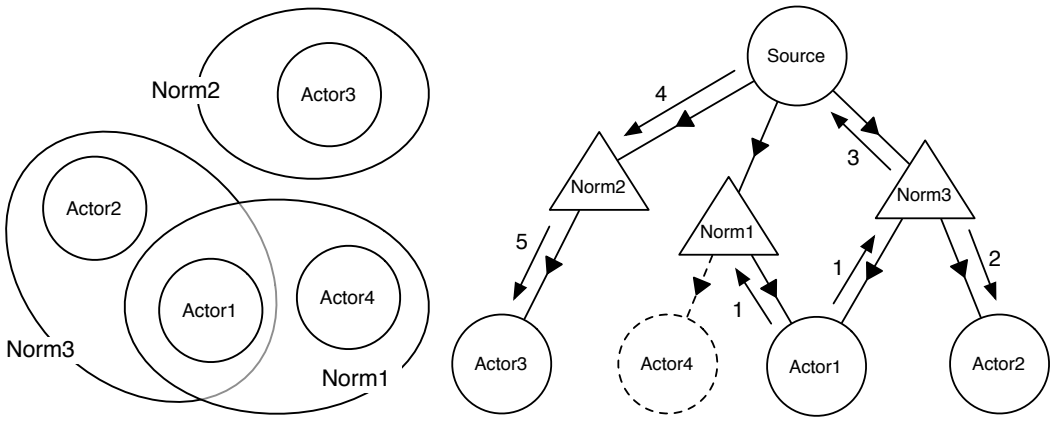


Figure 6.2: (a) The applicable laws with respect to the stakeholders they address. (b) The discovery flow.

existence of [Actor1], while [Actor2], [Actor3] and [Actor4] are hidden. Here hidden means that the interviewed stakeholder(s) did not explicitly mention them, or if they did, they did this without highlighting their role. This is a typical problem of tacit knowledge. Returning to the example, we know that [Actor1] is called to comply with two laws, Norm1 and Norm3, and so we proceed with the analysis of such laws (Figure 6.2(b), step 1). The analysis consists in either a human activity, consisting in reading law texts and evaluating them; or, it may consist in automatic, tool-supported parsing of laws. In any case, if a law fragment is considered relevant on the given domain, a normative proposition is instantiated, and added to the model.

If the laws address other actors, they are added to the domain model. For example, the arrow of step 2 indicates that [Actor2] is added. At this point [Actor3] is still hidden. However, by analysing the source of Norm3 (step 3), we are able to find Norm2 (step 4), which in turn leads us to [Actor3] (step 5). We store all this information in a norm diagram such as the one in Figure 6.2(b) for further analysis of the model. So we have discovered [Actor2] and [Actor3]; but are those actors actually part of the domain? For sure we only want to model those ac-

Algorithm: discoveryLaws
Input: $R = \{r_1, ..., r_n\}$, $L = \{np_1, ..., np_m\}$
Output: $\{np_1, ..., np_v\}$
1 L' = $applicableLaws(R, L)$;
2 A = $actorsIn(L')$;
3 **foreach** $j \in A$ **do**
4 L' = $L' \cup discoveryLaws(R, L \cup L')$;
5 **end**

Algorithm 2: Law discovery process.

tors that are relevant for the requirements specification. For this purpose, the analysis of the norms schema allows us to discard those actors that are irrelevant for the problem under study. For instance, having discovered Norm1, we could observe that it lays down prescriptions attaining different topics, not in our interest. So, [Actor4] will not enter in the description of the domain.

6.3 Generating Compliance through Modelling

In Section 6.1 we have introduced the $generateNextComplianceAlternative$ function as a means to generate a solution of law-compliant requirements through modelling. Such function is basically a requirements engineering method that takes as input a set of requirements R and a set of normative propositions L, and returns a new set of requirements R', which is law-compliant.

The process consists in 6 basic steps, namely:

1. The establishment of the proper embodiment relations between domain actors and legal subjects.

2. The processing of the modelled law in order to identify the possible legal alternatives.

3. check of applicability condition of normative propositions to existing goals.

4. identification of top-level compliance goals possibly able to realise the normative propositions.

5. decomposition of compliance goals until finding a task assignment that operationalises them.

6. attachment of autidability resources to compliance tasks.

Running Example To describe the modelling tasks that will be performed during this process, we will use an example scenario. The input of the process is the model of law depicted in Figure 7.1. The initial requirements R is given by a healthcare-centered scenario: a US hospital has its own internal reservation system, consisting in the employee personnel answering phone calls and scheduling doctors appointments on an agenda. The hospital wants now to set up a new information system - to manage the reservations, quickly retrieve the availability of rooms and devices in the hospitals, and ultimately optimise the reservation according to the needs of the patients and doctors - and to reduce expenses the hospital wants to outsource the call center activity to a specialised company. Since the reservation system is intended to deal also with the patients PHI, system requirements have to be carefully analysed to be made compliant with the HIPAA law.

The output of the process for our running example is depicted in Figure 7.2. In the following, we will detail the modelling process that produces that output, describing the *why* and *how* of each step of the process, and its results.

Step 1. Bind domain stakeholders with subjects addressed by law

Why. In the *Nòmos* meta-model of Figure 4.5, actors represent the binding element between laws and goals, but during modelling this binding can't be automatically deduced. Actors wanting goals are extracted from the domain analysis, while actors addressed by laws are extracted from legal documents. The different sources of information, as well as the different scope and interests

covered, raises the need to know who is actually addressed by which law.

How. The binding is operated by the analyst, possibly comparing how actors are named in the law, with respect to how they are named in the domain analysis - or, if law identifies the addressee by recalling the most notable (intentional) elements of its behaviour, then those elements are compared with the elements of the stakeholders actors behaviour. As of the *Nòmos* meta-model, the binding relies on the `embodiment` relation. When a domain actor is recognised to be a law subject, an embodiment relation is instantiated, and the corresponding rights are automatically assigned to the actor. Actors that are not part of the domain, but that interact with other domain actors have to be added to the requirements model. Otherwise, law subjects can be excluded from the requirements model.

Result. The result of this step is a model of rights as in Figure 7.1, in which actual domain stakeholders replace law subjects.

Running example. The [Hospital] under analysis in our domain is an entity covered by the law ([CE]). The [Patient] is the actor referred to as the [Individual] in the law. And the [Call Center] in this scenario is a business associate ([BA]) of the covered entity. Some actors, such as the [Secretary] and what has been called the [Authority] were not introduced in the domain characterisation, but have legal relations with other actors. Finally, some actors, such as the [Doctor] and the [Data Monitor] are not mentioned in the legal documents taken into consideration.

Step 2. Identify legal alternatives

Why. Dominance relations establish a partial order between NPs such that not every NP has actually to be fulfilled. For example, a law $L = \{np_a, np_b, np_c\}$, with $np_b > np_a$. This means that np_b *dominates* np_a: as long as np_b holds, np_a does not, and it is quite common in law. Let suppose that np_a says that it is mandatory to pay taxes, and np_b says that it is possible to use the same amount of money, due for taxes, to make investments. $np_b > np_a$ means that, if a company makes an investment, then it does not have to pay taxes for the

same amount. Now, with the given normative propositions and dominance relations, companies have two alternatives: $L^1 = \{np_a, np_c\}$, and $L^2 = \{np_b, np_c\}$. We call these alternative prescriptions *legal alternatives*. As long as many alternative prescriptions exist, the need arises for selecting the most appropriate one. The above example of tax payment is simple because only one normative proposition is different in the two cases, so it could easily be solved. Generally speaking, legal alternatives can be different for a large number of normative propositions, which can change, appear or disappear in a given legal alternative, together with their dominance relationships, so that the overall topology of the prescription also changes. This causes the risk that the space of alternatives grows too much to be tractable, so the ultimate problem is how to cut it.

How. To solve this problem, we introduce a decision making function that determines pre-emptively whether a certain legal alternative is acceptable in terms of domain assumptions, or if it has to be discarded. The decision making function is applied by the analyst whenever a legal alternative is detected, to accept or discard it. We define four basic decision making function (but hybrid or custom functions can be defined as well):

a) Precaution-oriented decision maker. It wants to avoid every sanction, and therefore tries to realise every duty. Immunities are also realised to avoid sanctions to occur.

b) Opportunistic decision maker. Every alternative is acceptable - including those that involve law violation - if it is convenient in a cost-benefit analysis with respect to the decision maker's goals. In a well-known example of this function, a company has decided to distribute its web browser application, regardless of governmental fines that have been applied, because the cost of changing distribution policy has been evaluated higher than the payment of the fine.

c) Risk prone decision maker. Sanctions are avoided by realising the necessary duties, but ad-hoc assumptions are made that the realised duties are effective and no immunities are needed. This is mostly the case in small companies

that do not have enough resources to achieve high levels of compliance.

d) Highly conform decision maker. This is the case in which legal prescriptions are taken into consideration also if not necessary. For example, car makers may want to adhere to pollution-emission laws that will only be mandatory years in the future.

Result. The result of this step is a set of normative propositions, subset of L, together with their dominance relationships, which represent a model of the legal prescription that the addressed subject actually wants to comply with.

Running example. Dominance relations of Table 7.1 define the possible legal alternatives. np_1 ([Don't disclose PHI]) is mandatory to avoid the sanction. np_5, [No known violations], is also mandatory; however, law recognises that the CE has no control over the BA's behaviour and admits that the CE can be not able to respect this normative proposition. To avoid being sanctioned, in case of violation the CE can perform some actions, [End the violation] (np_6) or [Terminate the contract] (np_7). So ultimately, np_6 and np_7 are alternative to np_5. In Figure 7.2, the hospital adopts a risk-prone strategy. According to the law model, if a BA of the hospital is violating the law and the hospital is aware of this fact, the hospital itself becomes not compliant. It is however immune from legal prosecution if it takes some actions, such as reporting the violation to the secretary (normative proposition [Report violation]). However, in the diagram the hospital does not develop any mechanism to face this possibility. Rather, it prefers to believe that the BA will never violate the law (or that the violation will never be known).

Step 3. Select the normative proposition to realise

Why. Another source of variability in law compliance consists in the *applicability conditions* that often exist in legal texts. For example, an actor may have a duty but only within a fixed period of time or only when a certain event occurs. So the problem arises, of which normative proposition has actually to be realised.

How. The hohfeldian taxonomy adopted by the *Nòmos* framework has a *de-*

scriptive nature: it means that normative propositions expressed in terms of the meta-model of Figure 4.5 does not specify how the world should be; instead, a normative proposition informs that a legal statement exists in the legislation. A property of such a descriptive approach is that legal statements, modelled with normative propositions, exist regardless of whether they are applicable or not. The *applicability* of a certain normative proposition could depend on many factors, both objective and subjective - such as time, happening of certain events, the decision of a certain actor and so on - and trying to exhaustively capture all these possibilities is hard and possibly useless for purposes of requirements elicitation. So, instead of trying to describe applicability in an absolute way (i.e., specify exactly when a normative proposition is applicable), we describe it in *relative* terms: i.e., we describe that *if* an existing normative proposition is actually applicable, then another normative proposition is not applicable. More specifically, we use dominance relation between two normative propositions, np_1 and np_2, and write $np_1 > np_2$ to say that, whenever np_1 holds (is applicable), then np_2 does not hold.

Result. This step returns the bottom-most normative proposition that has to be realised. I.e., if np_1 is still not realised, and np_2 is already realised, then $np_1 > np_2$ and np_1 is returned. If no other normative proposition exists, it returns nothing.

Running example. np_1 says that *"the CE may not disclose patient's PHI"*, and np_3 states that *"A covered entity is required to disclose patient's PHI when required by the Secretary"* - in this case, np_1 and np_3 are somehow contradicting each other, since np_1 imposes the non-disclosure, while np_3 imposes a disclosure of the PHI. But the dominance relation between np_3 and np_1 states that, whenever both np_3 and np_1 - i.e., when the [Secretary] has required the disclosure, then the dominant normative proposition prevails on the dominated one.

Step 4. Identify potential realisations of normative propositions

Why. Normative propositions specify to addressed subjects actions to be

done (*behavioural actions*, according to the terminology used in (Sartor, 2006)), or results to be achieved (*productive actions*). As they are specified in legal texts, actions recall goals (or tasks, or other intentional concepts); however, actions and goals differ as (i) goals are *wanted by* actors, whereas actions are *specified to* actors and can be in contrast with their goals; and (ii) goals are *local* to a certain actor - i.e., they exist only if the actor has the *ability* to fulfil them - while actions are *global*, referring to a whole class of actors; for example, law may address health care organisations, regardless whether they are commercial or no-profit, but when compliance is established, the actual nature of the complying actor gains importance; for the same reason, actions are an *abstract* characterisation of a whole set of potential actions as conceived by the legislator. It becomes so necessary to switch form the point of view of the legislator to the point to view of the actor.

How. Given a normative proposition *np* that specifies an action A_{np}, a goal g is searched for the addressed actor, such that: (i) it is *acceptable* by the actor, with respect to its other goals and preferences; (ii) the actor is known to have, or expected to have, the ability to fulfil the goal; and (iii) there is at least one behaviour that the actor can perform to achieve the goal, which makes *np* fulfilled. In the ideal case, every behaviour that achieves g also fulfils NP; we write in this case $g \subseteq np$. Otherwise, g is decomposed to further restrict the range of behaviours, until the above condition is ensured. If it is not possible to exclude that $g \nsubseteq np$, then g is considered *risky* and the "Identify legal risks" step is performed.

Result. If found, g (also if it is risky) is put in realisation relation with *np* and becomes the top *compliance goal* for *np*.

Running example. One of the assumptions made for building the diagram of Figure 7.2 is that the requirements analysis concerns only the treatment of electronic data. As such, from the point of view of the hospital the non-disclosure duty (normative proposition [Don't disclose PHI]) is fulfilled if the PHI is not dis-

closed *electronically*. In the diagram, for the hospital a well-designed set of policies for accessing electronic data (goal [policy-based data access]) is enough to have the duty realised. This may be true, or may be too simple-minded, or may need further refinement of the goal. This is part of the modelling activity.

Step 5. Operationalise compliance goals

Why. Identifying the root compliance goals is not enough to generate a compliant set of requirements. Goals are states of affairs desired by stakeholders, but in order to become operative - i.e., in order to be implementable and executable by a system - they need to be operationalised. In other words, it is necessary to find a task assignment such that, once executed, allows the achievement of the compliance goals.

How. This step follows strictly standard goal-oriented requirements engineering methodologies, such as Tropos. Goals are refined into sub-goals, until the operational tasks can be identified.

Result. The result of this step is a set of tasks, in means-end relation with leaf compliance goal.

Running example. In Figure 7.2, the normative proposition [Don't disclose PHI] is realised by the goal [Policy-based data access], which in turn is operationalised by means of the tasks [Assign login to doctors and call center], [Monitor electronic transactions], and [Prevent PHI data printing].

Step 6. Identify proof artefacts

Why. During the requirements analysis we aim at providing evidence of *intentional compliance*, which is the assignment of responsibilities to actor such that, if the actor fulfil their goal, then compliance is achieved. *Actual* compliance will be achieved only by the running system. However, in a stronger meaning, compliance can be established only *ex-post* by the judge, and at runtime this will possible only by providing those documents that will prove the compliance.

How. After a compliance goal is identified, it can be refined into sub-goals. The criterion for deciding the decomposition consists in the capability to identify a proof resource. If a resource can be identified, then such a resource is added to the model; otherwise, the goal is decomposed. The refinement process ends when a proof resource can be identified for every leaf goal of the decomposition tree.

Result. The result of this step is a set of resources that, at run-time, will be able to prove the achievement of certain goals or the execution of certain tasks.

Running example. In Figure 7.2, the task [Assign login to doctors and call center] can be proved to keep the PHI not disclosed by means of two auditing resources: the [Users DB] and the [Transactions report].

6.4 Compliance verification

The process equilibrium of Equation (6.1) can be altered for two reasons: because R changes in a non-compliant way, or because L changes, and makes compliance-risky also the already-checked requirements. In both cases, requirements are no more compliant, so the need arises to verify their compliance. Compliance verification is a process that takes as input a set of existing requirements and a set of laws, and establishes whether they are aligned or not.

Legal alternatives As stated in Chapter 5, compliance verification firstly depends on the legal alternative an actor wants to comply with. Legal alternatives are created uniquely by those normative propositions that carry discretional rights — i.e., rights that can be exercised by a free choice of the addressed actor. Such rights are: privileges, claims, powers and immunities. Duties (except for applicability concerns) do not give any choice and must be realised by the addressee, as well as liabilities, and disabilities and No-claims simply don't allow the addressee to do anything. Consequently, given a set of normative

propositions, and a set of dominance relations between them, for each normative proposition, whose right type is a privilege, claim, power, immunity, a legal alternative is added.

Compliance relations After having selected a legal alternative for compliance, the compliance verification consists in ensuring that every normative proposition in the legal alternative has at least one realising goal. In Chapter 5, we have introduce the abstract AND-relations and OR-relations. Such relations can be mapped to *i** relations.

And-decompositions, as well as Or-decompositions into sub-goals, are mapped to AND-relations. This is because, by deciding that a certain goal is a sub-goal of another one, the semantics of that specific sub-goal is assumed to define, by design, a subset of the parent goal. Similarly, means-end relations between a task and a goal are AND-relations, because we assume that the task is executed within the boundaries of the goal.

Differently, contributions are OR-relations, because the contributor can be totally disjoint with respect to the contributed goal. Also, if a goal is achieved through a dependency from another actor, the dependency introduces an OR-relation between the depending goal (i.e., the source of the *why* link) and the goal that belong to the dependee.

Analysis process The compliance verification process consists in the traversal of a *Nòmos* model to verify that the compliance conditions introduced in Chapter 5 hold. The process, as depicted in Algorithm 3, takes as input three arguments: the first is a set of requirements R represented in the *Nòmos* model as goals; the second is the law L whose compliance has to be checked, and consists in a set of normative propositions; Finally, the third argument, C is the set of realisations that bind elements of R to elements of L. The process returns the first compliance issue found, if any; otherwise, it returns $true$.

The process relies on the definition of legal alternative to simplify the analysis. The idea is that, since R, L and C come form a *Nòmos* model, by design the relations in C should contain the compliance information added by the analyst during the *generateNextComplianceSolution*. So, assuming that such method has been used sometime before this check, if requirements are compliant in C there should be a number of realisation relations suitable to match at least one of the legal alternatives allowed on L. In step 1, such legal alternative is searched and assigned to L^{choice}. Now, L^{choice} does not necessarily contain a *legal* alternative: it contains the *intentional* alternative — i.e., the one chosen by the analyst.

Afterwards, the intentional alternative is compared in step 2 against the set of valid legal alternatives. If they match, we can say that the chosen alternative is also legal. Otherwise, if no valid permutation on L is found, the function returns with an error (step 3).

If a valid legal alternative is found, the compliance verification process is reduced to check that, for every normative proposition within L^{choice} the condition described in Chapter 5 holds. So in step 4 a cycle simply iterates over each normative proposition.

The compliance analysis checks in step 5 that a valid operationalisation exists for the compliance goal. In other words, the function here follows the decomposition of each goal that realises a normative proposition and verifies that there are tasks in means-end relation with the leaf compliance goals.

Finally, the algorithm checks in step 6 the existence of the auditability condition. As in the previous step, the algorithm simply checks that, for each compliance task, at least one resource is attached to the task, such that when the task is executed, the resource is affected.

If the algorithm succeeds, it returns nothing. Otherwise, it returns a report of the found issues. If the issue consists in matching a legal alternative, no further processing is made.

Algorithm:checkRequirementsCompliance

Input: $R = \{g_1, ..., g_n\}$, $L = \{np_1, ..., np_m\}$, $C = \{\rightarrow_1, ..., \rightarrow_c\}$

```
1 L^choice = deriveSelectedLegalAlternative;
2 if ¬L^choice ∈ alternativesOf(L) then
3    return [violation];
4 foreach np ∈ L^choice do
5    check( [np is operationalisable] );
6    check( [np is auditable] );
7 end
8 return true;
```

Algorithm 3: A sketch of the algorithm followed by the analyst for arguing about intentional compliance.

6.5 Process outcome

Figure 6.3 depicts the process described in Algorithm 1. The process is basically incremental: the modelling activities in $generateNextComplianceAlternative$ always returns an addition to the input requirements, under the constrains specified in Chapter 5, so we can say that $R', K \models R, L$. The $discoverLaws$ adds requirements without ensuring compliance, however, they are aligned again by the $checkRequirementsCompliance$. Viceversa, the $discorverLaws$ function changes the set of normative proposition L. So we can say that, as long as that function returns $\emptyset$, the process function of Equation (6.1) holds. This happens for example when compliance is searched for a specific law, and all the applicable normative propositions are in L from the first iteration. Otherwise, when a new law comes into the scene, not taken into consideration before, compliance has to be checked again.

6.5.1 Vulnerability analysis

As last, worth saying that a *Nòmos* model correctly engineered acts as a starting point for other types of analysis. We want to highlight here two of them, which

are particularly important.

Identification of security issues To achieve goals that are otherwise not in their capabilities, or to achieve them in a better way, actors typically delegate to each other goals and tasks. When an actor delegates a strategic goal, a weakness arises, which consists in the possibility that the delegatee does not fulfil the delegated goal. If the delegated goal is intended to realise a legal prescription, this weakness becomes critical, because it can generate a non-compliance situation. As such, law is often the source of the security requisites that a certain requirements model has to meet.

Specifically, three cases exist for delegation:

1. Compliance goals. Goals that are the realisation of a normative proposition, or belong to the decomposition tree of another goal that in turn is the realisation of a normative proposition, can be delegated to other actors only under specific authorisation.

2. Proof resources. We have highlighted how the identification of proof resources is important for compliance purposes. The usage of proof resources by other actors must then be permitted by the resource owner.

3. Strategic-only goals. Goals that have no impact on the realisation of normative propositions, can be safely delegated to other actors without need to authorise it.

The result of this activity is a network of delegations and permissions that maintain the legal prescriptions across the dependencies chains.

Running example Going back to Figure 7.2, the hospital delegates to the doctors the PHI disclosure to the patients. However, the hospital is the subject responsible towards the patient to disclose its PHI. This means that a vulnerability exists, because if the doctor does not fulfil its goal then the hospital is not compliant. For this reason, using the security-enhanced *i** primitives offered by

SecureTropos, in the model we have to reinforce the delegation by specifying the trust conditions between the hospital and the doctor (refer to (Giorgini et al., 2005) for a deeper analysis on trust, delegation and permission).

Identification of legal risks At organisational level, risks have a negative impact on the capability of the organisation to achieve its goals. Using *i**, risks can be treated with risk management techniques that allow to minimise them (Asnar and Giorgini, 2006). For organisations, law is also a source of a particular type of risk, or *legal risk*, which "includes, but is not limited to, exposure to fines, penalties, or punitive damages resulting from supervisory actions, as well as private settlements" (bas, 2006). Legal risk comes from the fact that compliance decisions may be wrong, incomplete or inaccurate. In our framework, the "realisation" relation that establishes the link between a normative proposition and a goal can't prevent legal risks to arise: for example, a wrong interpretation of a law fragment may lead to a bad definition of the compliance goal. Legal risk can't be completely eliminated. However, the corresponding risk can be made explicit for further treatment.

Specifically, when a goal is defined as the realisation of a certain normative proposition, a search is made in the abilities of the actor, with the purpose of finding other intentional elements of its behaviour that can generate a risk. Given a certain risk threshold ϵ, if the subjective evaluation of the generated risk is greater than ϵ, then the risky element has to be modelled.

If some of the requirements may interfere with the compliance goals, then the requirements set is changed accordingly and the new set is returned. If no risky goals have been identified, the requirements set is not changed.

Running example In Figure 7.2, we have depicted the need for the hospital to have a hard copy of certain data: it's the goal [Print data] (assigned to the hospital for sake of compactness). If doctors achieve this goal to print patients PHI,

this may prevent the use of a policy-based data access to succeed in the non-disclosure of PHI. This is represented as a negative contribution between [Print data] and [Policy-based data access]. To solve this problem, a new goal is added: [Prevent PHI data printing], which can limit the danger of data printing. (Notice that here we don't further investigate how PHI printing prevention can actually be achieved.)

6.6 Modelling Tool

In the next Chapter, we will present the results of the application of the *Nòmos* framework to three case studies. Such experiences have been performed with the support of a tool developed specifically for this purpose. The GUI of the tool is depicted in Figure 6.4. It presents a right panel for visually editing models, and a left panel for browsing the model, its elements ant its environment. The tool has been developed mainly for modelling purposes, but it relies on a meta-model and can therefore be extended for performing queries and analysis on the models. Although it has not yet published due to stability concerns, it can already be freely downloaded from `http://brenta.disi.unitn.it/~asiena/nomoswiki/`.

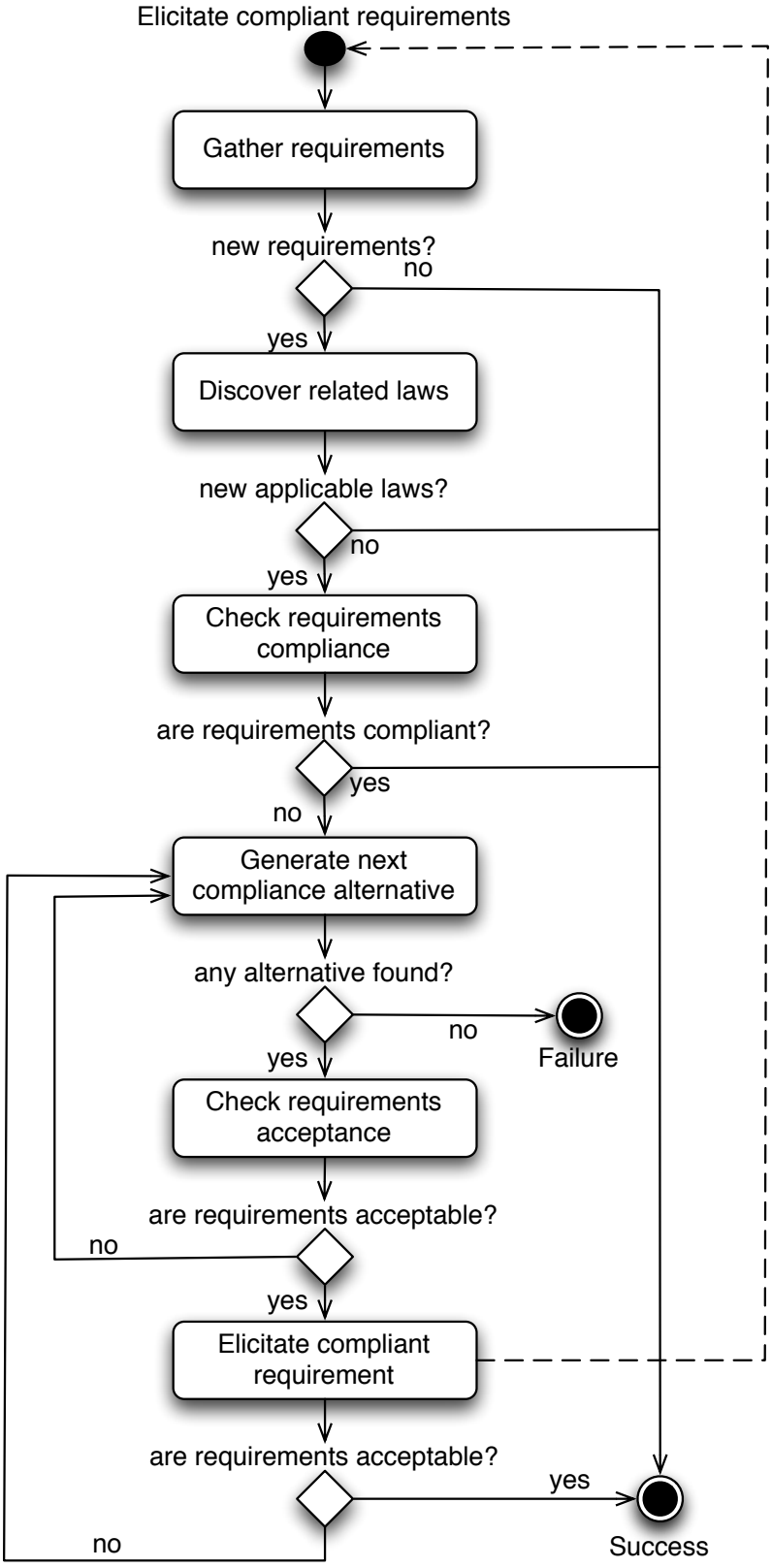


Figure 6.3: Overall law-compliant requirements elicitation process.

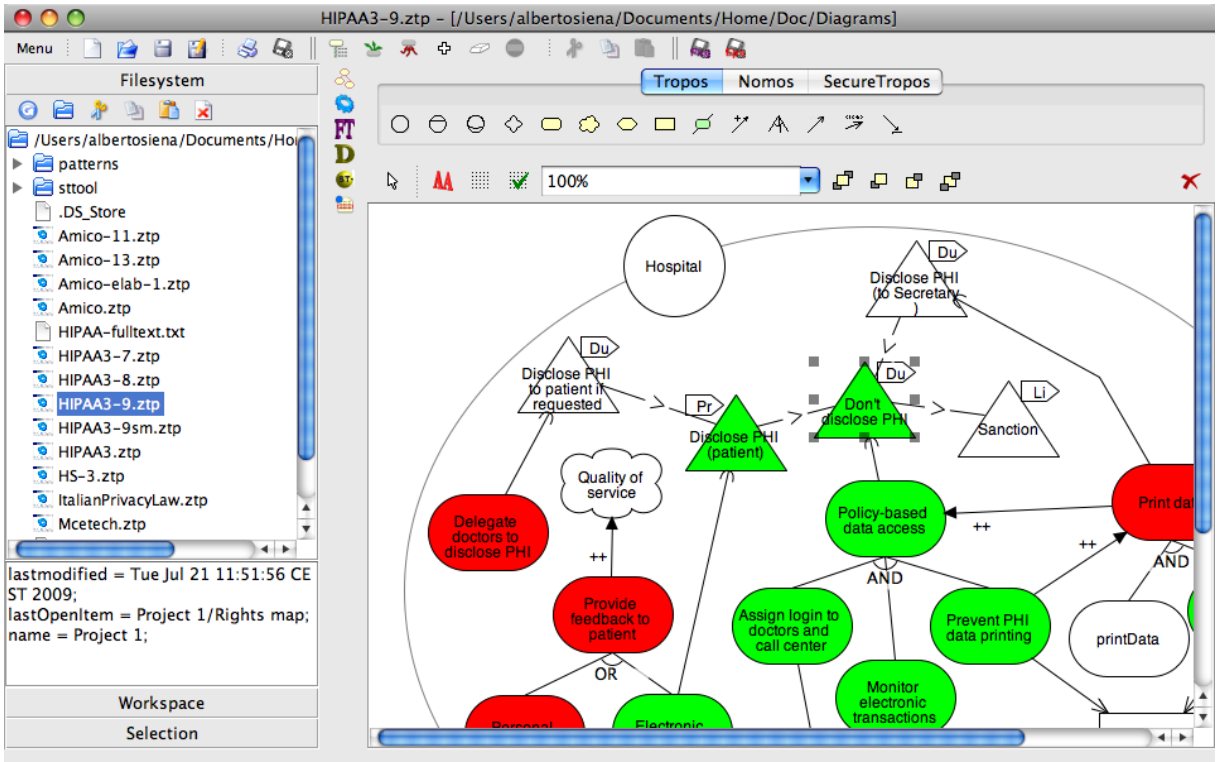


Figure 6.4: A screenshot of the *Nòmos* modelling tool.

Chapter 7

Application

7.1 Building requirements compliant by construction

H.I.P.A.A. The *Nòmos* framework, has been firstly developed and tested on a case case study. The case study uses a sandbox scenario to generate requirements. The scenario is targeted on a U.S. hospital that wants to set up a new information system to manage service reservations. the hospital wants the new information system to be able to quickly retrieve the availability of rooms and devices in the hospitals, and optimise the reservation according to the needs of the patients and doctors. Moreover, it wants to increase the accessibility to the system and to reduce the durations of the processes for both the patients and the doctors, and for this reason, it is evaluating the impact of Internet. At the same time, it needs to be strongly compliant with the U.S. Health Insurance Portability and Accountability Act (HIPAA).

HIPAA was enacted in the USA in 1996. HIPAA includes two titles. Title 1 protects health insurance coverage for workers and Title 2 enforces the establishment of national standards, health insurance plans and employees. In addition, Title 2 addresses the privacy and security of health data. In our work, we consider Articles §164.502 and §164.314 of HIPAA.

Law modelling When facing a law, the first step consists in producing a model of its prescriptions, or at least a model of those prescriptions that are applicable to the domain. To do this it is necessary to access the legal sources and extract information from them. Specifically, it is necessary to extract the normative propositions from the natural language text.

For example, Article §164.502 says that: *(a) A CE may not use or disclose PHI, except as permitted or required by this subpart [...] (1) A covered entity is permitted to use or disclose PHI [...] (i) To the individual; (2) A CE is required to disclose PHI: (i) To an individual, when requested [...]; and (ii) When required by the Secretary.*

For convenience, we restate the law by splitting it down into smaller slices; for example, for §164.502 we have:

1. A CE may not use or disclose PHI, except in the subsequent cases.
2. A CE is permitted to use or disclose PHI to the individual.
3. A CE is required to disclose PHI to the individual.
4. A CE is required to disclose PHI when required by the Secretary.

Similarly, form Article §164.314 we have:

5. A CE may not work if it knows a practice of the business associate (BA) that constituted a material breach or violation of the BA's obligation.
6. A CE that knows of this infringement is still compliant is it ends the BA's violation.
7. A CE that can't end the violation is still compliant if it terminates the contract.
8. A CE that can't terminate the contract is still compliant if it reports the violation to the Secretary.
9. A BA is required report every security incident to the CE.

Out of these law slices, it is possible to derive the normative propositions that compose the law fragment. The identified normative propositions are summarised in Table 7.1. The first row of the table contains a reference to the source text. "Id" is a unique identifier of the normative proposition. "Holder"

and "counterparty" are the involved actors. "Action characterisation" is the description of the action specified in the normative proposition. To identify the normative propositions, prescribing words have been mapped in the right specifiers; e.g., "is permitted" has been mapped into a *privilege*, "is required" has been mapped into a *duty*, and so on. The name of the subjects are extracted by either using an explicit mention made by the law (e.g., "a CE is not in compliance if..."); or, when no subject has been clearly detected, by identifying who carries the interest that the law is furthering. Finally, the priority column establishes the dominance relationships between normative propositions. For example, an exception like the one in the first sentence ("A CE may not [...] except [...]") has been mapped into a dominance of every other proposition of §164.502 over NP1. Figure 7.1 depicts a diagram of §164.314 and §164.502. The diagram is a graphical representation of the normative propositions listed in Table 7.1[1].

Compliance analysis The process described in Algorithm 3 requires as input a model of law and a model of stakeholders goals. For our purposes, the goal model is built using the classical *i** approach (Yu, 1996). Figure 7.2 depicts a diagram integrating the law model and the stakeholders goals. The diagram has been built by merging an *i** model of the hospital with the model of law. For building the diagram, we assume that (i) the hospital is the CE of HIPAA; (ii) the operations of the call center have been outsourced, so that the Call Center company is now a business associate of the hospital; and (iii) the hospital only holds data of patients, so Patient is the individual addressed in the law. The diagram addresses only a part of HIPAA and a part of the possible goal model. Following the process described in Algorithm 3, we can see in the model that there are four duties: (1) [Don't disclose PHI] (from [Patient]), which is fulfilled by

[1]Notice that in the diagram we added a "Sanction" liability that the CE may suffer if not compliant with other duties. In general this information may be retrieved from the source or related documents, as in this case

Src §164.-	Id	Right type	Holder	Counterparty	Action characterisation	Dominances
§502a	NP1	claimDuty	Individual	CE	not DiscosePHI	-
§502a1i	NP2	privilegeNoclaim	CE	Individual	DisclosePHI	NP1
§502a2i	NP3	claimDuty	Individual	CE	DisclosePHI	NP1,NP2
§502a2ii	NP4	powerLiability	Secretary	CE	DisclosePHI	NP1
§314a1ii	NP5	claimDuty	CE	BA	no KnownViolations	NP6,NP7,NP8
§314a1ii	NP6	immunityDisability	CE	Authority	EndViolation	NP7,NP8
§314a1iiA	NP7	immunityDisability	CE	Authority	TerminateContract	NP8
§314a1iiB	NP8	immunityDisability	CE	Secretary	ReportTheProblem	-
§314a2iiC	NP9	claimDuty	CE	BA	ReportSecurityLacks	-

Table 7.1: Some Normative Propositions identified in §164.314 and §164.502.

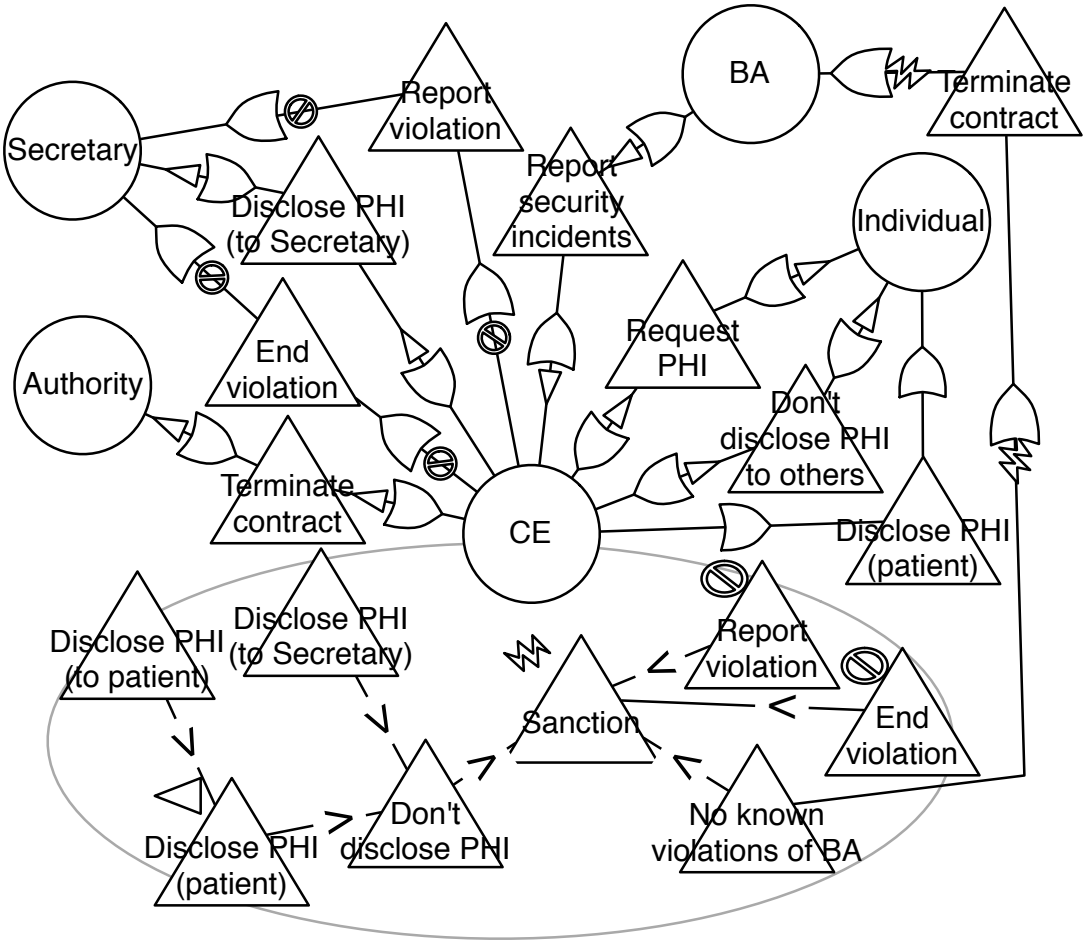


Figure 7.1: A model of law.

the decision to implement a policy-based data access; (2) the claim of [Patient] to [Request PHI], delegated to the doctors that interact with the patient; (3) the duty, towards the [Secretary], to [Disclose PHI], if requested, and this is done by the administration (due to lack of space we can't model the complete organisational structure); (4) the duty, towards the authority, of not working knowing the non-compliance of a BA; for this purpose, the same law assign to the CE the power of terminating the contract with the BA, if needed. Since the call center has been outsourced, the call center company is also responsible for preserving the patients PHI. Vice versa, doctors working in the hospital have the responsibility to care patients and need to access their PHI. Therefore, the hospital has a sim-

ple strategic dependency on the doctors for not disclosing PHI. A [Data monitor] actor is added to the model, and it is delegated to monitor data usage and to warn the hospital in case of access violations. There are **alternative ways** to be compliant. In Figure 7.2 a piece of the strategy is also depicted. Specifically, to improve the [Quality of service], the Hospital may want to [Provide feedback to patient]. There are two ways to do this. In the first alternative, the Hospital may decide a policy such that each patient is flanked by an employee that support him - this is represented with the goal [Personal assistant]. In the second alternative, an account is provided to the patients, to allow them to access their PHI electronically. In this specific case, the Hospital can omit to comply with NP1 - i.e., [Don't disclose PHI]. Both alternatives are law compliant according to the rules defined in Chapter 5.

Finally, worth noticing that with the combined use of normative propositions and "realise" relations, it is possible to assign to each normative proposition information about where does the normative proposition come form in terms of law's fragment, thus allowing **traceability** between goals and law. For example, the delegation to doctors to disclose PHI derives form the need to comply with paragraph §164.314(a)(2)(i), because, as shown in Table 7.1, the duty of disclosing PHI to the patient (NP9 in the table) has been extracted form that paragraph.

7.2 Reconciling legal and strategic requirements

The A.M.I.C.O. project Amico[2] is an industrial research and development (R&D) project in the health care domain. Overall, the project lasted 18 months of work, with around 25 people working on it, including project manager, analysts, software architect, programmers and researchers. The Amico project is intended to define the architecture for an integrated services system, aiming at increas-

[2] Assistenza Multilivello Integrata e Cura Ovunque

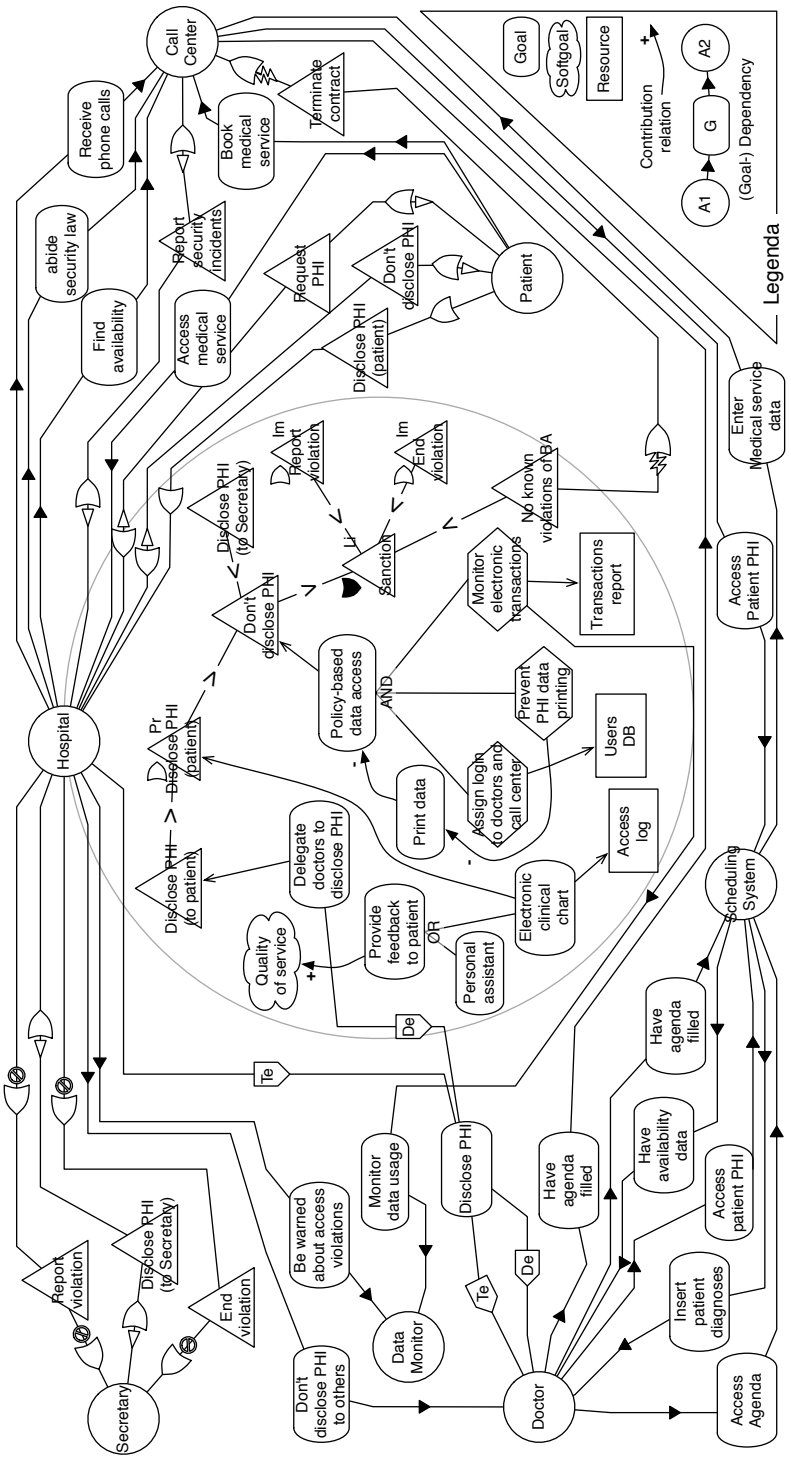


Figure 7.2: A goal-oriented model of law-compliant requirements.

ing the possibilities of self-supporting life for elder or disabled people in their own home. The project is focused on the realisation of an Electronic Patient Record (EPR) for storing social and health information to be used in health care. Information is stored and accessed independently by the subjects that operate in the health care system: social workers, doctors, social cooperatives, relatives. The EPR, accessed via web, allows for a collaboration among the subjects, for improving health care and having social and health information, as well as economic and managerial data. Technological devices are applied in patients' home and according to the patient's needs, to both support him and to monitor his health conditions. Data produced by the devices is integrated with patients' health history, and with the human activity of health and social workers to create a health centre, able to provide fast assistance actions if needed, improve life quality of patients, reduce unneeded hospitalisations, and rationalise costs.

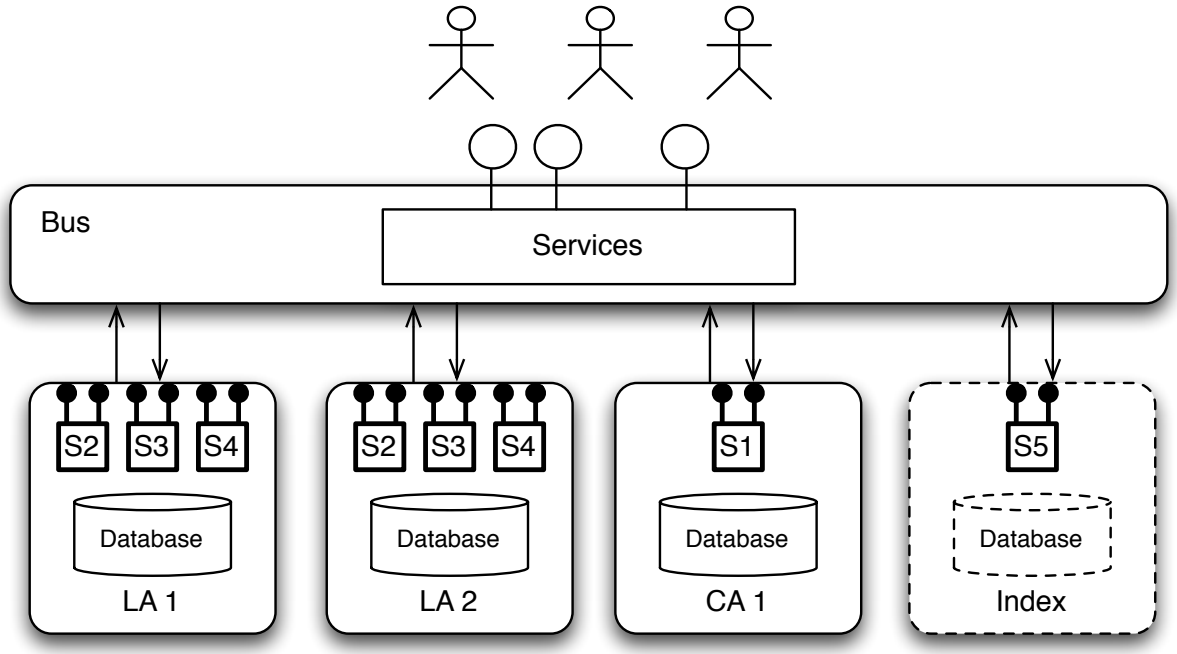


Figure 7.3: The demo scenario for the Amico's system architecture.

Amico has been conceived as a network of interconnected systems, as de-

picted in Figure 7.3. Nodes of the network are mainly the health care facilities with their information systems, called *Local Authorities* (LA). The Local Authorities run their own databases, and provide services such as data search and retrieval to other members of the network. Another type of node of the network are the so-called *Certificate Authorities* (CA). Certificate Authorities are the reference actors for Local Authorities. Certificate Authorities keep a copy of those data that have been verified and can be trusted. So, the data that the Local Authorities retrieves form the Certificate Authorities are called "clean". On the contrary, data retrieved from other Local Authorities, are not verified and are called "dirty". An Index node manages the list of members of the network. Through the Index, a Local or Certificate Authority can know of others Authorities registered system-wide.

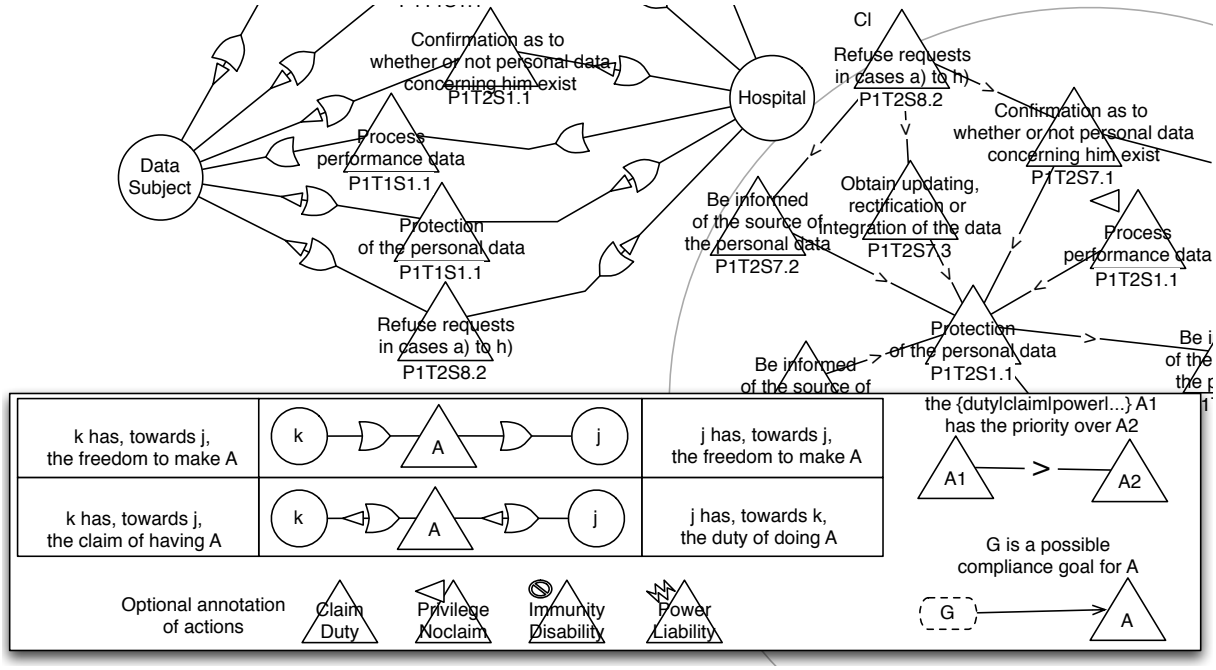


Figure 7.4: The Nomos modelling language: visual representation of the Italian Personal Data Protection Code.

In a workpackage of the Amico project, we were called to refine the analysis of gathered requirements from the point of view of legal compliance. Specif-

ically, the law under analysis was the Italian Personal Data Protection Code D.Lgs. n. 196/2003 (depicted in Figure 7.4), limitedly to Part I, Title II (Data Subject's Rights) of the law. To the workpackage have participated 8 people: 3 analysts, 1 industry partner, 1 software architect, 2 designers, 1 programmer.

The analysis of the compliance requirements was grounded on the definition of a *demo scenario*. The scenario concerns the management of the check-in procedures. A patient may turn to various healthcare facilities (actors, according to *i**) to use their health cares. On reception, the facility needs to know the history clinic of the patient, in order to select the most appropriate cure. The clinical data can be in the local database of the accessed facility; or, it can be distributed somewhere across the Amico network; or, it can be completely absent. Through the Amico system, it should be possible to access an integrated EPR, which collects every useful information available for the patient wherever in the network; alternatively, it should be possible to create the EPR from scratch, and broadcast it through the network.

With regard to the described scenario, five services are provided by the nodes of the network. The nodes have been labelled S1 to S5, and are depicted in Figure 7.3. Services S1, S2 and S3 are provided by the Local Authority. Service S4 is provided by the Certificate Authority. Service S5 is provided by the Index. The Local authority accesses S1 to S5, regardless whether the services are provided by itself, or by another node. The Certificate Authority accesses S1 and S5. The services do the following:

- S1: Each local authority provides access to service S1 which is responsible for accessing the underlying system and provide service such as data search locally. It is also possible to insert new information or update information using this service.

- S2: Each local authority provides access to service S2 which is responsible for accessing Certificate Authority and provides service of data search

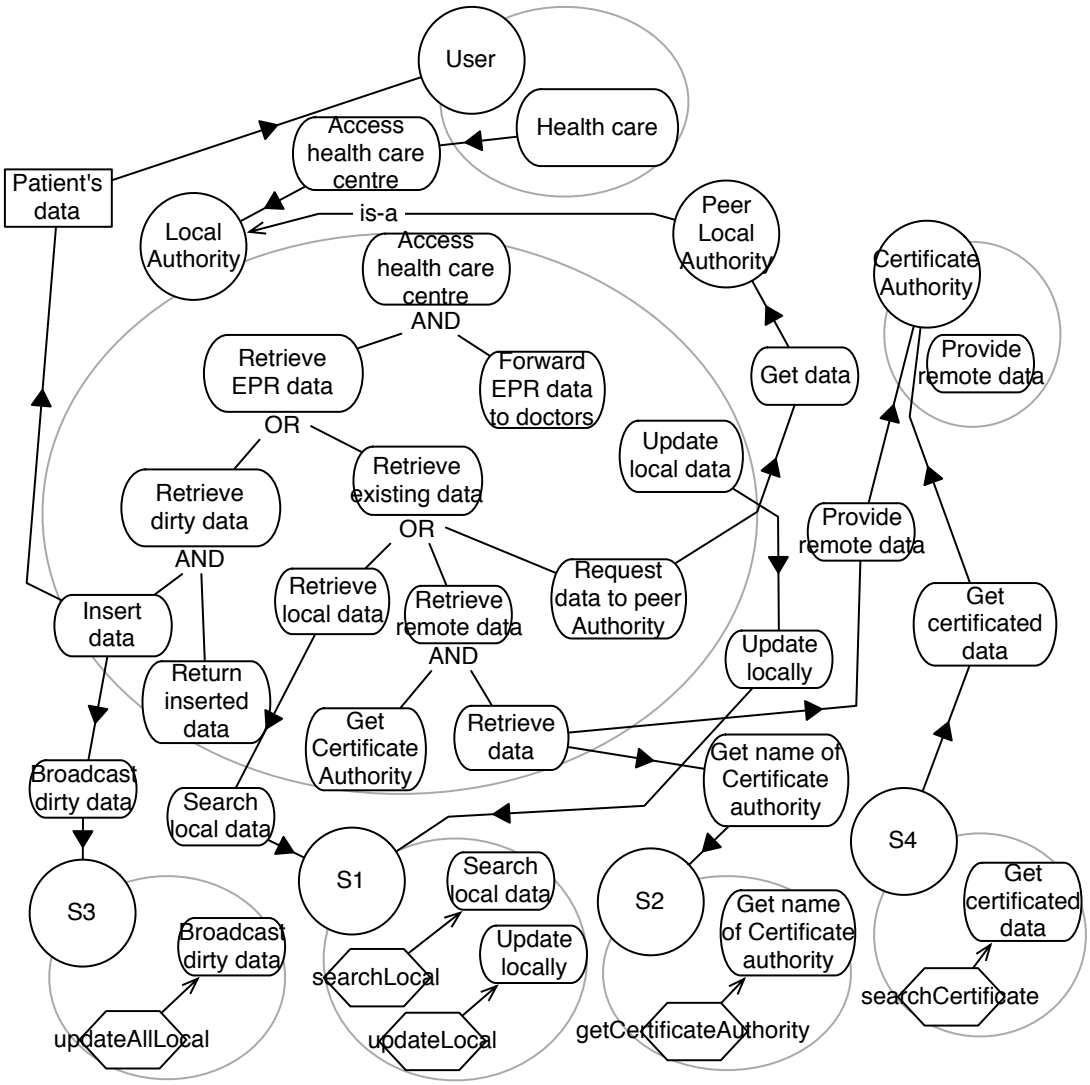


Figure 7.5: A goal model for the demo scenario of the Amico project.

remotely. It is not possible to update information in this case as it read only service. Once information is available from the Certificate Authority it's possible to update or create new information locally by invoking S1.

- S3: Each local authority provides services for updating other local authorities to maintain integrity of data between local systems; i.e. to broadcast "dirty" information (information still not verified) to all local authorities.

- S4: Each Certificate Authority is responsible for providing certified data

search — i.e., it provides a read-only access to some verified data.

- S5: This is a service accessible from anywhere and it returns information of corresponding Certificate Authority.

We modelled the requirements of the demo scenario by means of goals. In goal models, goals express the *why* of requirements choices. Goals are decomposed into sub-goals and operationalised by means of plans. Plans, in turn, may need resources to be executed. In the Amico project, we used *i** (which *Nòmos* is based on) to create goal models representing the rationale behind the demo scenario. Figure 7.5 represents such rationale, limitedly to the [Local Authority] actor. When a patient ([User]) accesses a health care centre, at the check-in the EPR of the patient has to be retrieved from the system. In the health care centre accessed by the patient, the system (a [Local Authority]) executes a query on the local database, and the [S1] service furnishes such data. If the data is not found in the local database, the [Local Authority] forwards the request to the [S2] service, which returns the name of the reference [Certificate Authority]. The Authority is queried to have certified data. But [Certificate Authority] can also be unable to provide the requested data. In this case, the local authority contacts another Local Authority (the actor [Peer Local Authority] in the diagram), which in turn executes a local search or queries its own reference Certificate Authority. If the searched data don't exist in the system, the Local Authority proceeds inserting it, and marking it as "dirty". In this case, after the data insertion, the Local Authority invokes the [S3] service, which broadcasts the data to the whole system. When the broadcast notification is received, each Local Authority updates its local database.

Compliance verification and construction We moved from the analysis of the Italian Privacy Code, which lays down many prescriptions concerning the processing of personal data (in particular, sensitive data) of patients. We modelled the

relevant fragments of the law through the *Nòmos* language, as in Figure 7.4. Afterwards, those goals have been identified, which could serve for achieving compliance with that particular law fragment, and were associated with the corresponding normative proposition. If no appropriate goals were identified, a compliance lack was considered to be found.

For example, the Italian privacy law requires the data owner's authorisation for processing personal data. In Figure 7.4, this is depicted by means of the normative proposition [Confirmation as to whether or not personal data concerning him exist]), extracted from article 7.1. The normative proposition is modelled as a claim of the patient, held towards the Local Authority, which has therefore a corresponding duty. Figure 7.5 shows that in the above described behaviour of the Local Authority, no answer exists to such a legal prescription. In the diagram, at least 2 points have been identified, in which a specific behaviour of Local Authority has to be elaborated for legal compliance. The first one concerns the insertion of the data into the local database, and subsequent broadcast to the system. In this case, before the broadcast is executed, it is necessary to obtain the patient's authorisation (goal [Ask user authorization]), and to add such information in the broadcast message. The second case concerns the reception of the broadcast system by a Local Authority. In this case, before updating the local data with the received one, the Local Authority must verify that in the broadcast message the authorisation to data processing is declared (task [Verify user authorization]).

This has been done for every normative proposition considered relevant for the demo scenario. The resulting models (partially depicted in Figure 7.6) contained 10 normative propositions relevant for the described demo scenario. This approach allowed for assigning to domain's actors a set of goals, which represent their responsibility in order to achieve compliance.

Auditability Within the *Nòmos* framework, we distinguish goals with respect to their role in achieving compliance. We define *strategic goals* those goals

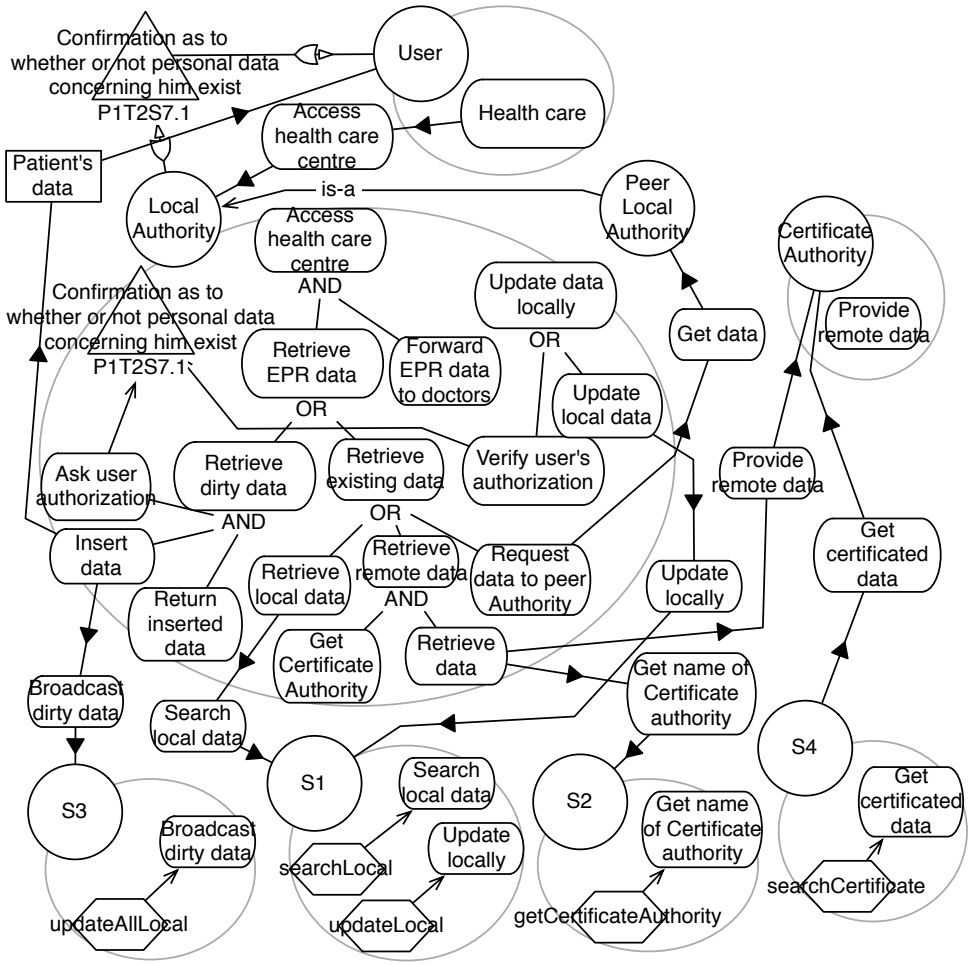


Figure 7.6: A *Nòmos* diagram that represents the Amico scenario under legal constraints.

that come from stakeholders and represent needs of the stakeholders. We define *compliance goals* those goals that have been developed to cope with legal prescriptions. For example, in Figure 7.6, the goal [Update data locally] is a strategic goal, because it is only due to the reason-to-be of the owning actor; viceversa, [Ask user authorization] is a compliance goal, because it is due to the need of complying with the [Confirmation as to whether or not personal data concerning him exist] claim of the user. The identification of compliance goals, and specifically the identification of *missing* compliance goals, was actually the objective of our analysis.

At this point, we had models containing a set of compliance goals. Consider-

ing the needs of the industrial partner, we asked ourselves how to give strength to the decisions made for reaching compliance. We considered the system and the running processes based on the described requirements. Design-time compliance leaves open issues in the actual development of the system. For example, mistakes in the architectural design compromise the quality of software; or, even if the software is well designed, software programs could be bugged and behave differently from what expected; and in any case, people working with the software once deployed could behave differently from what is assigned to them according to their roles, thus creating run-time compliance issues. Intuitively, developing and deploying the system (or a prototype of it) in order to verify compliance issues is not feasible for such a large and distributed system. So the problem has arisen, of how to provide evidence to the industrial partners, of the correctness of chosen compliance goals. For this reason, we have exploited the notion of auditability.

Organisations (healthcare as well as others, such as banks), face the problem of auditing the execution of procedures — i.e., controlling internal log data, generated during processes execution. The idea is that, if an activity or a whole process is not executed or is executed incorrectly, this is reflected in the log data, so that through analysis of the log data, it is possible to monitor the execution of processes and detect problems. One of the ideas underlying *Nòmos*, is that log data, upon which audits rely, have to be correctly designed. Thus, designing for auditability means deciding which log data have to be produced, by which process, when, and so on. For the information systems supporting the processes, it's important to specify requirements of compliance auditability together with other requirements. Consequently, compliance auditability has to be conceived during the requirements analysis.

In order to assess auditability we associate data log resources to goals. More properly, the resources are associated to the plans intended to fulfil compliance goals. Or, as a shortcut, the plans are omitted and resources are associated

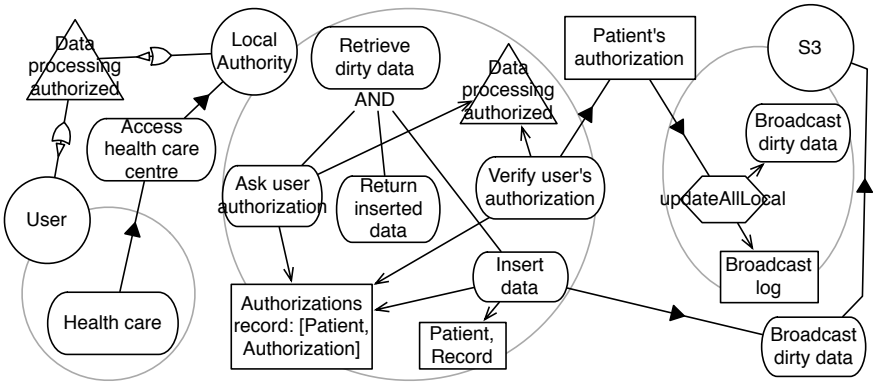


Figure 7.7: The rationale for an auditable process.

directly to goals. In any case when plans (either explicitly modelled or not) are executed to achieve a goal, they use the associated resource. The idea, conceived in the *Nòmos* framework, is that such resources can be the key to monitor, at run-time, the execution of the processes. When a resource - such as a database - is used, its state is affected and the state change can be recorded. Or, even the simple access to a resource can be recorded.

Upon this idea, 2 analysts have undertaken a modelling session. An excerpt of the results is depicted in Figure 7.7. The ultimate purpose was to revise existing models, searching for compliance goals. Once found, compliance goals have been elaborated to be made auditable. For example, the [Local Authority] has the goal [Ask user authorisation], which has been developed to comply with the duty to have such authorisation from the user before processing his data, can't be proved at run-time. Even if the authorisation is requested, if a legal controversy arises, the developed models don't inform on how to prove that this request has been made. To deal with this situation, we added the [Authorisation record] to the model, and associated it to the [Ask user authorisation] goal. This way, we are saying that, whenever the goal is achieved, this is recorded in the authorisation record, where we store the name of the patient, who gave the authorisation, and the authorisation itself (if the authorisation has been provided electronically; otherwise, the ID of the archived copy of it).

On the other hand, the [Local Authority] receives broadcasted messages when other local authorities commit dirty data in their databases. In this case, the goal to [Verify user's authorization] had been added, to avoid that failures of other local authorities in getting the authorisation from the user may lead to compliance issues. Again, in this case, from the model it's not possible to specify how the achievement of this goal can be monitored. For this reason, after having verified the presence of the authorisation information within the broadcasted message, the [Local Authority] stores such information in another log, which informs on which authorisations have been received and from which peer. Additionally, we enriched the model with a resource dependency form the [Local Authority] to the service [S3] (of the peer authority), to indicate that the authorisation has to be provided by that service. In turn, this raises another problem, of where that information has tom come from. This is not explicit in the model, so it's not possible to associate the behaviour of [S3] with its auditability resources. So finally, we associated the [Insert data] goal to its resources: the [Authorizations record], where it takes the authorisation information from; and the [Patients data], where the actual data is stored. This way, patients' data is associated to the authorisation to process them.

Results The proposed approach has led to some important results. The Software Requirements Specification (SRS) document has been integrated to reflect the compliance solutions found, and to support the auditability of the compliance solutions.

Table 7.2 reports the most important additions introduced by the *Nòmos* analysis. In the table, the first column reports the law or law fragment, which the requirement has been developed for; the second column presents the description of the requirements, gathered or induced by the law fragment; the third column indicates whether the requirement is intended to be audited at run-time.

As the first column shows, not all of the articles in the selected law slice

Table 7.2: An excerpt of the Software Requirements Specification document.

Law article	Requirement	Audit
Art. 7.1	The Local Authority registers users' authorisations	
	The Local Authority writes the User's in the Authorisations base	✓
	The Local Authority inserts the data into the local DB	✓
Art. 7.2e	The Local Authority verifies the entrance of new peers	
	The Local Authority maintains the list of verified peers	✓
	S1 gets the list of verified peers from the Local Authority	
Art. 7.3a, 7.3b	The Local Authority writes data modifications to log	✓
Art. 9.4	The Local Authority identifies the patient by means of identity card	
	The Local Authority records patients' ID card number	✓
...	...	
Art. 157.1	The Local Authority produces a report with the collected data to the Garante	✓

were addressed. This is due to the demo scenario, which concerned only on search, addition and update of data: so, law articles not impacting on these functionalities were not addressed. On the contrary, one article - the 157.1 - has been taken in consideration, even if not contained in Title II of the law. The reason has been a cross-referencing from article 8.3, in relation to the role plaid by the [Garante] actor, a public body in charge of controlling the law application. The Garante has monitoring responsibilities, and may request from the data processors to "provide information and produce documents". Such information

Table 7.3: An excerpt of the Auditability Requirements document.

Auditability document	Reponsible	When used
Authorisations record	Local Authority	Request of user's authorisation Insertion of dirty data into the local DB ...
Database log	Local Authority	Insertion of dirty data ...
Broadcast log	S3	Broadcast of dirty data entries
Requests log	Local Authority	Requests of data modifications are received from the patient Changes are made in the local database
Peers list	Local Authority	Addition of a new peer to the list of known peers

and documents are those derived with the use of the *Nòmos* framework, and are in the following enumerated in Table 7.3.

Table 7.3 shows the overall results of the analysis process. It reports the auditability documents for the identified compliance goals. The table has been created by collecting from the model all the data logs used as auditing sources. The first column contains the name of the audit document. The second column contains the name of the actor, who is in charge of maintaining the document. The third column contains the cases, in which the document is modified. Basically, the content of the table has to be attached to the SRS, and specifies what documents does the system need to produce once developed, to provide compliance evidence at run-time.

7.3 Gathering implicit legal knowledge

TRACEBACK TRACEBACK was a EU-funded Integrated Project seeking to introduce new technologies to improve traceability in European food chains. Assuring the total traceability of food and feed along the whole chain from production to consumption is a cornerstone of EU policy on the quality and safety of food. This is a complex procedure involving identification, detection and processing of a vast amount of information. Profit margins of food producers and processors are already very tight, so they require a tracking mechanism that is not only reliable and easy to use, but does not entail a major cost burden. With a concerted effort and input from expert institutions, modern technology could provide such a system. TRACEBACK is developing innovative solutions based on micro-devices and innovative service-based architectures to provide innovative new information services to actors from primary food producers to consumers and health authorities. Solutions, which will include new micro-devices and a service-oriented reference architecture for traceability information systems (RATIS), are to be trialled on two major product chains — feed/dairy and tomatoes.

During the initial project activity, a team of 3 analysts, produced *i** models describing actors in the dairy food chain. The models were developed using information from descriptions of current processes and workflows in the dairy food chains in Europe, one-on-one interviews with stakeholders who fulfil modelled actor roles in these food chains, *i** modelling workshops at project partner sites, and electronic distribution of SD and SR models to stakeholders for comment and feedback. Overall the process lasted 6 months. Key results are reported in 4 basic *i** models — 1 SD and 1 SR model each for the 2 food chains: dairy and tomato food chain.

The basic *i** SD model of actors in the dairy food chain is depicted in Figure 7.8, and the inset shows part of the model in a readable form. The model

expresses 79 strategic dependencies between 13 actors from feed suppliers to transportation and even the media in a dairy food chain. The inset shows dependencies between the Feed supplier and Farm actors. For example, the Farms depend on the Feed supplier to achieve the softgoal feed contamination detected early.

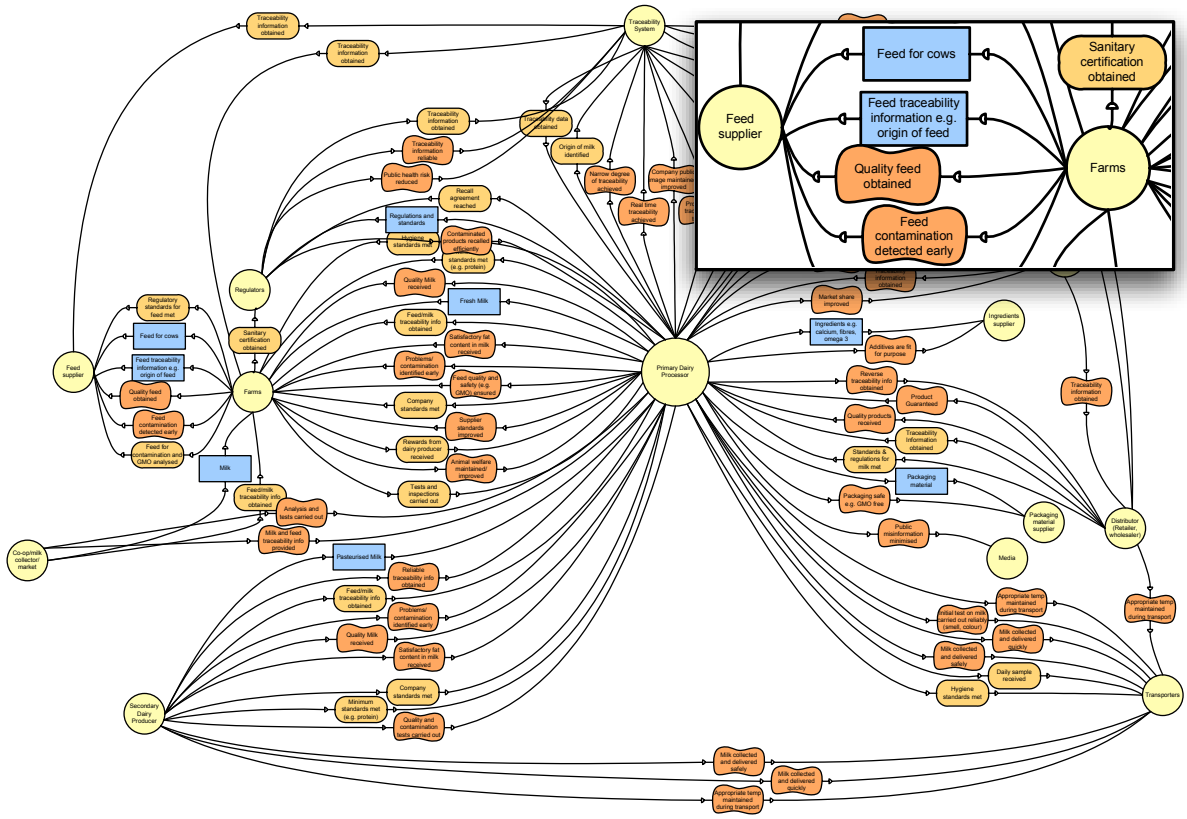


Figure 7.8: The basic *i** SD model of actors in the dairy food chain, with an inset showing dependencies between the feed supplier and farm actors.

The basic *i** SR Model for the same dairy food chain actors is depicted in Figure 7.9. The model specifies 251 different process elements and 257 different associations between these elements. The inset demonstrates part of the SR model, the feed supplier actor, in a readable form. The feed supplier undertakes the task supply feed to farms. To do this the feed supplier provides feed traceability data and uses the resource feed for cows, and seeks to achieve the

softgoal quality product stocked.

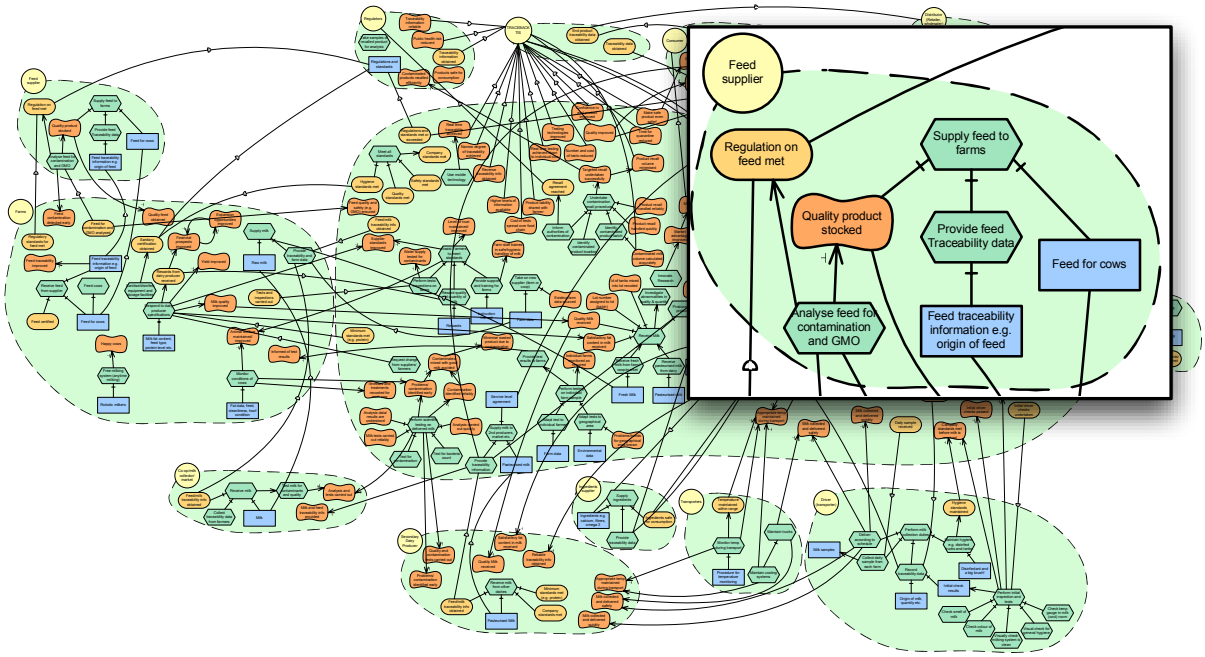


Figure 7.9: The basic *i** SR model of actors in the dairy food chain, with an inset showing the expanded food supplier actor.

Moving from the *i** models, the legal environment has been analysed with the purpose of both discovering the applicable norms and finding related stakeholders. Using documentation and information gathered from a one-on-one stakeholder interview, 7 models were developed.

An excerpt of the *Nòmos* models is depicted in Figure 7.10. The actor Food Safety Authority has been instituted by the EC178/2002 for monitoring the entire food market, whilst the Rapid Alert System, which is comprised by the national governments and the EU bodies, is in charge of receiving and dispatching alerts on food-related events. In Figure 7.10 are also depicted the results of the norms schema analysis, based on the same EU178/2002. In the following we discuss the four elements that are pointed out by the dashed arrows labelled 1, 2, 3 and 4:

1. The Rapid Alert System is devoted to the collection and forwarding of

recalls across Europe, so the Food industry operators depend on it for dispatching alerts. At the same time, the Rapid Alert System depends on the food operators for having detailed traceability information to dispatch.

2. Some goals that had emerged as Food industry operator goals did actually come from EU laws. Recalling unsafe products or warning customer is not a free choice of producers, but are needed to comply with the law.

3. To minimise the impact that the recalling policy has on the budget, food industry operators try to discover potential unsafeties as early as possible (softgoal unsafe products early detected), so they monitor the quality of the raw materials and, when possible, the production processes of their suppliers. For example, in the picture we show how operators that work in the dairy production prescribe to the farmers a sort of non-legislative regulation, the Good Manufacturing Practices (GMP), to ensure the achievement of internal goals.

4. The farmers, in turn, put into action tasks and generate goals to be able to comply with the GMP.

Whilst undertaking our RESCUE goal modelling process using basic *i**, we were aware of the existence and importance of laws and standards of behaviour but did not model these explicitly. Instead, we modelled these implicitly through the goals of the actors — for example, Farms seek to attain the goal sanitary certification obtained. However, a comparative analysis of the basic *i** and *Nòmos* models for the feed/dairy chain (Figure 7.8 and Figure 7.9) revealed that the identification and subsequent modelling of norms added some useful detail that was either overlooked or not clearly expressed in the basic *i** model. This analysis is described further below and summarised in Table 7.4. Taking the GMP norm for Farms as an example, we can see that the norms approach leads to 2 new goals being introduced — GMP minimum standards met and other GMP

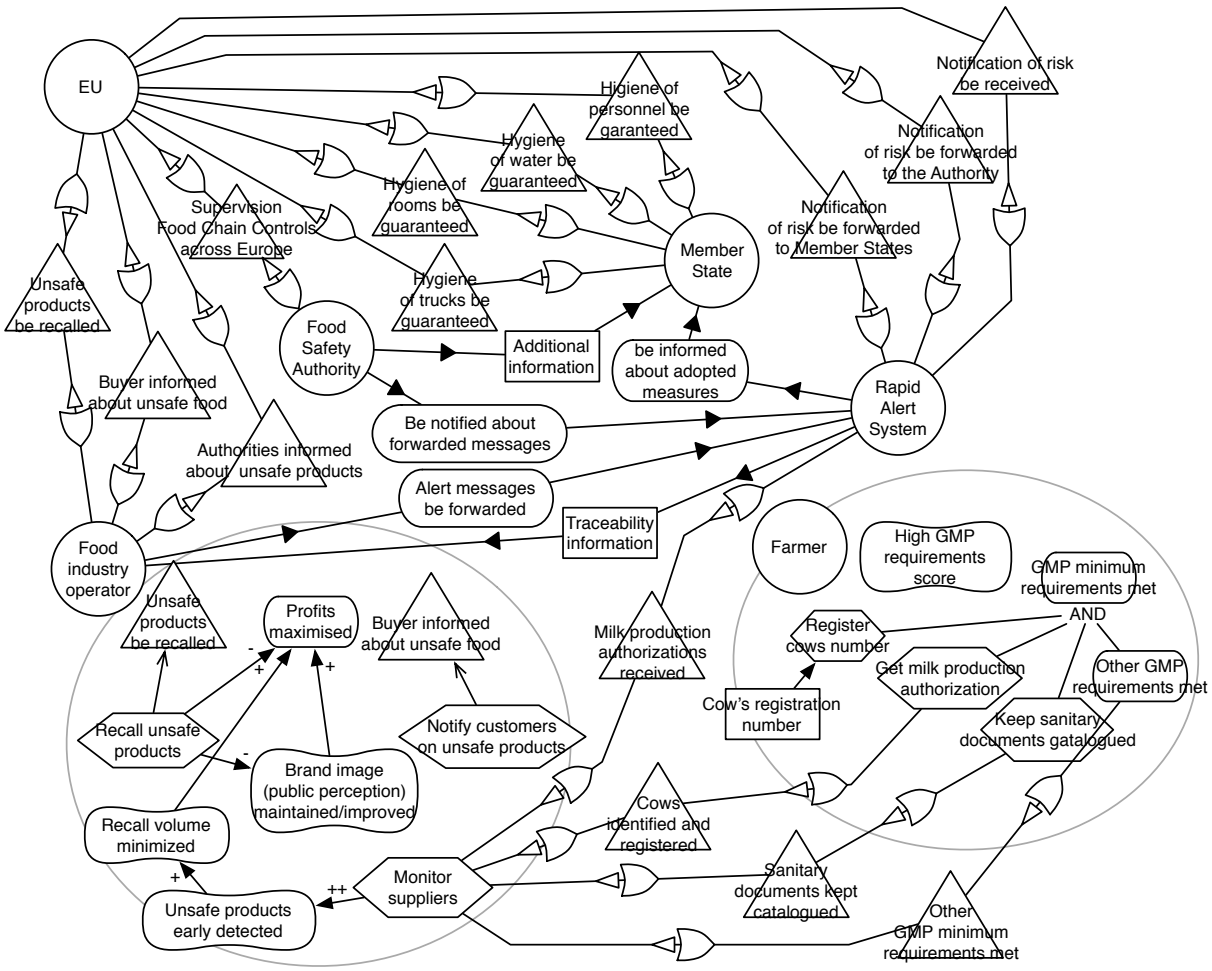


Figure 7.10: A snapshot on the domain, with regard to the regulation EC178/2002.

requirements met. As mentioned before, we touched upon some of the GMP areas using the basic *i** approach such as sanitary certification obtained (goal), whereas undertaking the task keep sanitary documents catalogued is introduced in order to meet the high level requirement (goal) of the GMP. From this simple example we can see that the normative approach can add more precision.

Whilst we believed we had already modelled the strategic elements of the Farms actor, the introduction of the GMP norm resulted in: 1 new softgoal (10 already in basic), 2 new goals (6 already in basic), 3 tasks (8 already in basic), and 1 resource (6 already in basic). The resource, softgoal and 3 tasks constitute important additions to the model, whereas the 2 new goals encompass 5 of

***Nòmos* Actor**	**Matches to basic *i** model**	**Additions to basic *i** model**	**Amendments to basic *i** model**
Food industry operator / Primary dairy producer	3 softgoals, 2 tasks, 1 goal	1 task	
Farmer/Farm		3 tasks, 1 softgoal, 1 resource	5 existing goals to be reconciled with 2 new goals
EC178/2002 / Primary dairy producer	1 goal	3 sub-goals	Must review the goal boundary between the primary dairy producer and the regulator

Table 7.4: Summary of the comparative analysis undertaken between the feed/dairy basic *i** SR model and its equivalent with addition of laws.

the original 6 goals related to standards, certification, analysis and inspections. Through further analysis of the GMP these goals could be aligned to the norm or modified accordingly.

Looking at the Primary Dairy Processor/Food Industry Operator actor boundary, we can see that the GMP norm contributed one additional task — monitor suppliers — whilst it is apparent that the other elements featured in the norms model were derived from the original *i** model. Under the basic approach, the goals contained within the GMP boundary should, in theory, feature within the actor boundary of the norm creator — in this instance the Primary Dairy Processor. A review of this actor boundary reveals 7 goals related to standards, regulations and requirements. However, none of these goals explicitly refers to the GMP goals of milk production authorisations received, cows identified and registered or sanitary documents kept catalogued, therefore we could argue for their inclusion within the basic *i** model.

Returning to the 7 goals relating to standards, regulations and requirements mentioned above, it is interesting to note that the high level goals hygiene standards met and safety standards met are elaborated upon in the norms model. The

norm 43/93/CEE provides us with the additional detail of 4 hygiene-related sub-goals, whilst the EC178/2002 norm details 3 additional safety-related sub-goals. EC178/2002 also provides us with the softgoal human health be protected that is touched upon within the Regulator actor boundary in the basic *i** model by the softgoal public health risk reduced. This brings us back a limitation of basic *i** mentioned earlier, the issue of whose actor domain the goal belongs to — is it the goal of the regulator, the diary processor or both? *Nòmos* provides us with the opportunity to treat the normative layer of the domain as a separate concern in domain modelling, hence removing this issue and supporting more effective analysis.

Another area completely overlooked by the basic *i** model was that of collecting and dispatching risk alerts, as addressed by the norm EC178/2002 (food supervision and controls). The normative model draws our attention to 3 new actors — Food Safety Authority, Rapid Alert System and Member State — which provide us with 8 additional goals. It is possible that overlooking these actors in the basic *i** approach may have had consequences further down the line for the analysis and design of the TRACEBACK socio-technical systems.

As mentioned earlier, we originally applied our standard RESCUE goal modelling process to TRACEBACK and did not explicitly model laws and regulations using basic *i**. Therefore, there is clearly an overhead associated with using the *Nòmos* approach that needs to be analysed with respect to the additional benefits it provides. We can divide our analysis into four main activities: interaction with stakeholders, inspection of documents, analysis of norm scope, and building the models. Such activities were mostly interleaved, but approximately we can estimate a 1-day interview with stakeholders; 3 days for deepening the knowledge on the norms; 7 days for exploring the norms scope and to identify the relevant ones; and finally 5 days to synthesize them and build the actual models. So, in total we can estimate that 16 person-days were spent applying the *Nòmos* approach to TRACEBACK.

evaluation Evaluating this particular use of the *Nòmos* framework we got some interesting conclusions. In purely quantitative terms, 3 new actors and 24 new process elements, including 18 new goals and 1 new softgoal, were expressed and analysed in models developed for 3 separate pieces of legislature that impacted on two existing actors — the primary dairy producer and farm actors.

The comparative analysis we undertook showed that applying the normative approach generally added more detail to the standards-related goals already present in the basic *i** model — such added detail included cow registration and sanitary documentation cataloguing. In essence, we were able to disambiguate a number of high-level goals and derive more precise properties of the system being modelled. Furthermore, explicitly modelling the laws and standards adds richness to the models that can provide benefits later on in the software development process. As TRACEBACK is developing a service reference architecture that will provide multiple instantiations of traceability information systems, knowledge of each individual domain including GMP and EU laws is important. The *Nòmos* can be used as a reference model from which analysts explore the finer details to discover important system properties and final specifications.

Another point to note in support of the *Nòmos* approach is its usefulness and effectiveness where stakeholder access is limited. For example, we did not have the means to access the farms directly, so we obtained documentation from the dairy producers about the GMP and used *Nòmos* models to infer, from scratch, the missing knowledge. In this case the norms approach was a useful and effective way to better understand the domain and capture more detailed requirements. Results from applying *Nòmos* to TRACEBACK also provided qualitative evidence to support our initial assertions. The basic *i** goal/actor metaphor cannot support a sufficiently complete representation and exploration of normative contexts in complex domains such as food traceability. Several problems identified and addressed subsequently in the project were a further

exploration of important goal boundaries between the primary dairy producer and regulator actors or between the primary dairy producer and the farmers. Evidence from TRACEBACK indicated that stakeholders often did not venture knowledge and model feedback beyond the boundaries of the actors representing them on the *i** models, and the modelling of norms helped us to overcome this limitation of the goal/actor metaphor.

Worth saying that draft basic *i** models were already available when the normative modelling began. Clearly the basic *i** models did not explicitly model the norms. Instead, with hindsight, stakeholders perceptions of norms can be inferred from the basic models. So for instance, the goal Feed regulation met in the SR model of the actor Feed Supplier depicted in Figure 7.8 represents the actor perception of the law EU178/2002.

Considering the efficiency of the approach, we estimated the time in TRACEBACK to produce and analyse the normative models against the advantages reported previously. A crude quantitative analysis of the number of modelled elements per day revealed a productivity measure of 1.7 elements/day (27 new model elements divided by 16 person-days). Although this modelling rate is low we also need to take into account the qualitative benefits of the *Nòmos* approach. Also, further analysis of the data in Table 7.4 suggests little overlap between the modelled elements in the two models, with 9 matches to the basic *i** version compared with 27 additions. This result implies that *Nòmos* complements its basic equivalent giving us benefits that appear cost-effective.

Overall, our subjective opinion is that our application of *Nòmos* to TRACEBACK was cost-effective, but further research and a detailed cost benefit analysis would need to be undertaken to provide a more objective and definitive answer to this question.

Interestingly, the laws we considered were generally quite clear and readable. It was apparent that the well-organised structure and unambiguous nature of the legislature supported the cost-effectiveness of the *Nòmos* approach. In contrast,

scope analysis resulted in being the most time-expensive activity, due to the large number of laws, several of them cross-referring each other and mostly out of scope. Building models of the legal documents is also quite time-consuming, but less than scope analysis, since norms are expressed in natural language, and to reduce ambiguity they tend to be extremely analytic. In order to get useful information from them to represent their intentional characteristics, we need to synthesize them.

Chapter 8

Conclusion

8.1 Conclusion

The interconnection of computer into a single, global, easy to access and pervasive network - the Internet - has transformed them into parts of complex entities such as software systems, software-intensive systems, or socio-technical systems. Such systems support processes and business, governments and companies, as well as professional, social and private life. Living in the Internet era has become dependent on the communication capabilities of computers, and it is not surprising that a failure in the Internet backbone may be the most disruptive event for global business. This fact raised the attention of governments on the need to regulate by law citizens' electronic interaction. In recent years, new laws have increasingly been enacted to explicitly regulate sensitive matters, such as business and health assistance, when performed on networked systems. Also, previous laws have gained new meaning when referred to an Internet-based activity.

The social relevance of information systems impacts on the way the information system has to be conceived. If misaligned with legal prescriptions, a functionality of the system can violate the rights of users, and ultimately breaches the law. Nevertheless, it's trivial mentioning that the system itself is not responsible for the breach, and someone else - the committer? the administrator? the

analyst? - will pay for it. Preventing this situation to happen is in the hand of those who are alled to define the system's functionalities: the requirements analysts.

Goal-oriented requirements engineering techniques rest on the idea of deriving the requirements for a software system from the analysis of the goals that the system-to-be will support once developed and deployed. However, when the stakeholders are addressed be laws, the system-to-be has to be aligned with the legal prescriptions too, and goals, *per se*, don't provide information about such alignment. This is the problem of law compliance of goals models.

In Chapter 3 we have envisioned the problem of law compliance of requirements as $R, K \models L$ which means that, given a set of requirements represented as actors goals, R, and a set of domain assumptions K, the requirements are compliant with a (set of) law L if, for every possible state of the world, if R holds, then L holds. Finding a solution to this problem means finding the assignment of actors responsibilities (goals) such that if every actor fulfils its goals, then law is respected — we called this condition ***Intentional Compliance***.

To produce a usable solution to the problem of law compliance of requirements, we adopt a modelling approach, which consists in starting from a model for legal prescriptions, and then building the model of goals in an incremental way that maintain the alignment with the prescriptions. The specific problem that we are trying to address here, is to **provide a definition for the notion of "alignment"** in terms of the necessity or possibility for certain strategic element to exist or not.

The contribution of this thesis is summarised as a set of conceptual tools for modelling legal prescriptions, a method to analyse the models for compliance, and a methodology to guide analysts through the modelling and analysis phases.

In Chapter 4 we have proposed a language for modelling legal prescriptions alongside stakeholders' requirements. The language is rooted in a taxonomy of legal concepts taken from the juridical literature. The modelling language rests

on a meta-model, and it is integrated into an existing goal-oriented modelling language. It also supports visual modelling, adapted from the requirements modelling notation.

In Chapter 5 we have illustrated the theoretical framework that allows to achieve compliance in a model build using the *Nòmos* framework. The conceptual tools represent a means to argue about compliance or not compliance of a requirements model. Additionally, they are a reference point for building requirements model that are compliant by construction. Basically, having represented requirements as graphs, we identify the properties of the graph to be mapped onto the definition of compliance.

In Chapter 6 we have presented a set of guidelines and a methodology for iteratively build models of compliant requirements. Overall, the methodology is comprised by several methods, which support the analyst in respectively discover the laws applicable to a given domain, check the compliance of an existing requirements set against a law, and incrementally build requirements models consistently with the theory previously exposed.

With these three components, it is possible to face those cases in which laws play a relevant role on the acquisition and engineering of requirements. We have given the flavour of this in the last chapter, but ultimately practice may differ from case to case.

8.2 Future Directions

We strongly believe that research results have not to remain isolated. We have briefly sketched up in the present work how our framework could integrate other requirements engineering frameworks for security and risk analysis, and be completed by their functionalities. To have practical relevance, it is important that the framework is linked with other techniques, and specifically with other phases of the software development process. Specifically, we expect that natu-

ral language processing techniques could improve the efficiency of the *Nòmos* process, cutting the time needed to build law models. We have reported the state of the art in this area, and will evaluate how to use such results.

More concretely, we have to acknowledge that in our framework there is a wide margin for human intervention and judgement. Entailment relations, realisations, and-relations and so on are ultimately the result of human brain, because no automatic approach has ever demonstrated to be able to replace humans. It turns out that a strongly complementary research objective has to be the support of analysts and stakeholders in managing this un-formalisable aspect. Specifically, we believe that argumentation-based approaches could help in filling those still open gaps that in the proposed framework are left to human will. Questions such as if a certain goal actually entails a normative proposition can't be objectively demonstrated, however it is possible to keep track of the decisional process and its pro/con arguments, so that the motivation of every modelling choice is at least documentable. We are currently evaluating the feasibility of this direction.

Envisioning a broader scope for the proposed framework, but moreover for the motivating issues that led to its development, we want to point out from a more critical standpoint the role played by laws in very recent years, which results even more important than how we described it. We made in the introduction a reference to the impact of the Internet on our society. We believe that this impact is still underestimated. Information technologies are growing at an increasing speed. Also, other technologies - bio-, nano-, quantum-, etc. - are growing quickly, powered by software and interconnected with each other. Technology is making possible the impossible, which human societies are not experienced to cope with. Technology and law will be most probably more and more interleaved in the future, with the first creating new scenarios, and the latter trying to regulate such scenarios. It turns out that compliance will no longer be just matter of stakeholders choices, but it will get a fallback from the nature

itself of the new technologies. This relation, between the properties of a technology and how they will have to be regulated, will become the real challenge for future scientists.

Bibliography

Basel committee on banking supervision, footnote 97, 2006.

Carlos Alchourron and Eugenio Bulygin. *Normative Systems*. Springer-Verlag, Wien/New York, 1971.

Layman Allen. Symbolic logic: A razor-edged tool for drafting and interpreting legal documents. *The Yale Law Journal*, 66(6):833–879, 1957. ISSN 00440094. URL `http://www.jstor.org/stable/794073`.

Yudistira Asnar and Paolo Giorgini. Modelling risk and identifying countermeasure in organizations. pages 55–66, Samos, Greece, August 2006. URL `http://dx.doi.org/10.1007/11962977_5`. LNCS 4347.

Carlo Biagioli and Davide Grossi. Formal aspects of legislative meta-drafting. In *Proceeding of the 2008 conference on Legal Knowledge and Information Systems*, pages 192–201, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press. ISBN 978-1-58603-952-3.

Guido Boella, Leendert van der Torre, and Harko Verhagen. Introduction to normative multi-agent systems. In *Normative Multiagent Systems 2007*, 2007.

Travis D. Breaux, Matthew W. Vail, and Annie I. Anton. Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In *RE'06: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, pages 49–58, Washington, DC, USA, September 2006. IEEE Society Press.

Travis D. Breaux, Annie I. Antón, and Jon Doyle. Semantic parameterization: A process for modeling domain descriptions. *ACM Trans. Softw. Eng. Methodol.*, 18(2):1–27, 2008. ISSN 1049-331X. doi: http://doi.acm.org/10.1145/1416563.1416565.

J. G. Cederquist, R. Corin, M. A. C. Dekker, S. Etalle, J. I. den Hartog, and G. Lenzini. Audit-based compliance control. *International Journal on Information Security*, 6(2):133–151, 2007. ISSN 1615-5262. doi: http://dx.doi.org/10.1007/s10207-007-0017-y.

Robert Darimont and Michel Lemoine. Goal-oriented analysis of regulations. In Régine Laleau and Michel Lemoine, editors, *ReMo2V, held at CAiSE'06*, volume 241 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

Mehdi Dastani, Virginia Dignum, and Frank Dignum. Organizations and normative agents. In *EurAsia-ICT*, pages 982–989, 2002.

Marc Esteva. *Electronic institutions. from specification to development*. PhD thesis, Universitat Politecnica de Catalunya, 2003.

Ariel Fuxman, Lin Liu, John Mylopoulos, Marco Roveri, and Paolo Traverso. Specifying and analyzing early requirements in tropos. *Requiruirements Engineering*, 9(2):132–150, 2004.

Sepideh Ghanavati, Daniel Amyot, and Liam Peyton. Towards a framework for tracking legal compliance in healthcare. In John Krogstie, Andreas L. Opdahl, and Guttorm Sindre, editors, *CAiSE*, volume 4495 of *Lecture Notes in Computer Science*, pages 218–232. Springer, 2007. ISBN 978-3-540-72987-7.

Sepideh Ghanavati, Alberto Siena, Liam Peyton, Daniel Amyot, Anna Perini, and Angelo Susi. Integrating business strategies with requirement models of legal compliance. *International Journal of Electronic Business (IJEB)*, 2010.

Paolo Giorgini, John Mylopoulos, Eleonora Nicchiarelli, and Roberto Sebastiani. Reasoning with goal models. In *21st International Conference on conceptual Modeling (ER2002)*, Tampere, Finland, 2002.

Paolo Giorgini, Fabio Massacci, John Mylopoulos, and Nicola Zannone. Requirements engineering meets trust management: Model, methodology, and reasoning. In *ITRUST-04*, volume 2995 of *LNCS*, pages 176–190. SVG, 2004.

Paolo Giorgini, Fabio Massacci, John Mylopoulos, and Nicola Zannone. Modeling security requirements through ownership, permission and delegation. In *RE*, pages 167–176, 2005.

Giancarlo Guizzardi, Gerd Wagner, Nicola Guarino, and Marten van Sinderen. An ontologically well-founded profile for uml conceptual models. In *CAiSE*, pages 112–126, 2004.

Herbert L. A. Hart. *The Concept of Law*. Clarendon Press, Oxford, 1961.

Wesley Newcomb Hohfeld. *Fundamental Legal Conceptions as Applied in Judicial Reasoning*. Yale Law Journal 23(1), 1913.

Ivan Jureta, John Mylopoulos, and Stéphane Faulkner. Revisiting the core ontology and problem in requirements engineering. *CoRR*, abs/0811.4364, 2008.

Hans Kelsen. *General Theory of Norms*. Clarendon Press, Oxford, 1991.

Nadzeya Kiyavitskaya, Nicola Zeni, Travis D. Breaux, Annie I. Antón, James R. Cordy, Luisa Mich, and John Mylopoulos. Automating the extraction of rights and obligations for regulatory compliance. In Qing Li, Stefano Spaccapietra, Eric Yu, and Antoni Olivé, editors, *ER*, volume 5231 of *Lecture Notes in Computer Science*, pages 154–168. Springer, 2008. ISBN 978-3-540-87876-6. URL `http://dblp.uni-trier.de/db/conf/er/er2008.html#KiyavitskayaZBACMM08`.

Luigi Logrippo. Normative systems: the meeting point between jurisprudence and information technology?a position paper. In *Proceeding of the 2007 conference on New Trends in Software Methodologies, Tools and Techniques*, pages 343–354, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press. ISBN 978-1-58603-794-9.

L. Thorne McCarty. The TAXMAN project: Towards a cognitive theory of legal argument. In *Computer Science and Law*. Cambridge University Press, 1980.

L. Thorne McCarty. A language for legal discourse i. basic features. In *ICAIL '89: Proceedings of the 2nd international conference on Artificial intelligence and law*, pages 180–189, New York, NY, USA, 1989. ACM. ISBN 0-89791-322-1. doi: http://doi.acm.org/10.1145/74014.74037.

Yoram Moses and Moshe Tennenholtz. Artificial social systems. *Computers and AI*, 14:533–562, 1995.

John Mylopoulos, Lawrence Chung, and Eric Yu. From object-oriented to goal-oriented requirements analysis. *Communications of the ACM*, 42(1):31–37, 1999. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/291469.293165.

Bashar Nuseibeh and Steve M. Easterbrook. Requirements engineering: a roadmap. In *International Conference on Software Engineering - Future of SE Track*, pages 35–46, Limerick, Ireland, 2000.

Andre Rifaut and Eric Dubois. Using goal-oriented requirements engineering for improving the quality of iso/iec 15504 based compliance assessment frameworks. In *RE '08: Proceedings of the 2008 16th IEEE International Requirements Engineering Conference*, pages 33–42, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3309-4. doi: http://dx.doi.org/10.1109/RE.2008.44.

Corrado Roversi. *A Treatise of Legal Philosophy and General Jurisprudence*, volume 1, part II, chapter 16. Springer Netherlands, 2005.

Giovanni Sartor. Fundamental legal concepts: A formal and teleological characterisation. *Artificial Intelligence and Law*, 14(1-2):101–142, April 2006. ISSN 0924-8463. doi: http://dx.doi.org/10.1007/s10506-006-9009-x. URL `http://dx.doi.org/10.1007/s10506-006-9009-x`.

Giovanni Sartor. *Concepts in Law*, volume 88, chapter Understanding and Applying Legal Concepts: An Inquiry on Inferential Meaning, pages 35–54. Springer Netherlands, 2009.

M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. The british nationality act as a logic program. *Communications of the ACM*, 29(5):370–386, 1986. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/5689.5920.

Alberto Siena. Engineering normative requirements. In Colette Rolland, Oscar Pastor, and Jean-Louis Cavarero, editors, *1st IEEE International Conference on Research Challenges in Information Science (RCIS'07)*, pages 439–444, 2007. URL `http://dblp.uni-trier.de/db/conf/rcis/rcis2007.html#Siena07`.

Alberto Siena, Neil A. M. Maiden, James Lockerbie, Kristine Karlsen, Anna Perini, and Angelo Susi. Exploring the effectiveness of normative i* modelling: Results from a case study on food chain traceability. In *20th International Conference on Advanced Information Systems Engineering (CAiSE'08)*, pages 182–196, 2008a.

Alberto Siena, John Mylopoulos, Anna Perini, and Angelo Susi. From laws to requirements. In *1st International Workshop on Requirements Engineering and Law (Relaw'08)*, 2008b.

Alberto Siena, John Mylopoulos, Anna Perini, and Angelo Susi. Designing law-compliant software requirements. In *31st International Conference on Conceptual Modeling (ER'09)*, pages 472–486, Gramado, Brasil, November 09 2009a.

Alberto Siena, John Mylopoulos, Anna Perini, and Angelo Susi. Towards a framework for law-compliant software requirements. In *ICSE Companion*, pages 251–254, 2009b.

Alberto Siena, John Mylopoulos, Anna Perini, and Angelo Susi. A meta-model for modeling law-compliant requirements. In *2nd International Workshop on Requirements Engineering and Law (Relaw'09)*, 2009c.

Ronald Stamper. Legol: Modelling legal rules by computer. *Computer Science and Law*, 1980.

Angelo Susi, Anna Perini, John Mylopoulos, and Paolo Giorgini. The Tropos metamodel and its use. *Informatica (Slovenia)*, 29(4):401–408, 2005.

André Valente. *Legal Knowledge Engineering: A Modelling Approach*. IOS Press, Amsterdam, The Netherlands, 1995.

Axel van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Fifth IEEE International Symposium on Requirements Engineering*, pages 249–262, Toronto, Ontario, Canada, 2001. IEEE Computer Society.

Axel van Lamsweerde and Emmanuel Letier. Handling obstacles in goal-oriented requirements engineering. *IEEE Transactions on Software Engineering*, 26(10):978–1005, 2000.

Georg Henrik von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

Eric Siu-Kwong Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, 1996.

Pamela Zave and Michael Jackson. Four dark corners of requirements engineering. *ACM Transactions on Software Engineering Methodology (TOSEM)*, 6 (1):1–30, 1997.

Glossary

Applicability
: Relation between a goal and a normative proposition, which says that the goal, according to how it is achieved, may bring to compliance or uncompliance with the normative proposition. 63

Claim
: Legal right to have something done by somebody else. 29

Disability
: Legal lack of ability for accomplishing a certain action. 30

Duty
: Legal imposition to somebody to perform a certain action towards somebody else. 29

Embodiment relation
: Relation between an actor of the domain and a legal subject. 63

Hohfeldian taxonomy
: Taxonomy of legal relations. Created by W. N. Hohfed. 28

Immunity
: Legal ability of a subject to be kept untouched from others' power. 30

Legal subject

Abstract characterisation of an actor. It's the way laws identify actors in the domain without explicitly addressing them. 31

Liability

Legal subjection of a subject to somebody else's power. 30

No-Claim

Lack of legal ability to pretend something from somebody else. 29

Power

Legal ability to produce changes in somebody else's legal position (set or rights). 30

Privilege

Legal liberty whether to accomplish an action or not. 29

Realization relation

Relation between a goal and a normative proposition, which says that, for the owner, achieving the goal allows to be compliant with the normative proposition. 63