Nòmos 3: Legal Compliance of Roles and Requirements

Silvia Ingolfo[1], Ivan Jureta[2], Alberto Siena[3], Anna Perini[3], and Angelo Susi[3]

[1] University of Trento, via Sommarive 7, Trento, Italy
[2] University of Namur, 8, rempart de la vierge, 5000 Namur, Belgium
[3] FBK-Irst, via Sommarive 18, Trento, Italy
ingolfo@unitn.it, ivan.jureta@unamur.be, {siena,susi,perini}@fbk.eu

Abstract. The problem of regulatory compliance for a software system consists of ensuring through a systematic, tool-supported process that the system complies with all elements of a relevant law. To deal with the problem, we build a model of the law and contrast it with a model of the requirements of the system. In earlier work, we proposed a modelling language for law (Nòmos 2) along with a reasoning mechanism that answers questions about compliance. In this paper we extend Nòmos 2 to include the concepts of role and requirement so that we can reason about compliance in specific domains. Also, Nòmos 3 represents the distribution of responsibilities to roles, distinguishing social from legal roles. Nòmos 3 models allow us to reason about compliance of requirements and roles with the norms that constitute a law. A small case study is used to illustrate the elements of Nòmos 3 and the kinds of reasoning it supports.

Keywords: requirement engineering, regulatory compliance, roles.

1 Introduction

Government has become increasingly interested in how to regulate software systems, given numerous mishaps with significant and expensive consequences to the public. When designing a new system, it is becoming necessary to demonstrate that the system complies with applicable legislation. Similarly, given an existing running system and a new law coming into force, it is important to evaluate its compliance and adapt the system design accordingly. The new challenge for software engineers is to understand the various ways a system can achieve its purpose, while complying with applicable laws.

In previous work we have introduced Nòmos 2 [13], a modeling language, tailored to represent norms and support formal reasoning about alternative ways to comply with them. It relies on the intuition that laws generally establish *norms* (i.e., duties and rights), but also the *context* in which these norms apply, their pre-conditions, exceptions and relationships. Nòmos 2 takes into account this complex structure of a law and allows the analyst to answer questions regarding the *applicability* and *satisfaction* of norms in different situations. In Nòmos 2 *compliance* to a norm is analyzed with respect to those two factors.

E. Yu et al. (Eds.): ER 2014, LNCS 8824, pp. 275–288, 2014.

However, an important factor in evaluating the compliance of requirements to a norm comes from the analysis of *who* must comply with it, which is just as important as *what* compliance entail. One of the contributions of this paper rests in the intuition that if the right requirement is assigned to the wrong actor in the domain, we have non-compliance. This is particularly critical when designing the system-to-be. Software systems are typically complex systems comprised by many software and hardware components and integrated into social systems, such as business processes and other organizational structures. Requirements analysts assign responsibilities to these technical and social parts of the systems, and the responsibilities must meet those that are given by law. For example, The Italian Privacy Law addresses *Data Subject* and *Data Processor* in its norms, but in a set of requirements describing a software system for phone companies, relevant roles might include *SIM-user* or *Phone Operator*. *When* the analyst specifies that the *SIM-user* provides its data, it is binding this role of the domain to that of the *Data Subject*, making the *SIM-user* responsible for the norms addressing this legal role. In other words, by designing system roles and deciding which requirements they are responsible for, the analyst implicitly defines to which legal roles are involved. These two aspects — identifying which norms a role is responsible for, and *who* is the responsible for the norms in the domain — are crucial aspects for the evaluation of compliance of a set of requirements, and we need to be able to represent these aspects in our modeling language.

In this paper we extend our previous work in order to represent roles and requirements, and reason about roles and their responsibilities in the evaluation of compliance for a set of requirements. The main contribution of this paper is to introduce Nòmos 3, a modelling language for evaluating compliance of roles and requirements in the domain of Requirement Engineering (RE). This modeling language is supposed to be used by requirement analysts who use requirements models and existing models of the law to perform the analysis. One important feature of Nòmos 3 is the ability to represent and clarify: when a role is responsible for bringing about a situation, and what it means for a role to have 'fulfilled' its responsibilities. Nòmos 3 extends the reasoning capabilities of Nòmos 2 by offering answers to the following important questions: **Q1**: Which roles in the requirements are subject to norms?; **Q2**: To which norms these roles must comply?; **Q3**: Which roles in the requirements have fulfilled their responsibilities/which roles comply?; **Q4**: If a role in the domain fulfills its requirements, has it fulfilled its legal responsibilities?

The paper is organized as follows. We first introduce Nòmos 2 as our baseline (section 2). Section 3 describes the core language of Nòmos 3, some basic relationships, and section 4 how Nòmos 3 supports reasoning for compliance evaluation. We describe in section 5 the validity of our proposal through an illustrative case study. In section 6 we discuss related work, and we conclude in section 7.

2 Baseline

Regulatory documents, such as laws, regulations and policies, are complex artefacts: they contain elements such as conditions, exceptions or derogations

defining the applicability or satisfaction conditions for the norms (rights and obligations) that constitute a law. Nòmos 2 is a modelling language proposed in [13] that aims at capturing the complexity of regulatory documents by means of conceptual models. The language is founded on the premise that conceptual models may help the communication with the stakeholders, and is based on the concepts of Norm and Situation. A *Norm* is the most atomic fragment of law with a deontic status, and in combination with Situations it is used to make inferences about compliance. A Norm can be *applicable*, if its pre-conditions hold; *satisfied* if its post-conditions hold; *complied* if it is applicable and satisfied. The applicability and satisfaction of a norm depends on the situations (partial states of the world, or states-of-affairs) holding, the idea being that if some situations hold, the norm will apply/be satisfied.

Relationships in Nòmos 2 act as label-propagation channels from their source (for example, a situation or norm) to their target. Forward and backward reasoning algorithms are used to support useful analysis of norm models, such as *applicability analysis*, intended to find the (sub)set of norms applicable to a given (sub)set of situations; *compliance analysis*, which aims at providing evidence of compliance (or violation) of a norm model; *compliance search*, which aims at finding a solution described in terms of a set of situations holding that will make some given norms complied with.

Nòmos 2 effectively explores compliance alternatives, however it fails in capturing the implications that follow from the existence of different roles, which can charged with the duty of complying.

3 Concepts, Relations, and Value Types in Nòmos 3

A Nòmos 3 model consists of a set of propositions, which can be either instances of concepts or relationships between concepts. All concepts and relationships have two purposes. One is *representation*, for the sake of documentation of a domain for model users; after all, this is what "making models" or "modelling" usually refer to. The other is *reasoning*, which here means making inferences from the elements of a model. This is accomplished by allowing propositions to obtain values. Values in Nòmos 3 reflect concerns that we believe are useful for evaluating compliance, and value types constitute a generalization of the notion of truth value in logic. We introduce the following notational conventions: if the name of a concept is "norm", we write C-Norm, and Norm for its generic instance; if the name of a relation is "satisfaction", then we write R-Satisfaction; if the name of a value type is "assignment", we write V-Assignment.

3.1 Primitives

Concepts, relations, and value types presented below are primitive, that is, not defined in terms of others, or of one another. All other concepts, relations, and value types in Nòmos 3 are defined from this language core.

Situations. Instances of C-Situation are propositions used to represent states of the world. Situations are used to represent pre- and post-conditions of norms (Nòmos 2), and will be used similarly to represent requirements in Nòmos 3. It is the concept of Situation that makes Nòmos 3 versatile, in the sense that its models can be combined with requirements models made with various requirements modelling languages. For example, a requirements model may include a goal that "User should be able to ask for reimbursement, if dissatisfied with a product". The resulting system may generate two Situations "User is dissatisfied with a product" and "System allows the user to ask for reimbursement"; if we want to look at the consequences on compliance of the goal being satisfied, we can set the values of both Situations to be satisfied, and compute how this influences the compliance with Norms.

C-Situation is associated with a single value type, called V-Satisfaction. This value type allows an instance to take one of three values:

- Satisfied (abbreviated ST), if the Situation corresponds to the world,
- Failed (SF), if the Situation does not correspond to the world,
- Unknown (SU), if it is unknown if the Situation is satisfied. We use the third value (unknown), because our assignments of labels may be incomplete, yet we may want to do reasoning over models where we do not know, or do not want to assign a value before we start reasoning. With the third value, we avoid assuming that such leaf nodes without values are, by default, satisfied or failed (the use of three values is somewhat non-standard, but is not new [12]).

The satisfaction value that a Situation gets in a Nòmos 3 model depend on the values assigned to relationships that it is in. Rules for computing values are defined later.

Roles. We use Roles to represent the distribution of responsibility, and to evaluate the effects of fulfilling or failing these responsibilities. *Responsibilities are allocations of Situations to Roles*, such that if the Situations are satisfied, the responsibilities are fulfilled, and if all responsibilities are, then we can say that the Role is fulfilled.

There are two kinds of Roles. C-Role.Legal is responsible for Situations that appear in Norms, while C-Role.Social is responsible for Situations that appear in Goals. The intuition is that there are Roles which are not defined by laws and regulations, but rather in the domain where the requirements are defined: these are Social Roles. The Goals assigned to a Social Role represent requirements the role has, or goals assigned to a role for fulfillment. Legal Roles are defined by laws and regulations, and they exist only because laws and regulations dictate that some situations become true (obligations) and others can become true (rights), and in both cases somebody has the obligation/right to make this happen.

Since we allocate Situations to Roles — and call these allocations responsibilities — we will talk about the fulfilment of responsibilities, and thereby of Roles. We use the value type V-Fulfilment, with the following values:

- Fulfilled (FT), if the Role fulfilled all its responsibilities,
- Unfulfilled (FF), if the Role has not fulfilled all its responsibilities,
- Unknown (FU), if it is unknown if the Role is fulfilled.

The satisfaction values of Situations, which the Role is responsible for, determine the fulfilment value of the Role. Asserting that a Role is fulfilled, in turn, equates to asserting that all the Situations it is responsible for, are satisfied.

Primitive Relationships. In Nòmos 3 a relationship instance between two proposition p_1 and p_2 is also a proposition that has the format "if p_1 is satisfied, then p_2 is/is-not satisfied".

Since the value of p_2 depends on (a) the value of p_1 and (b) the type of relationship between the two, we consider relationship as propagating a value to their target when the relationship holds. In table 1 we summarize how V-Satisfaction and V-Fulfillment value are propagated over the three primitive relationships (see the complementary technical report [6] for more details).

Table 1. Propagation of values across the three basic relations

$p_1 \xrightarrow{\text{make}} p_2$		$p_1 \xrightarrow{\text{break}} p_2$		$p_1 \xrightarrow{\text{take}} p_2$	
p_1	p_2	p_1	p_2	p_1	p_2
ST	ST	ST	SF	ST	FT
SF	SU	SF	SU	SF	FF
SU	SU	SU	SU	SU	FU

Make. The R-Make relationship instance is a proposition that has the format "if Situation s_i is satisfied, then Situation s_j is". R-Make is a relationship between Situations s_i and s_j — R-Make(s_i, s_j) — and the V-Satisfaction value propagated depends on the V-Satisfaction value of s_i (see table 1).

Break. A R-Break relationship instance is a proposition that has the format "if Situation s_i is satisfied, then Situation s_j is not". R-Break is, roughly speaking, the opposite of R-Make. R-Break is also a relationship between Situations s_i and s_j — R-Break(s_i, s_j) — and the V-Satisfaction value propagated depends on the V-Satisfaction value of s_i (see table 1).

Take. R-Take relationship instance is a proposition with the format "Role r is responsible for Situation s", used to indicate that the Role is responsible for satisfying the Situation, and that the V-Fulfilment value of the Role depends on the V-Satisfaction value of the Situation. R-Take is a relationship between Situations and Roles s and r — R-Take(s, r) — and it is read as "if Situation s is satisfied, then Role r is fulfilled". We the say that a Role r fulfills its responsibility if Situation s has value ST, when it has value SF, r is not fulfilling its responsibility (see table 1).

AND/OR. R-AND/R-OR are meta-relationships. An instance of R-AND or R-OR is a relationship over a set of Make, Break, or Take relationships of the same type (so the set cannot include, for example, both Make and Break relationships). The value of a meta-relationship instance depends on the values of the relationships it is over. These values are assigned as in Nòmos 2: for example, R-AND will have the value ST iff all the sources of the relationships have the value ST, and when the target concept of the relationship is a Situation, it will therefore be assigned ST (FT if the target is a Role). An R-OR will be ST if at least one of the sources of the relationships in it is ST, and when the target concept of the relationship is a Role, it will be assigned FT. All rules for computing R-AND and R-OR values are defined in the technical report [6]. For example when two situations s_i, s_j in conjunction target a Role instance r, then the fulfillment value of r is FT only when the when both situations are ST.

Similarly, when two situations s_i, s_j in disjunction target another Situation instance s, then the satisfaction value of s is ST when s_i or s_j is ST. Nòmos 3 has its own rules for aggregating values that a situation or another node accumulates (see [6]). These aggregation rules apply whenever a situation has at least two or more incoming edges. For example, suppose that a Situation X has two incoming edges, one saying that X should have the value ST, the other SF. So we say that this Situation X accumulates ST and SF, and the aggregation rule should say which single value X should have, given those that it accumulates.

3.2 Language Parts for Compliance Evaluation

We evaluate *the compliance of a set of Requirements to Norms*. Requirements are represented in terms of Goals and Social Roles desiring them, and are described in a model in some modelling language (e.g. i*). We will call *domain model* this representation of the requirements. The model may describe actual elements (e.g., Goals) and conditions that will always hold in the domain (e.g. Domain Assumptions), and these can be represented by Situations. The model would also describe desirable actions and conditions whereby we still can identify Situations, but now both desirable ones, and undesirable ones, and evaluate the consequences of these alternatives on compliance. Moreover, by defining responsibilities for a Role, we are representing that some sets of Situations should be satisfied by that Role. We do not require all Situations to be responsibility of a Role: for example a Situation like "It is Christmas season" is not responsibility of any Role, and we refer to these as accidental Situations. We refer to intentional Situations those who are responsibility of a Role, like "Expressed shipping is requested".

To evaluate the compliance of the requirements to Norms, we need to represent actions and conditions imposed by law and regulations, that is define instances of the C-Norm.

A central idea in Nòmos 3 is that *to evaluate the compliance of a domain model to Norms, we need to determine (i) which Norms are applicable, and therefore need to be complied with, and (ii) which Social Roles, because of their responsibilities in the domain model, also are subject to Norms, and therefore accumulate the responsibilities of Legal Roles.* These two are closely tied: it is because of the accidental or intentional satisfaction of Situations, that some specific Norms become applicable; once they do, we have to worry about new Legal Roles and their responsibilities. The Social Roles in the domain model have to accumulate the responsibilities of these Legal Roles, so as to ensure compliance.

Norms. Instances of C-Norm are representations of conditions, which are given by laws or other regulations that the system-to-be should comply with, or are otherwise relevant in evaluating compliance. Like in Nòmos 2, a Norm is a written as the five-tuple ($n.type$, $n.holder$, $n.counterpart$, $n.antecedent$, $n.consequent$) where n is the identifier for the Norm instance; $n.type$ categorizes the Norm as either duty or right;[1] $n.holder$ is the C-Role.Legal instance which takes the

[1] Obligations and prohibition in Deontic Logic can be captured by our concept of duty, and permission by the concept of right.

responsibility for satisfying the *n.consequent*; *n.counterpart* is the optional C-Role.Legal instance which benefits when the *n.consequent* is satisfied (if specified); *n.antecedent* are the C-Situation instances, which if satisfied, make the Norm applicable, that is *n.holder* should satisfy *n.consequent*; *n.consequent* are the C-Situation instances, such that the Norm is complied with if and only if these Situations are satisfied.

A Norm is evaluated with a V-Compliance value type. Each allowed value of V-Compliance corresponds to a composite predicate on the values of a Norm's holder, counterpart, antecedent, and consequent — this is a departure from Nòmos 2, where this value only depends on the antecedent and consequent Situations. The following rules introduce the compliance values:

- Compliant (CT) if the norm applies, is satisfied, the holder is fulfilled, and the counterpart is fulfilled (*when that counterpart Role exists*). In values we write compliance iff V-Fulfilment(*n.holder*)=FT and V-Fulfilment(*n.counterpart*)=FT and V-Satisfaction(*n.antecedent*)=ST and V-Satisfaction(*n.consequent*)=ST.
- Incidental Compliant if the norm applies, is satisfied, but the holder/counterpart is either not fulfilled, or its fulfilment is unknown. The interesting idea is that it allows to identify cases in which the correct Situations are brought about, but the Legal Roles have not actually fulfilled their responsibilities.
- Tolerance (CT) if the norm *does not* apply, it is satisfied and the holder/counterpart is fulfilled. The idea is that the role is correctly complying with a norm even if it does not have to.

The other two compliance values are the same as in Nòmos 2, and do not depend on the values on the holder/counterpart:

- Non-Compliant (CF) if the norm of type duty applies but it is not satisfied.
- Inconclusive (CU) if it is unknown if the norm applies.

Goals. Goals are state of affairs desired by a Social Role,[2] and are used to represent Requirements. For example "Apply for a reimbursement" represents a Goal desired by the Social Role *SIM-user*. Nòmos 3 evaluates the compliance of a given goal by taking into account: (i) the Role and its associated Goals, and (ii) the Situations describing the Goal. A set of Situations, in conjunction or disjunction, can represent the achievement of the Goal as well as the Situations that can be satisfied to achieve the Goal. For example the situation "Reimbursement claim is filed" is the situation associated to the achievement of the Goal "Apply for a reimbursement", and it can also be read as the situation that is satisfied when the goal is achieved. This former reading allows us to read the *SIM-user* as the holder of the Goal, and therefore being assigned/allocated the situations satisfying the Goal.

Domain Assumptions. A domain assumption is a state of affairs that is known to hold in a given scenario. For example the situation "The company is non-profit organization" can be included as an assumption where in the reasoning it is always considered with satisfaction value ST.

[2] In Nòmos 3 we are not interested in specific agents, so we relate a social role to the goal. The agent who ends up occupying it is the one who is assumed to desire the goal.

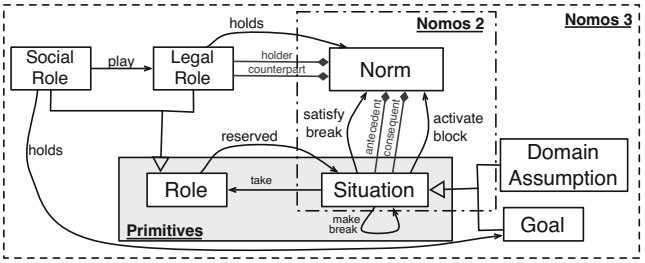

Fig. 1. Metamodel of Nòmos 3

Representing Responsibility and Compliance in Nòmos 3 Models.
In Nòmos 2 four basic relations are identified to propagate applicability and satisfiability values from Situations to Norms. The same is achieved and expanded in Nòmos 3 (figure 1), where we summarize the relations derived from the core-relations used in Nòmos 3 models:
• "activate/block" are Nòmos 2 relations which can be written as a R-Make/R-Break relations from Situations to Norms: when a Situation is satisfied, then the *antecedent* of the Norm is satisfied/not-satisfied.
• "satisfy/break" are Nòmos 2 relations which can be written as R-Make/R-Break relations from Situations to a Norm: when a Situation is satisfied, then the *consequent* of the target concept is satisfied/not-satisfied. In Nòmos 3 we use the same relation to also relate Situations with Goals. For example in figure 2, when either of the situations s_1 or s_2 is satisfied, then the goal G_1 is satisfied.
• "hold" is a relation from a Legal-Role to a Norms — and from a Social-Role to a Goal — representing the fact that the Role is responsible (R-Take) for bringing about the Situations in the consequent of that Norms/Goal. In figure 2, 'SR_3 *holds* G_3' represents that the *SIM User* is responsible for the Situations satisfying G_3 (i.e., s_{11}).
• "play" is a relation from a Social-Role to a Legal Role representing the fact that the Social Role plays the Legal Role and becomes responsible for bringing about the Situations for whom the Legal Role is responsible. In figure 2, 'SR_3 *plays* LR_3' represents that the Social Role SR_3 is responsible for the Situations consequent of the Norms that LR_3 holds.
• "reserved" is a R-Take from a Situation to a Role, identifying that the Role is *the only* responsible for that Situation. In figure 2 for example, the *Data Subject* is the only Legal Role who can give a consent ($s_{14} \xrightarrow{\text{reserved}} LR_4$); similarly, the association of a phone number to a user is only responsibility of the *Company Operator* ($s_{10} \xrightarrow{\text{reserved}} SR_2$).
• "imply/derogate/endorse" are relations between Norms introduced in Nòmos 2 to represent legal variability: an *imply* relation from N_1 to N_2 represents that when N_1 is complied, then also N_2 is; a *derogate* relation is used to represents that when N_1 is applicable, then N_2 is not; an *endorse* relation represents that when N_1 is applicable, also N_2 is. For example in figure 2, the Right R_3 for the

customer to obtain a reimbursement is complied when the Seller complies with its duty D_3 to reimburse the money ($D_3 \xrightarrow{\text{imply}} R_3$).

Responsibility for a Role is represented through two relations in Nòmos 3: the *reserved* relation for situations which can *only* be brought about by a specific Role, and through the *hold* relation for Situations that are assigned to a given Role in order to fulfill its Social/Legal responsibilities.

Visual notation. In a Nòmos 3 model (see figure 2) we represent Situations with rectangles, Norms with triangles, Goals with ovals and Roles with circles.

4 Reasoning in Nòmos 3

Adding Norms and Goals to the primitive language lets us make Nòmos 3 models that can answer interesting questions about compliance. Reasoning in Nòmos 3 amounts to different type of if-then analysis. By assigning values to a selected set of Situations, we can compute the values of other Situations and Roles, which are related via Make, Break, and Take relations. The compliance analysis provided by Nòmos 2 and still supported in Nòmos 3 was related to the following question:

SituationsToNorms: *Which Norms obtain the compliance value w, if we assign the satisfaction value v_1 to Situation s_1, v_2 to s_2, ..., v_n to s_n?*

The question summarizes a type of if-then analysis, where we are assuming some Situations are satisfied, failed, or whose satisfaction is unknown, and we want to understand the effects of this on compliance (which is now defined in terms of both Roles fulfillment and Situation satisfactions).

A new kind of analysis in Nòmos 3 involves Roles, which we use to evaluate the consequences on compliance, of fulfilling or failing responsibilities. If we assert that some specific Legal Roles are fulfilled, we can determine how that relates to compliance, which amounts to asking this question for a given Nòmos 3 model:

RolesToNorms: *Which Norms obtain the compliance value w, if we assign the fulfilment value v_1 to Legal Role r_1, v_2 to r_2, ..., v_n to r_n?*

We can also assert the fulfilment of Social Roles, to see the consequences on Legal Roles. This is summarized in the following question.

SocialRolesToLegalRoles: *Which Legal Roles obtain the fulfilment value w, if we assign the fulfilment value v_1 to Social Role r_1, v_2 to r_2, ..., v_n to r_n?*

It is an interesting question, because if the Social Role is fulfilled, it will satisfy Situations which may make Norms applicable. The Norms in turn introduce Legal Roles, and we want to know which of these Legal Roles are fulfilled as well.

Another issue is to look at the consequences on responsibility fulfilment, of having specific satisfaction values assigned to Situations:

SituationsToRoles: *Which Roles obtain the fulfilment value w, if we assign the satisfaction value v_1 to Situation s_1, v_2 to s_2, ..., v_n to s_n?*

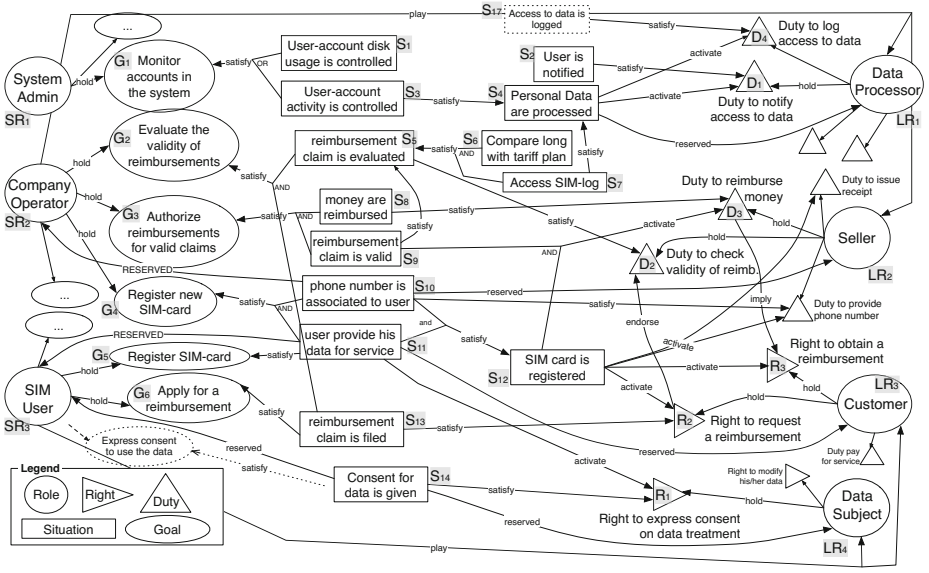

Fig. 2. Nòmos 3 model of the scenario

Adding Goals to Nòmos 3 does not change the questions we can ask, but simply how we read these questions and their answers. For example, if we want to evaluate the compliance of some Goals, then we are asking the same question as SituationsToNorms, except that now, we know that these Situations originate in instances of concepts from the goal model, and reflect the responsibilities of some Social Roles. In a goal model, then that question can read "Which Norms obtain the compliance value w, if goals $g_1, \ldots, g_n$ are achieved (that is, the situations $s_1, \ldots, s_n$ get the value ST?"

5 Nòmos 3 at Work

In this section we illustrate the Nòmos 3 language and its reasoning capabilities using as example a set of requirements for a software managing the reimbursement requests of the customers of a phone company. In figure 2 we provide the Nòmos 3 model of the example. On the left side of figure 2 we have represented some goals of the social roles involved (*System Admin*, *Company Operator*, and *SIM-user*). On the right side we have represented some norms regulating phone contracts and some norms from the Privacy Law regulating data processing.

Q1 *Which roles in the domain are subject to norms?* To answer this question, we start with a given set of goals (G_1–G_6) represented by: a set of situations holding, and social roles responsible for bringing about these situations. These situations make applicable norms in the legal model for which some legal roles are responsible. We identify and link a social role with the norms it has to comply with through the $\xrightarrow{\text{play}}$ relation. The *Company Operator* is therefore subject to the norms addressing the *Seller*. Moreover the *Company Operator* has to evaluate

the reimbursement claim (G_2) and this requirement is satisfied also by accessing the SIM-log of the user (s_7). However when the log of the phone is accessed, the law considers this operation as type of processing of personal data ($s_7 \xrightarrow{\text{satisfy}} s_4$). Since s_7 is responsibility of the legal role *Data Processor*, the *Company Operator* is also subject to the norms addressing this legal role ($SR_2 \xrightarrow{\text{play}} LR_1$). Interestingly, the social role *System Admin* may/may not be subject to norms depending on the different situations involved: its requirement of monitoring the accounts (G_1) can involve a simple monitoring of disk usage (s_1), *or* monitoring the activity of the accounts (s_3). In the latter case the *System Admin* is also subject to the norms of the *Data Processor* because the control of user activity involves processing of personal data ($s_7 \xrightarrow{\text{satisfy}} s_4$). So unless the requirements for the *System Admin* differently specify or avoid accessing user accounts, this social role will be also subject to the norms of the *Data Processor*.

Q2 *To which norms these roles must comply?* The situations representing the set of goals (G_1–G_6) make norms applicable. For example, the *Company Operator* must comply with the norms addressing the *Seller*: some norms become directly applicable when "a SIM-card is registered" (e.g., duty to issue a receipt or the duty to provide a phone number), some other norms are applicable when also some other conditions happen. For example, the duty to check the validity of a reimbursement (D_2) applies because the *Customer* has the right to request a reimbursement ($R_2 \xrightarrow{\text{endorse}} D_2$). The *Company Operator* must be able to include in its goals ways to comply with this norm. Similarly, the *Seller* must comply with the duty to actually reimburse the money to the *Customer* (D_3), when the reimbursement claim is valid. The *Company Operator* must comply with the norms addressing the legal role it plays. Evaluating to which norms the roles must comply with, amounts to identifying the applicable norms and in a second step making sure that the norms are not violated — which amounts to identifying the situation that the legal role (through the goals of a social role) should bring about.

Q3 *Which roles in the domain have fulfilled their responsibilities?* Evaluating whether a social role has fulfilled its responsibilities amounts to identifying and evaluating *how a role fulfills its responsibilities*. A social role is responsible for the situations describing the goals it holds, however it also accumulates the responsibilities of the legal role(s) it plays: once the consequent of the norms of the legal role it plays are brought about, the social role has fulfilled its responsibilities. The *SIM-user* for example must comply with the norms applying to the legal role of *Customer* and *Data Subject*. So given the initial set of goals, we know that the situations holding make several norms applicable, like the right for the *Data Subject* to express consent on data treatment (R_1). The social role has therefore fulfilled its responsibilities when this applicable norm is complied with. This right is complied with when the *Data Subject* expresses a consent (s_{14}), and this situation should be brought about by the *SIM-user* in the domain. Until it is included in the requirements that the *SIM-user* should express the consent ($s_{14} \xrightarrow{\text{satisfy}} G_x$, G_x being a new goal "Express consent for data", dotted element in figure 2), the social role will have not fulfilled its responsibilities. The explicit

indication that s_{14} is responsibility of the *SIM-user* identifies exactly which social role should include this responsibility in form of requirements. Moreover, the social role *Company Operator* is the one who must comply with the norms applying to the *Data Processor*, like the duty to log access to data is complied with when the access to the data is logged (s_{17}). However in this case it is not relevant for compliance *who* brings about *that* situation: s_{17} is not 'reserved' by any social role. So, as long as a social role includes it in its responsibility (e.g., by adding a new goal to the *System Admin*), the norm can be satisfied: both the legal role and the social role of *Company Operator* will have fulfilled their responsibilities. Should however the *System Admin* fail to bring about this situation, the responsibility of the violation of the norm would be still traced to its holder (the *Data Processor*, the *Company Operator*).

Q4 *If a role in the domain fulfills its responsibilities, what is the fulfillment value of the legal role it is playing?* When a social role fulfills its responsibility in the domain, it satisfies the situations it is responsible for — i.e., the situations describing the requirements it holds. For example, when the *SIM-user* fulfills its responsibilities in the model of figure 2, we can assert that the situations describing the goals G_5, G_6 are satisfied (s_{11}, s_{12}, s_{13}). These situations make applicable 3 norms for two legal roles (the *Data Subject* and *Customer*): R_1, R_2, R_3. These legal roles played by the *SIM-user* are fulfilled when the applicable norms are respected. R_1 and R_2 norms are complied with when the respective consequent is satisfied (s_{14}, s_{13}), while the compliance of R_3 depends on whether the *Seller* complies with his duty to reimburse money of a valid reimbursement claim ($D_3 \xrightarrow{\text{imply}} R_3$). By 'only' fulfilling its responsibility in the domain, the *SIM-user* does not also fulfill the responsibility of its legal counterpart as 1) some situations are not brought about 2) the compliance of a norm depends on the compliance of a norm of another role.

Compliance Discussion. Compliance of a set of requirements — expressed in terms of Goals — can be evaluated and explored using the Nòmos 3 modeling language and its reasoning capabilities. The identification of applicable norms and legal roles allows for example to identify requirements that should be revised in order not to make too many roles in the domain have to comply with norms: this is the case of the *System Admin* and whether it access the user account (aka, becomes a *Data Processor* or not). Secondly, our language allows to evaluate requirements compliance given the applicable norms, the situations that should be brought about to satisfy the norm, and the role — if a particular one is needed — that should bring about the situation. For example, in order for the *Data Processor* to comply with the duty to log the access to user's data, it is sufficient that the situation is brought about in the requirements. However, to comply with the *Data Subject* right to include its data, a consent should be given *only* from the *SIM-User* (s_{14}). In this way, by evaluating the satisfaction of the consequent of the applicable norms and the roles responsible for them, it is possible to identify the requirements compliance with a set of norms.

6 Related Work

Nòmos 3 is related to three research areas: role ontologies, normative multi-agent systems, and compliance of requirements. First, in the research area of ontology for roles, the work by Masolo et al. [9] and by F. Loebe [8] provide an important ontological characterization of the concept of Social Role. In the first work Masolo provides an extension of the DOLCE foundational ontology to deal with social concepts. In the second work, Loebe also provides a distinction between social and abstract roles. None of these works explicitly considers legal roles, which are instead characterized by Gangemi et al. [4]. Our characterization of the concepts of Legal and Social Roles is aligned with the principles laid down in these works.

In the AI field of Normative MultiAgent System (MAS) norms act as behavioural constraints that regulate and structure social order within a MAS. Typical problems addressed include the definition and derivation of those rules, also monitoring an agent's behaviour to determine compliance. For example, Bench Capon et al. [3] describes a method to dynamically assign norms to agents and, in line with our work they, also treat Norms as conditional elements. [11] proposes a normative extension to their framework for MAS based on Event Calculus which allows to model the responsibility of an agent depending on the role he plays. [15] takes a social perspective of MAS and proposes formal model for governance that provides similar type of compliance analysis similar to ours, in the domain of Socio Technical System with autonomous agents. [10] clarifies the concepts used in Normative MAS and illustrates how the so called *normative context* influences the actions that the autonomous agents can perform. The heavy-weight formal approach involved in most of these approaches makes it difficult to apply them to Requirement Engineering (RE).

In RE different solutions have been proposed for evaluating compliance. For example, Breaux et al. [1] use a text-based approach to extract and prioritize rights and obligations from legal texts and align them with a set of requirements. Darimont et al. [2] have used KAOS for the representation of elements extracted from legal texts. [5] also adopt a goal-based approach and provide an extension to the Goal-oriented Requirement Language to manage law and identify which requirements are not aligned. Siena et al. [14] also adopt a goal-modeling approach and introduce a 'dedicated' modeling language to provide a legal extension for the i* framework. [7] extended this framework by Siena et al. to represent non-compliance and used an argumentation-base approach to revise a set of requirements and achieve their compliance. Most of these approaches however tend to provide ad-hoc solutions for a specific requirement language or for a specific type of law and do not explicitly consider the attribution of responsibilities of roles or what happens when the wrong roles satisfy a norm.

7 Conclusions

In this paper we have presented Nòmos 3, a modeling language for evaluating compliance of roles in the domain of Requirement Engineering. Our propositional approach allows our modeling language to be an adequate lightweight solution

for requirement engineering as opposed to other modeling language for law. The readability and complexity of our models is an important limitation that we plan to investigate and improve in future work. Work in progress is dedicated to a methodology to support the analyst in revising a set of requirements, as well as implementing our primitives in a reasoning tool like DLV in order to support the methodology with a reasoning tool (like for Nòmos 2). Current work in progress is also dedicated to a tool-supported methodology for the semi-automatic generation of Nòmos 3 model. Further work will also be dedicated in investigating and expanding our language to answer questions regarding delegations (of responsibilities, requirements, ...).

Acknowledgments. This work has been supported by the ERC advanced grant 267856 "Lucretius: Foundations for Software Evolution" (April 2011 – March 2016).

References

1. Breaux, T.D., Vail, M.W., Antón, A.I.: Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations. In: RE 2006 (2006)
2. Darimont, R., Lemoine, M.: Goal-oriented analysis of regulations. In: ReMo2V, held at CAiSE 2006 (2006)
3. Derakhshan, F., Bench-Capon, T., McBurney, P.: Dynamic assignment of roles, rights and responsibilities in normative multi-agent systems. Journal of Logic and Computation 23(2), 355–372 (2013)
4. Gangemi, A., Sagri, M.-T., Tiscornia, D.: A constructive framework for legal ontologies. In: Benjamins, V.R., Casanovas, P., Breuker, J., Gangemi, A. (eds.) Law and the Semantic Web. LNCS (LNAI), vol. 3369, pp. 97–124. Springer, Heidelberg (2005)
5. Ghanavati, S., Amyot, D., Peyton, L.: Towards a framework for tracking legal compliance in healthcare. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 218–232. Springer, Heidelberg (2007)
6. Ingolfo, S., Jureta, I., Siena, A., Susi, A., Perini, A., Mylopoulos, J.: Legal compliance of roles and requirements. Tech. rep., University of Trento, Italy, tR14-03-001 (2012), `http://selab.fbk.eu/lawvariability/`
7. Ingolfo, S., Siena, A., Mylopoulos, J., Susi, A., Perini, A.: Arguing regulatory compliance of software requirements. Data & Knowledge Engineering 87 (2013)
8. Loebe, F.: Abstract vs. social roles-a refined top-level ontological analysis. In: Procs. of AAAI Fall Symposium Roles 2005 (2005)
9. Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gangemi, A., Guarino, N.: Social roles and their descriptions. In: Procs. of KR 2004, pp. 267–277 (2004)
10. Okuyama, F.Y., Bordini, R.H., da Rocha Costa, A.C.: Situated normative infrastructures: the normative object approach. J. Logic Comput. 23, 397–424 (2013)
11. Sadri, F., Stathis, K., Toni, F.: Normative kgp agents. Comput. Math. Org. Theor. 2006, 101–126 (2006)
12. Seipel, D., Minker, J., Ruiz, C.: A characterization of the partial stable models for disjunctive databases. In: ILPS, pp. 245–259 (1997)
13. Siena, A., Jureta, I., Ingolfo, S., Susi, A., Perini, A., Mylopoulos, J.: Capturing variability of law with Nòmos 2. In: ER 2012 (2012)
14. Siena, A., Mylopoulos, J., Perini, A., Susi, A.: Designing law-compliant software requirements. In: ER 2009, pp. 472–486 (2009)
15. Singh, M.P.: Norms as a basis for governing sociotechnical systems. ACM Trans. Intell. Syst. Technol. 5(1), 21:1–21:23 (2014)