Choosing Compliance Solutions through Stakeholder Preferences

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

[1] University of Trento, via Sommarive 14, Trento, Italy
{silvia.ingolfo,a.siena,jm}@unitn.it
[2] University of Namur, 8, rempart de la vierge, 5000 Namur, Belgium
ivan.jureta@fundp.ac.be
[3] FBK-Irst, via Sommarive 18, Trento, Italy
{susi,perini}@fbk.eu

Abstract. [**Context and motivation**] Compliance to relevant laws is increasingly recognized as a critical, but also expensive, quality for software requirements. [**Question/Problem**] Laws contain elements such as conditions and derogations that generate a space of possible compliance alternatives. During requirements engineering, an analyst has to select one of these compliance alternatives and ensure that the requirements specification she is putting together complies with that alternative. However, the space of such alternatives is often large. [**Principal ideas and results**] This paper extends Nòmos 2, a modeling framework for laws, to support modeling of and reasoning with stakeholder preferences and priorities. The problem of preferred regulatory compliance is then defined as a problem of finding a compliance alternative that matches best stakeholder preferences. [**Contribution**] The paper defines the concept of preference between situations and integrates it with the Nòmos 2 modeling language. It also presents a reasoning tool for preferences and illustrates its use with an extract from a use case concerning the Italian law on Electronic Health Record.

Keywords: Regulatory compliance, stakeholder preferences, models of law.

1 Introduction

We have entered an era where software quality is determined not only by the degree to which a software system meets its requirements (fitness-for-purpose), but also by the degree to which it complies with relevant norms (fitness-to-norms) [3]. There is now a rapidly growing number of laws and regulations world-wide that impacts on software systems, and requirements engineers are challenged to understand and analyze the various ways their systems can fulfill their requirements, while complying with all applicable laws.

Fitness-to-norms, or compliance, is usually understood as a binary criterion: either a system complies with a given law, or it is in violation. However, there can

J. Doerr and A.L. Opdahl (Eds.): REFSQ 2013, LNCS 7830, pp. 206–220, 2013.

be (and generally there are) multiple ways to comply with a given law because of variability elements contained in legal texts, such as conditions, exceptions, derogations, alternatives, cross-references, etc. Such elements allow alternative ways to comply, depending on which conditional elements apply for a system under design. This variability implies that there isn't a single compliance solution, but rather a space of compliance alternatives. While alternatives in law are equal to the legislator, they are not equal to stakeholders: some alternatives may fit better existing requirements, while others may cost less to comply with. In other words, if a software system has to comply with a given law, *how* it complies also defines how well stakeholder requirements are met. So the problem of ensuring regulatory compliance of requirements includes a search for the best way to comply. We define the Preferred Compliance Problem as the problem of finding the best compliance solution, given a law and a set of stakeholder preferences.

The main objective of this paper is to formulate and address the Preferred Compliance Problem. Our solution to the problem is based on the idea that stakeholder preferences drive the search in a space of compliant alternatives. Norms are modelled with Nòmos 2, a modelling framework tailored to law that supports reasoning about regulatory compliance of software requirements [22]. In this proposal, our norm models [22] are enriched with preferences between situations (partial states of the world) entailed by norms. Moreover, stakeholder assumptions can be expressed and included in the model as hard constraints. Models built in terms of such concepts are subsequently analyzed to find candidate compliance solutions. We acknowledge that the usefulness of the analysis of these models — like any engineering model — critically depends on its quality. Our proposal is illustrated through a small part of a real use case involving the Italian law on Electronic Health Record. A prototype tool is used to analyze Nòmos 2 models using disjunctive logic programming. Details about the tool are presented in a companion paper [10].

The rest of the paper is organized as follows. Section 2 recalls basic definitions of the Nòmos 2 conceptual modelling framework for laws, founded on the concepts of norm and situation. Section 3 provides a formulation of the Preferred Compliance Problem and defines the solution concept to this problem. An overview of a functioning prototype tool is presented in section 4, while section 5 illustrates our approach and the capabilities of the tool with a use case. Related work is discussed in section 6, while section 7 concludes the paper.

2 Modelling Law with Nòmos 2

Nòmos 2 is a modeling framework proposed in [22] that aims at capturing the variability of compliance alternatives for norms. Indeed, legal texts contain elements such as conditions, exceptions or derogations defining the *applicability* of alternative norms within a piece of law. In Nòmos 2 a Norm[1] is defined as

[1] 'Norm' refers to the concept, while lowercase 'norm' refers to an instance. Similarly with 'Situation'/'situation'.

a 5-tuple (*type*, *hol*, *ctrpart*, *ant*, *cons*), where *type* is the type of the Norm (e.g., duty or right); *hol* is the *holder* of the Norm, the role having to satisfy the Norm, if that Norm applies; *ctrpart* is the *counterpart*, the role whose interests are helped if the Norm is satisfied;[2] *ant* is the *antecedent*, the conditions to satisfy for the Norm to apply; *cons* is the *consequent*, the conditions to satisfy for the Norm to be complied with.

The applicability and satisfaction of a norm depend on situations which are satisfied, the idea being that if some situations are satisfied, the norm will apply, and when other situations are satisfied, the norm will be satisfied. Situations and norms are partial states of the world that we may know to hold (meaning the situation or norm is satisfied), not hold, or neither (when we can't conclude satisfaction or denial).

In our model, situations are linked to norms in terms of four relations: two relations for applicability (activate, block), and two relations for satisfiability (sat, break). The relation *activate* (resp. *block*), from a situation to a norm, means that if the situation is satisfied the norm is applicable (resp. not applicable). The relation *satisfy* (resp. *break*), from a situation to a norm or another situation, means that if the situation is satisfied the norm or the other situation is satisfied (resp. not satisfied). On top of these four basic relations we have three composite relations for norms.[3] The relation *derogate*, means that complying with the first norm makes the second norm not applicable. The relation *endorse*, means that complying with the first norm makes the second norm applicable. The relation *imply*, means that satisfaction of the first norm entails satisfaction of the second norm.

Norm models are used to reason about compliance of requirements. Consistent sets of requirements satisfy one or more situations, and according to how situations are related to norms they make certain norms applicable or satisfied. Situations are labeled as ST (Satisfiability True) if there is evidence that they are satisfied; as SF (Satisfiability False) if there is evidence that they are not satisfied; as SU (Satisfiability Unknown) if there is no evidence or no decision is made. The relations of our model act as label-propagation channels that propagate labels from their source (situation or norm) to their target. Depending on the pair of labels associated to a norm (for applicability and satisfiability) the norm may be compliant, violated, tolerated or inconclusive. Anytime a norm is not applicable it is 'tolerated', when the applicability is not known it is 'inconclusive'. When a norm is applicable it can be either 'complied with', 'violated' or 'tolerated' depending on the satisfiability value and on the type of norm (duty/right).[4]

Use of the language. Nòmos 2 models allow us to represent fragments of laws or regulations by representing the different conditions and rules described by

[2] In the rest of the paper we use the term 'legal subject' to refer to the role holder and counterpart.

[3] These relations, *shortcuts* in [22], can be defined as a composition of the four basic relations.

[4] For more details see [22].

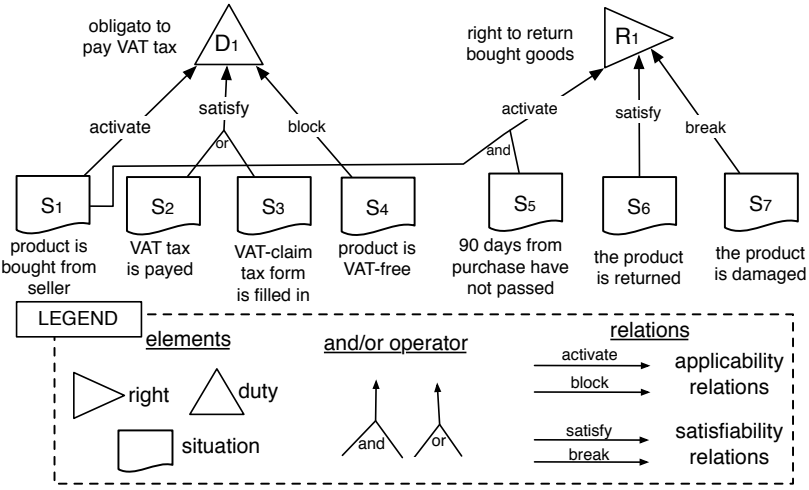

Fig. 1. An example of our modeling language and its graphical syntax

a law or regulation, and the alternative ways to comply with it. For example when the legal subject 'customer' is in the situation 'product is bought from seller', then several legal clauses regarding reimbursement, tax-payment and tax-declaration apply. We say that a legal subject *complies* with a clause if that clause applies to that legal subject and the subject satisfies that clause. If the customer pays the taxes on the product — i.e., the legal subject is in the situation 'VAT-tax on product is paid' — then the customer is *complying* with the clause about paying VAT taxes. Leveraging on applicability and satisfiability of the different situations that are holding, it is possible to identify how to comply with a law in different ways (e.g., by paying the VAT tax, by buying a VAT-free product, ...).

In figure 1 we show an example with the graphical syntax of our modeling language, applied to an hypothetical tax law. This simple example shows how, when a product is bought from a seller ($S_1 \xrightarrow{\text{activate}} D_1$), the duty to pay taxes is activated. You comply with this duty either by paying the taxes ($S_2 \xrightarrow{\text{satisfy}} D_1$) or by filling in the VAT-claim tax form ($S_3 \xrightarrow{\text{satisfy}} D_1$). This option is represented by an or-operator that satisfies the duty. VAT-free product are untaxed by definition, so if the product is VAT-free (i.e. S_4 is ST), the duty is no longer applicable ($S_4 \xrightarrow{\text{block}} D_1$). For purposes of this example, we consider that most stores have a return policy for which you have up to 90 days to return the product bought in that store. These two situations (S_1, S_5) activate the right to return the good, which in turn is complied with (*exercised*) when the product is returned ($S_6 \xrightarrow{\text{satisfy}} R_1$). However, if the product is damaged, then you can not exercise the right ($S_7 \xrightarrow{\text{break}} R_1$).

Notice that this simple model has different compliance solutions (for example, paying VAT tax, filling in the VAT-claim tax form, buying VAT-free products,

etc.). To generalize, since in a norm model situations constitute free variables, and their possible labels (satisfied, not satisfied, undefined) determine the number of permutations on the model, the total amount of possible permutations is 3^s, where s = number of situations. The total number of compliance solutions is between 0 and 3^s, and depends on the topology of the model. In any case, since the number of possible permutations grows exponentially, the number of compliance alternatives also grows quickly. Given a large number of compliance alternatives, it becomes necessary to compare them, so as to help identify one that best responds to stakeholder expectations.

3 The Preferred Compliance Problem

This section defines the *Preferred Compliance Problem* (PCP hereafter), as the problem of identifying alternative ways to comply with applicable Norms, and comparing these alternatives on the basis of stakeholder preferences. Since there can be different criteria for comparison and as their relative importance can vary across stakeholders and systems engineering projects, PCP does not prescribe a specific procedure or rule for ranking alternatives.

Compliance can be understood as a relation between a design of a system-to-be, environment conditions in which the system-to-be will operate, the requirements it will satisfy, *and* a set of applicable Norms. To get to the PCP, we will start from the known Zave & Jackson [24] (Z&J) requirements problem formulation, which abstracts from the issue of compliance. We will suggest below how to extend Z&J requirements problem to the Compliance Problem (CP) using Nòmos 2, explain where variability is in the CP, define the preference relation needed to compare alternatives in the CP, which will finally lead us to state the PCP.

Z&J requirements problem is: Given domain assumptions D and requirements R, find a design of the system-to-be, such that its specification S is consistent with D and R, and together with D satisfies R, i.e., $D \cup S \vdash R$, where $\vdash$ is the consequence relation of classical (propositional or first-order) logic.

How does Z&J Relate to Situations and Norms? Applicable Norms, to which a system-to-be needs to comply, are a function of Situations that will occur in the environment in which the system-to-be will operate. In other words, which Norms apply depends on what the system does, and on the environment in which it does it. Since the system is designed to satisfy requirements within that environment, it follows that applicable Norms will depend on all three components of the Z&J requirements problem — the requirements R, the conditions in the environment D, and the design of the system-to-be S.

Given this dependency, we can write the function $\mathrm{Sit}(D, R, S)$ for the set of Situations that can occur if we choose the design S of the system-to-be, for the requirements R, and domain assumptions D. If we choose some different D_j, R_j, S_j, then $\mathrm{Sit}(D, R, S)$ is not necessarily same as $\mathrm{Sit}(D_j, R_j, S_j)$, i.e., different systems, environments, requirements result, quite expectedly, in different Situations.

We know from Nòmos 2 that whether a Norm applies depends on the Situation that occurs. Therefore, the set of applicable Norms, given all potentially applicable Norms N, is returned by $\mathrm{App}(N, \mathrm{Sit}(D, R, S))$. Now, we need to distinguish two kinds of Norms in $\mathrm{App}(N, \mathrm{Sit}(D, R, S))$:

- Norms which play a role analogous to domain assumptions in the Z&J requirements problem, in the sense that Situations should be not be in conflict (i.e., Nòmos 2 break or block relations) with these Norms. We denote these $\mathrm{App}_1(N, \mathrm{Sit}(D, R, S))$.
- Norms which have a role analogous to new requirements in the Z&J requirements problem, requiring us to ensure that we satisfy Situations in which these Norms are satisfied. We will denote them $\mathrm{App}_2(N, \mathrm{Sit}(D, R, S))$.

What is the Compliance Problem? Following the argument above, CP is: Given domain assumptions D, requirements R, and Norms N, find a design of the system-to-be, such that its specification S ensures the following conditions are satisfied:

1. $D \cup \mathrm{App}_1(N, \mathrm{Sit}(D, R, S)) \cup S$ is consistent;
2. $D \cup \mathrm{App}_1(N, \mathrm{Sit}(D, R, S)) \cup S \vdash R \cup \mathrm{App}_2(N, \mathrm{Sit}(D, R, S))$.

Since Situations depend on D, R, and S, we can reformulate the CP as follows: *Given a set of potentially satisfiable Situations $\mathcal{S}$, find a set of Situations $X \in \wp(\mathcal{S})$, such that X satisfies $App_1(N, Sit(D, R, S)) \cup App_2(N, Sit(D, R, S))$*, where $\wp(\mathcal{S})$ is the powerset of Situations.

The limitation of this CP is that, when there are alternative sets of Situations which satisfy the said conditions, the CP does not compare these alternatives. This is misleading, as it makes the alternatives, which we call below Candidate Compliance Solutions, appear equally desirable, yet they are not: some of them will be produced by systems that satisfy more desirable requirements than others.

What is a Candidate Compliance Solution? It seems, from the CP, that a solution to the CP is a set of Situations. We prefer, however, to keep also in that solution, the Norms satisfied by these Situations. This leads us to the following *Candidate Compliance Solution* concept: A Candidate Compliance Solution i, to a CP instance, is a pair (X_i, N_i), such that:

1. $X_i \in \wp(\mathcal{S})$ is a set of Situations,
2. $N_i = \mathrm{App}_1(N, \mathrm{Sit}(D, R, S)) \cup \mathrm{App}_2(N, \mathrm{Sit}(D, R, S))$,
3. X_i satisfies all Norms in N_i.

How to Compare Candidate Compliance Solutions? To capture the information that some Candidate Compliance Solutions are more desirable than others, we add to Nòmos 2 a set of binary reflexive, antisymmetric and transitive relations $\leq_C \in \mathcal{S} \times \mathcal{S}$, each $\leq_C$ defining a partial order on Situations. Informally, we call these relations *preference relations*, and we read $\phi \leq_C \psi$ as "ψ is at least as desirable as ϕ according to criterion C". We let $\phi =_C \psi$ abbreviate

"$\phi \leq_C \psi$ and $\psi \leq_C \phi$", so that $\phi <_C \psi$ abbreviates "$\phi \leq_C \psi$ and not $\phi =_C \psi$", and informally reads "ψ is strictly more desirable than ϕ according to criterion C". Each criterion C defines a partial order over Situations. Note that adding preference relations to Nòmos 2 does not influence the satisfaction values, and other features of that language.

Preference relations allow us to record relative desirability of stakeholders between Situations, according to different criteria for comparison. Let $\mathcal{C}$ denote the set of all criteria. We can further add relations between criteria, to help comparisons. We can define a hierarchy of domain-specific criteria for comparison, such as, for example: Criterion *Cost* is an aggregate of criteria *Production cost*, *Infrastructure cost*, *Transportation cost*, etc. Such a structuring can help define aggregation functions and/or procedures to automatically rank alternative sets of Situations.

We do not discuss how preferences are negotiated between stakeholders, since different stakeholders can have opposing preferences over the same criteria. Both the definition of aggregation functions of preferences over criteria, and the negotiation of conflicting preferences are outside the scope of this paper.

What is the Preferred Compliance Problem? The presence of two or more Candidate Compliance Solutions, to a given CP, and the availability of preferences leads to the Preferred Compliance Problem. In contrast to CP, where the aim is to identify a (or at least one) Candidate Compliance Solution, PCP requires that preference be used to select one Candidate Compliance Solution, as *the* Compliance Solution to the PCP. We state the PCP as follows.

> *Preferred Compliance Problem:* Given a set of potentially satisfiable Situations $\mathcal{S}$, find a set of Situations $X \in \wp(\mathcal{S})$, such that:
> 1. X satisfies $\mathrm{App}_1(N, \mathrm{Sit}(D, R, S)) \cup \mathrm{App}_2(N, \mathrm{Sit}(D, R, S))$,
> 2. there is no set of Situations X' such that the Candidate Compliance Solution (X', N) ranks higher than Candidate Compliance Solution (X, N), according to a given ranking function r, which returns a total order over all Candidate Compliance Solutions.

Informally, PCP requires us to compare Candidate Compliance Solutions according to preferences, and to select one of the Candidate Compliance Solutions. Above, we assume that there is a ranking function r, which establishes from a set of preferences over Situations, a total order over all Candidate Compliance Solutions. Note that r need not be given as a mathematical function, but can be defined as a process that results in a ranking (for example, the process of asking stakeholders to vote for Candidate Compliance Solutions).

4 Automated Reasoning

As the size of a reasoning problem grows, it becomes harder for humans to deal with its complexity. It is therefore important to support automated reasoning of

large models in order to check for important and interesting properties, such as consistency. The overhead of building these kinds of model — which can be reduced with the support of automated tools such as [14] — is indeed compensated by the consistent and completeness of its automated analysis.

In order to support analysts to solve the PCP, we are developing a tool called NRTool.[5] It essentially performs bottom-up and and top-down analysis to search for a Nòmos 2 model for Candidate Compliance Solutions, and rank them according to the preference function r.

The space of situations to be analyzed in a Nòmos 2 model can become intractable (3^s, where s corresponds to the number of situations in the model), so in our implementation of the problem we give the analyst the possibility to specify *assumptions*: strong constraints on the satisfaction value of some specific situations in the model. The use of assumptions helps the tool cut down the space of possible solution, and allows the analyst specify known facts that must hold in every Candidate Compliance Solutions.

The NRTool works by translating the PCP into a disjunctive Datalog [1, 17] program. Disjunctive Datalog is a declarative logic language and a deductive system where facts and deduction rules are expressed in the logic language. Disjunctions may appear in the rule heads to allow multiple alternative consequences to be drawn from a rule. Situations and norms are mapped onto Datalog facts, while relations are mapped onto deduction rules. NRTool relies on DLV [2] as Datalog reasoning engine. DLV further extends disjunctive Datalog to also support weak constraints, priorities for their satisfaction, and costs for their violation. These extensions allow us to represent the *preferences* on the satisfaction value of pairs/group of situations (represented as weak constraints and priorities on them), and to have an evaluation of the costs to be payed for the set of violated preferences. Concerning the search techniques and heuristics used by DLV, it implements a back search similar to SAT algorithms and advanced pruning operators, (look-ahead and look-back techniques) for model generation, and innovative techniques for answer-set checking.

An important characteristic of DLV is the possibility to obtain the complete set of solutions (models) produced by a set of predicates and assignments to the variables or to prune the set of models depending on the preferences specified by the decision makers. We exploit these features indeed to generate and prune the alternatives that fits the preferences specified in the input compliance problem. With the specification of assumptions (fixed assignments to some situations of the model), we further help this pruning mechanism.

NRTool works as depicted in figure 2: using a custom input language, the analyst provides the description of the PCP to be solved: (i) a set of *preferences* between pairs or groups of situations, and (ii) a *query* about norms to be complied with. Additionally he can specify the value of some situations that are known or hypothesized to be true or false (*assumptions*). A Nòmos 2 model of the law is then added to this specification of the PCP problem, and it is then

[5] The tool is in a prototype development phase and can be made available upon request. As soon as a stable release will be ready, it will be published online.

converted by the tool into Datalog.[6] Finally, the tool parses the output of the reasoning engine and presents it to the user. The use of preferences is used to return the best solutions to the problem. In the next section we will see how the tool can help us find the best Candidate Compliance Solutions.

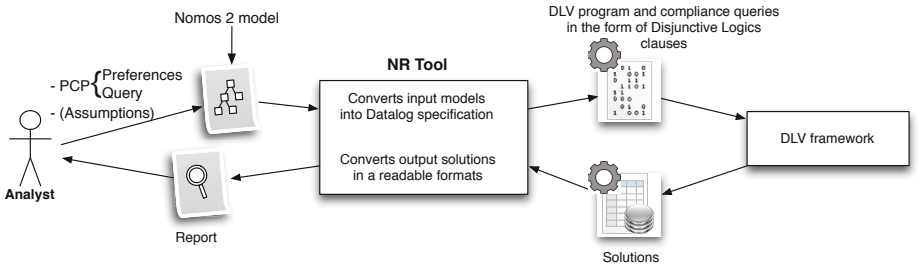

Fig. 2. NRTool transforms the input provided by the analyst (PCP and assumptions) into a disjunctive Datalog program, and reports the output of the Datalog engine back to the analyst

5 Use Case: The CSS Project and the Italian Law on Electronic Health Record

In a recent industrial case study [11], we have been involved in the analysis of the Italian law on Electronic Health Record (EHR).[7] The context of the study was that of an Italian organization involved in the design and development of a project called CSS (*Cartella Socio Sanitaria* — Electronic Social/Health Record) aimed at monitoring healthcare and social processes in Trentino, a region in northern Italy. The main goal of the CSS project was to support sharing the information of the patients among the different health care entities involved in the project (e.g., hospitals, family doctors, and other agencies for social, mental health and other medical services). The CSS system needed therefore to be designed and created in respect of the principles set forth by the Guidelines on EHR and the Italian Privacy Law.

During our study we have closely analyzed the problem of complying with section 3.10 and 3.11 of the law. In this section it is explained how a patient has the the right not to include some information in the EHR system, and also how the patient has the right to have some information removed from his record (called 'blanking' right). Moreover this section shapes an important duty for the EHR system which is forbidden to notify the event that some information has been blanked in a patient's record. In the context of CSS, the purpose of this EHR system was indeed to share the patient information among all authorized entities. To achieve this goal, it was envisaged for the system not to directly share

[6] We assume that a Nòmos 2 model of the law is given.

[7] `http://www.garanteprivacy.it/garante/doc.jsp?ID=1634116`

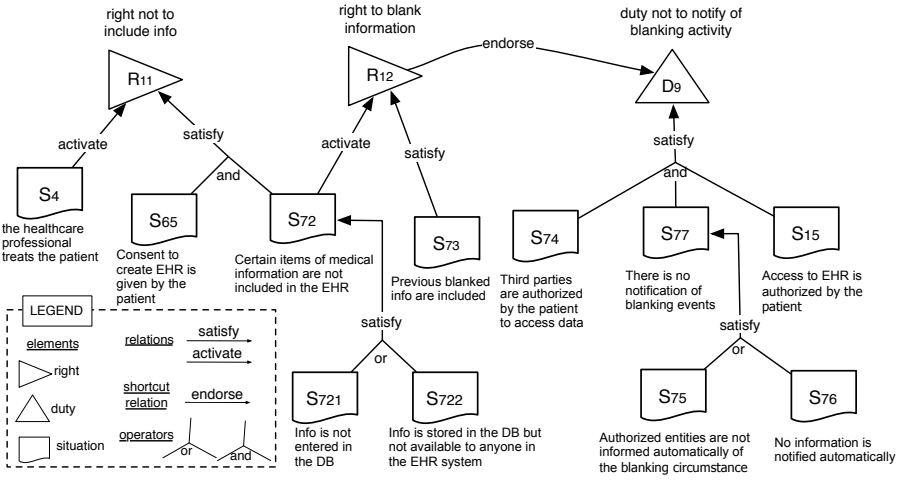

Fig. 3. Nòmos 2 model of section 3.10/3.11 of the Italian Law on EHR

the patient information using a centralized database, but rather to only share notifications regarding the metadata of the information. This way the original information would have remained stored in the database of the entity creating it. The need to comply with sections 3.10/3.11 of the Italian law was therefore opening the design possibilities.

In figure 3 we have represented the graphical model of sections 3.10/3.11 of the Italian Law on EHR that we are considering. This Nòmos 2 model shows how the three norms considered are activated ($S_4 \xrightarrow{\text{activate}} R_{11}$, $S_{72} \xrightarrow{\text{activate}} R_{12}$, $R_{11} \xrightarrow{\text{endorse}} D_9$),[8] how they can be satisfied through the satisfy relation, and how the and or-decomposition can help opening the possibilities for complying with the norms. Each of the 11 situations represented in the model of figure 3 can have three values (ST, SF, SU): we therefore have a total of $3^{11} = 177147$ possible models.[9] In the following paragraphs we will see how the tool can help us reduce this space of alternatives and find the best solutions to the PCP.

The main objective of the CSS project was indeed to manage information sharing among the entities involved. All the healthcare entities involved in the project provided some medical services to the patients, so one first assumption was that S_4 is satisfied (i.e., $\text{sat}(S_4) = \text{ST}$). Moreover, all entities responsible for collecting the patients' information were assumed to have the proper consent from the patient to create and access their info in the EHR ($\text{sat}(S_{65}) = \text{sat}(S_{15}) = \text{ST}$). Given these constraints, we have that one of the input to the tool is the assumption $a_1 = \{sat(S_4) = \text{ST}, sat(S_{65}) = \text{ST}, sat(S_{15}) = \text{ST}\}$. For example, only

[8] The endorse relation ($\text{Norm}_1 \xrightarrow{\text{endorse}} \text{Norm}_2$) is a shortcut relation meaning that when the first norm is activated and satisfied, the other one is activated.

[9] Depending on the topology of the model, some alternatives are not evaluated by the reasoner as they are not possible solutions (e.g., S_{72} is SF, while S_{721} is ST).

Table 1. The 4 Candidate Compliance Solutions solutions to the PCP problem that rank best for the example in figure 3. In the PCP problem all norms are asked to be tolerated or complied with, two preferences are expressed ($pref_1, pref_2$), and some assumptions (a_1).

	S_4, S_{65}, S_{73}, S_{74}, S_{75}, S_{15}, S_{722}	S_{76}	S_{721}	R_{11}, R_{12}, D_9
1:	ST	SU	SU	*com*
2:		SU	SF	
3:		SF	SU	
4:		SF	SF	

with the specification of this assumption and asking the tool to return all possible model where the assumption holds, the space of alternatives drops to 619.[10]

In this scenario, when we *query* the model and ask for all three norms to be complied or at least tolerated (so, both *tol* and *com* value are acceptable) we obtain 25 possible models. In order to further reduce the space of alternatives and return only the best solutions, we introduce *preferences*. In the context of CSS, patient information was maintained in the database of the healthcare entity creating the data. These entities evaluated that S_{721} ("Information not entered in the Database") was a "more expensive" operation than S_{722}, as it would have implied the need of an dedicated operator to add the information in a second moment. Moreover, the possibility of not-notifying in automatic any information (S_{76}) was considered as it would have allowed each entity in the system to autonomously evaluate on a case-by-case basis when/which information to notify. This option though was also considered to have higher costs as it also would have relied on an extra-operator in the process, making it a less manageable and flexible approach. In this context one set of preferences can then be expressed as $pref_1 = \{S_{721} >_{cost} S_{722}\}$ and $pref_2 = \{S_{76} >_{cost} S_{75}\}$, indicating the preferences with respect to the cost criterion.

The specification of these preferences can then be included in the input to the NRTool together with the assumption (a_1) and the query with the compliance values (all norms *tol* or *com*) to reduce the space of alternatives. In table 1 we show the 4 Candidate Compliance Solutions to the PCP problem returned by the NRTool that rank best with respect to the specified preferences. All Candidate Compliance Solutions presented in table 1 rank best among the other possible solutions to the model: they all have the desired compliance value for the norms, they rank best with respect to the specified preferences, and they satisfy the assumption. For example we can see how in the first solution, the assumption are respected: the healthcare entities provide medical service to the patient, and the consent to create and access the EHR is given (S_4, S_{65}, and S_{15} are satisfied). To respect the indicated preferences, we have that there is no automatic notification of a blanking event ($\text{sat}(S_{75}) = \text{ST}$), and 'inaccessible' information is safely stored in the DB ($\text{sat}(S_{722}) = \text{ST}$). Thanks to the label propagation rules encoded in the input to the reasoner, the NRTool can evaluate

[10] So in this case the query is for model where the norms can have any value and no preferences are expressed.

that the three norms R_{11}, R_{12}, D_9 are activated and satisfied, ergo, they are complied with.

What lies behind these four best solutions, is how all the other assignments to the Nòmos 2 model in figure 3 are either (a) less desirable (w.r.t. $pref_1$ and $pref_2$), or (b) do not satisfy the query (all norms complied with or tolerated), or (c) they violate the assumption a_1. The tool uses this information to indeed prune and rank the space of alternative solutions to the Nòmos 2 model in order to find the best ones.

For example, a less desirable Candidate Compliance Solution would be one were the assumption are respected (S_4, S_{65}, and S_{15} are satisfied), also S_{72} and S_{677} are satisfied, while all other situations are unknown. The norms respect the query (they are evaluated as complied with), though the two most desired situations (S_{721} and S_{76}) are not satisfied, therefore making the solution less desirable.

Similarly there are assignment to the situations of the Nòmos 2 model that are desirable but that violate one of the norms. For example one could consider the model where the healthcare entities provide medical service to the patient, the consent to create and access the EHR is given (S_4, S_{65}, and S_{15} are satisfied), the info is stored in the database but not accessible (S_{722} is satisfied), and authorized entities are not automatically notified of the blanking right (S_{722}) is satisfied). However, the patient did not authorize third parties to access the data, therefore S_{74} is not satisfied ($\text{sat}(S_{74}) = \text{SF}$). In this possible model the preferred situations hold and the assumption are met, but the model violates the duty D_9.

Lastly, the specification of the assumptions from the analyst allows the tool to prune the solution space and return only candidate solutions that do not violate the values expressed in the assumption. For example, consider one of the four best solutions of table 1, where instead S_4 is not satisfied ($\text{sat}(S_4) = \text{SF}$). This scenario represents an assignment to the model that respects preferences and queried norm values (R_{11} is evaluated to tolerated), but it violates the assumption that the healthcare treat the patient, therefore making it a useless solution to the analyst.

We acknowledge as limitation and as part of our future work the challenge to collect these preferences in a bigger model/scenario. Also, the effectiveness of our method and its scalability are currently under investigation in our future work.

6 Related Work

In this work we have proposed an extension to the Nòmos 2 modeling framework [22], where we exploit the notion of stakeholder preference as an heuristic to select a minimal set of compliance alternatives for the design of a compliant requirement model. In requirement engineering the notion of preference and preference-based prioritization is generally used to characterize different decision making processes. In Techne [12] a stakeholder preference over two requirements is defined as a binary relationship between the two elements representing them. In this framework, preferences provide criteria for the comparison of candidate

solutions to a requirements problem. In [15] the term stakeholder priority is instead used to refer to this notion, while the concept of preference is used to indicate a "nice-to-have" property. This formulation allows exploiting planning techniques to build solutions to requirements problems that satisfy mandatory requirements and the preferences to a different degree. Elicitation of stakeholder preferences to prioritize set of requirements for the purpose of release planning is a key issue, as discussed in [19], and calls for specific techniques to keep as lower as possible the preference elicitation effort by stakeholders, still resulting in an satisfactorily ranking of the candidate requirements.

The problem of regulatory compliance is being investigated for several years now in the requirement engineering community. A recent survey review [8] summarizes some of the main proposals concerning methods and techniques to address regulatory compliance with goal-oriented frameworks. In this context for example, Darimont and Lemoine have used Kaos to represent objectives extracted from regulation texts [6]. Ghanavati et al. [7] use URN (User Requirements Notation) to model goals and actions prescribed by laws. Likewise, Rifaut and Dubois use i^* to produce a goal model of the Basel II regulation [20]. Goal-oriented approaches are useful approaches for modeling norms when their complexity is small enough to be reduced to goal relations. However, when the notion of applicability is needed or the variability of the law becomes more prominent, goal-oriented techniques fail in capturing its effects on reasoning about alternatives [21]. Also recently Tawhid et al. [23] have proposed an new approach in dealing with the problem of compliance by enriching a GRL model of requirements with qualitative indicators.

Among the challenging problems these approaches need to cope with are the complexity, ambiguity, variability and evolvability of the law. In fact, as pointed out in a recent case study analysis in the transportation domain [18], these issues can indeed become particularly critical in complex application domains.

Focusing on the problem of complexity and ambiguity of regulatory code, recent works propose methodologies supporting the understanding of legal documents for the purpose of software requirements analysis. For instance in [4] it is described a systematic process called semantic parametrization, which consists of identifying in legal text restricted natural language statements (RNLSs) and then expressing them as semantic models of rights and obligations (along with auxiliary concepts such as actors and constraints). Heuristics are created to systematically convert unstructured legal texts into structured artifacts [14]. Artifacts are then combined into a frame-based method for manually acquiring legal requirements from regulations. Such approach has been used as a basis for tool-supporting the identification of requirements in legal documents. [5] focuses on supporting software developers while analyzing regulatory codes with the aim of identifying sections that are relevant to contractual and product level requirements. This task is formulated in terms of a traceability problem which is addressed exploiting machine-learning techniques, and combined with web-mining features to reconstruct the original trace query. The approach is illustrated using the HIPAA security rule. [13] proposes a framework that supports analyzing the

compliance of legacy Information Systems, which rests on the alignment of a model of the transactions in the legacy system with an ontology of the laws that regulated the IS domain. This law ontology explicits the organizational roles, which correspond to the legal subjects of the laws governing the IS domain, with the domain artifacts and processes under their responsibility. Aligning and reconciling requirements from multiple jurisdictions is one of the problem that arises because of the variability of a law from country to country [9]. In this work the authors propose an approach to identify similarities and differences between pairs of requirements. This is achieved by comparing variants of norms — encoded in a specific legal requirements specification language — with respect to a suitable set of metrics. Moreover, the problem of regulatory evolution and its impact on compliance requirements has been recently investigated in [16], where the authors suggest a taxonomy of legal cross-reference of the HIPAA Privacy Rule that can be used to identify possibly conflicting requirements. As a result of the proposed strategies to solve these conflicts, software engineers are guided to build requirement-models towards the more stable sections of the rule, thus limiting the impact on the deployed software of possible changes in norms.

7 Conclusions

In this paper we have defined the Preferred Compliance Problem (PCP) for legal alternatives impacting software requirements. Stakeholder preferences are used to compare the desirability of possible solutions to a compliance problem. Apart from defining the problem, our contributions include a prototype tool for reasoning with preferences, as well as a use case where the tool is applied to model and analyze an Italian law on Electronic Health Records. Because of space limitations, only parts of the use case are presented herein. Ongoing work is devoted to the consolidation of the tool, as well as experimentally evaluating its scalability and effectiveness on larger models. We're working towards revising the requirement process in order to exploit the use of these variability models (see [11] for some preliminary results in this area).

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Alviano, M., Faber, W., Leone, N., Perri, S., Pfeifer, G., Terracina, G.: The disjunctive datalog system DLV. In: de Moor, O., Gottlob, G., Furche, T., Sellers, A. (eds.) Datalog 2010. LNCS, vol. 6702, pp. 282–301. Springer, Heidelberg (2011)
3. Bray, I.: An Introduction to Requirements Engineering. Addison-Wesley (2002)
4. Breaux, T., Antón, A.: Analyzing regulatory rules for privacy and security requirements. IEEE Trans. Softw. Eng. 34, 5–20 (2008)
5. Cleland-Huang, J., Czauderna, A., Gibiec, M., Emenecker, J.: A machine learning approach for tracing regulatory codes to product specific requirements. In: Kramer, J., Bishop, J., Devanbu, P.T., Uchitel, S. (eds.) ICSE (1), pp. 155–164. ACM (2010)

6. Darimont, R., Lemoine, M.: Goal-oriented analysis of regulations. In: ReMo2V, held at CAiSE 2006 (2006)
7. 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. LNCS, vol. 4495, pp. 218–232. Springer, Heidelberg (2007)
8. Ghanavati, S., Amyot, D., Peyton, L.: A systematic review of goal-oriented requirements management frameworks for business process compliance. In: RELAW 2011, pp. 25–34. IEEE (2011)
9. Gordon, D.G., Breaux, T.D.: Reconciling Multi-jurisdictional Legal Requirements: A Case Study in Requirements Water Marking. In: RE 2012. IEEE (2012)
10. Ingolfo, S., Siena, A., Jureta, I., Susi, A., Perini, A., Mylopoulos, J.: Modeling and reasoning with stakeholder preferences among legal alternatives. Submitted to CAISE13 (2012)
11. Ingolfo, S., Siena, A., Mylopoulos, J., Susi, A., Perini, A.: Arguing regulatory compliance of software requirements. Accepted for publication in Data & Knowledge Engineering, DKE (2012), http://dx.doi.org/10.1016/j.datak.2012.12.004
12. Jureta, I., Borgida, A., Ernst, N.A., Mylopoulos, J.: Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In: RE 2010, pp. 115–124. IEEE Computer Society (2010)
13. Khadraoui, A., Leonard, M., Thi, T.T.P., Helfert, M.: A Framework for Compliance of Legacy Information Systems with Legal Aspect. In: Gronau, N. (ed.) AIS Transactions on Enterprise Systems, vol. 1. GITO Publishing GmbH (2009) ISSN 1867-7134
14. Kiyavitskaya, N., Zeni, N., Breaux, T.D., Antón, A.I., Cordy, J.R., Mich, L., Mylopoulos, J.: Automating the extraction of rights and obligations for regulatory compliance. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 154–168. Springer, Heidelberg (2008)
15. Liaskos, S., McIlraith, S.A., Sohrabi, S., Mylopoulos, J.: Representing and reasoning about preferences in requirements engineering. Requir. Eng. 16(3), 227–249 (2011)
16. Maxwell, J.C., Antón, A.I., Swire, P.: Managing Changing Compliance Requirements by Predicting Regulatory Evolution: An Adaptability Framework. In: RE 2012. IEEE (2012)
17. Minker, J.: Overview of disjunctive logic programming. Ann. Math. Artif. Intell. 12(1-2), 1–24 (1994)
18. Nekvi, M.R.I., Madhavji, N.H., Ferrari, R., Berenbach, B.: Impediments to requirements-compliance. In: Regnell, B., Damian, D. (eds.) REFSQ 2011. LNCS, vol. 7195, pp. 30–36. Springer, Heidelberg (2012)
19. Perini, A., Susi, A., Avesani, P.: A Machine Learning Approach to Software Requirements Prioritization. IEEE Transactions on Software Engineering (2012) (to appear)
20. Rifaut, A., Dubois, E.: Using goal-oriented requirements engineering for improving the quality of iso/iec 15504 based compliance assessment frameworks. In: RE 2008, pp. 33–42 (2008)
21. Siena, A., Ingolfo, S., Susi, A., Jureta, I., Perini, A., Mylopoulos, J.: Requirements, intentions, goals and applicable norms. In: ER Workshops, pp. 195–200 (2012)
22. Siena, A., Jureta, I., Ingolfo, S., Susi, A., Perini, A., Mylopoulos, J.: Capturing variability of law with Nòmos 2. In: ER 2012 (2012)
23. Tawhid, R., et al.: Towards outcome-based regulatory compliance in aviation security. In: RE 2012, pp. 267–272 (2012)
24. Zave, P., Jackson, M.: Four dark corners of requirements engineering. ACM Trans. Softw. Eng. Methodol. 6(1), 1–30 (1997)