A Method to Acquire Compliance Monitors from Regulations

Travis D. Breaux
Institute for Software Research
School of Computer Science
Carnegie Mellon University
breaux@cs.cmu.edu

Abstract—**Developing software systems in heavily regulated industries requires methods to ensure systems comply with regulations and law. A method to acquire finite state machines (FSM) from stakeholder rights and obligations for compliance monitoring is proposed. Rights and obligations define what people are permitted or required to do; these rights and obligations affect software requirements and design. The FSM allows stakeholders, software developers and compliance officers to trace events through the invocation of rights and obligations as pre- and post-conditions. Compliance is monitored by instrumenting runtime systems to report these events and detect violations. Requirements and software engineers specify the rights and obligations, and apply the method using three supporting tasks: 1) identify under-specifications, 2) balance rights with obligations, and 3) generate finite state machines. Preliminary validation of the method includes FSMs generated from U.S. healthcare regulations and tool support to parse these specifications and generate the FSMs**.

I. INTRODUCTION

Software engineering is concerned with automating tasks within and across the software development lifecycle. Software requirements are one of the first artifacts to enter this lifecycle. Due to their informal specification in natural language, they are difficult to manipulate and integrate directly into the verification and testing of large software systems – the goal of requirements monitoring. In highly regulated industries within the United States, such as healthcare and finance, requirements monitoring is necessary to ensure systems comply with the law. For example, the U.S. Health Insurance Portability and Accountability Act[1] (HIPAA) of 1996 and Gramm-Leach-Bliley Act[2] of 1999 require organizations to implement programs to develop and monitor legal compliance with security and privacy policy and regulations. *Legal compliance* refers to an organization's "ability to maintain a defensible position in a court of law" [9]. Legal compliance consists of maintaining clear evidence that demonstrates both *due diligence*, which means "reasonable efforts that persons make to satisfy legal requirements", and *good faith,* which means "faithfulness to one's duty or obligation" [14]. In this paper, I propose a manual method to systematically acquire finite state machines (FSM) from semantic models of stakeholder rights and obligations. I illustrate the method using legal requirements extracted from the HIPAA Privacy Rule during a prior case study [7] and with an attention to detail so that others may replicate and expound upon this method.

The FSMs correlate real world events (the transitions between states) to stakeholder rights and obligations (the states). To evaluate risk and compliance, software developers can map these events to requirements and design specifications. Systems developers can use these event mappings to focus their verification efforts on those components most at-risk for non-compliance. After deployment, the FSMs can operate as compliance monitors by instrumenting software components to provide records of these events and demonstrate an organization's ability to monitor compliance failures; a good faith practice. As systems evolve to adapt to new organizational needs, these monitors ensure that new changes to existing software systems continue to operate within the regulatory framework of stakeholder rights and obligations.

It is estimated that healthcare organizations have spent up to $17.6 billion over the last few years to bring their systems and procedures into compliance with the HIPAA [18]. Existing guidelines and standards not only fail to provide specific solutions, but also make compliance a significant challenge. According to a 2007 Ernst & Young survey of chief executives in over 1,100 international organizations, compliance with regulations and policy surpassed worms and viruses as the primary driver of information security policy [12]. The consequence of not complying with regulations is now forefront for those responsible for assuring that software systems containing sensitive information remain secure and protected.

[1] Public Law No. 104-191 Stat. 1936 (1996)
[2] Public Law No. 106-102, 113 Stat. 1338 (1999)

The remainder of this paper is organized as follows: Section II reviews related work; Section III introduces the method to generate compliance monitors; Section IV presents the results of applying the method to rights and obligations previously extracted from the HIPAA Privacy Rule [7]; Section V discusses limitations and future work, with the conclusion in Section VI.

II. Related Work

This section discusses approaches that model regulations, requirements and scenarios, that perform consistency and model checking, and approaches to requirements monitoring. It is important to stress that requirements are limited to the scope of software systems, whereas stakeholder rights and obligations (as expressed in law) govern the broader scope of business processes. Although requirements-based methods generally assume a degree of system control that cannot be assumed in a complete compliance framework, these methods are highly relevant to the specification of stakeholder rights and obligations as requirements.

Prior work focuses on modeling regulations in artificial intelligence [19, 24, 25, 26]. Sherman developed Prolog models from the Income Tax Act of Canada [26]. Sergot et al. have conducted similar case studies, developing logic programs from the British Nationality Act [24] and the Indian Central Service Pension rules [25]. These logic programs abstract rule elements as predicates to query legal expert systems. Our general approach has been to further decompose these predicates into semantic models [4, 5, 7]. I leverage this decomposition to align rights and obligations along shared events (see Section III.C.2). Alternatively, Kerrigan and Law propose a system that provides question-answering assistance with environmental regulations modeled in first-order logic [19].

In requirements engineering, relevant approaches include those to model policies [5], regulations [7] and stakeholder goals [20]. The notation developed from our previous work modeling privacy policies in healthcare and finance [5] is used to specify semantic models in the method proposed in this paper. In addition, Giorgini et al. present Secure Tropos (ST), a formal framework for modeling security requirements applied to Italian privacy regulations [15]. ST distinguishes between rights or permissions (at-most) and obligations (at-least) in the context of delegation. ST employs Datalog to perform model checking and find inconsistencies. The work reported in this paper complements their framework by providing new insight into how rights and obligations are conditioned on shared events. Landtsheer et al. show how to map KAOS goal models into Software Cost Reduction (SCR) specifications amenable to event-based model checking [20]. Goals are prescriptive actions of intent whose satisfaction may require agent cooperation. Like goals, our work with rights and obligations also express stakeholder intent yet within the expressed confines of regulations and normative theory.

Sutcliffe et al. and Maiden describe a partially automated method to generate scenarios using use cases and object system models [21, 28]. Maiden defines object system models as patterns of requirements that include attributes for agents, actions, objects, and pre-conditions, among others. My approach differs in that the compliance monitors derive event sequences from regulations whereas use cases are generally elicited from stakeholders. Furthermore, the FSMs are intended to be used in verifying requirements in post-deployed systems, whereas Maiden's scenarios were intended to be used in requirements validation.

It is important to distinguish aspects of my approach from those in consistency and model checking [1, 3, 10, 16]. Heitmeyer et al. propose consistency checking as a formal method to statically identify ambiguities in requirements specifications [16]. Similarly, the method in this paper utilizes static checks on semantic models to identify ambiguities, called under-specifications, and further balance rights with obligations; this process is called *consistency checking*. Alternatively, model checking is used to assert that formal properties hold across model states and has been applied to requirements specifications to: check safety properties using temporal logic [1]; reduce the number of model states using abstraction [3]; and derive FSM from design flow graphs (DFG) to check consistency between requirements and design [10]. A future goal of this work includes applying model checking techniques to the acquired FSMs.

In requirements monitoring, various approaches use different techniques to specify and deploy requirements monitors. Among these, others have recognized the need to: provide a generalized interface to query assumptions at runtime [11]; model relationships between events to distinguish between expected and recorded system behavior [27]; and align requirements monitoring with design methodologies [23]. The approach in this paper accommodates these needs by generating FSMs for event-based compliance monitoring to evaluate design decisions based on risk and compliance costs. On the other hand, Peters and Parnas discuss design issues for requirements monitors in real-time systems such as real-time notification under discrete-time sampling, sample quantization to measure error, and non-determinism [22]. However, the U.S. federal regulations we analyzed in healthcare did not exhibit these phenomena.

Fickas et al. describe a case study in which ephemeral requirements are modeled as finite state automata in the Promela language and monitored using a web service called Emu [13]. Ephemeral or personal requirements are difficult to monitor because the system environment is beyond the scope of reasonable control [13]. For example, advancing from one state to another may require an indefinite human response. The same limitation exists in compliance monitoring since not all regulations are implementable within the scope of software systems. However, due to diversity in regulated industries, it is infeasible for the law to prioritize implementing regulations based on factors such as available technology, business value or costs. Therefore, all regulations should appear in the scope of compliance monitors, so as organizations evolve, they can evaluate their individual non-compliance risks against these factors.

III. Acquiring Compliance Monitors

The method to acquire compliance monitors (see Figure 1) accepts a semantic model of regulations [7] as input and produces finite state machines (FSMs) as output in two phases: (1) a consistency checking phase to identify under-specifications, and balance rights with obligations; and (2) a generation phase to produce the states-event and transition tables comprising the FSMs.

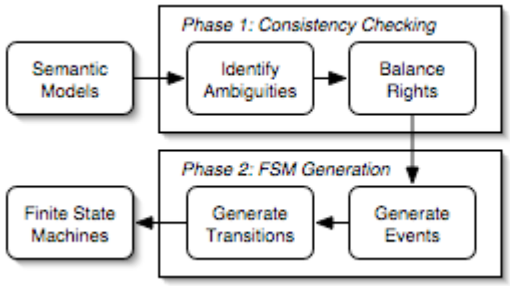

Figure 1: Method Process Overview

The inputs and outputs to the method are described in detail in Section III.A before proceeding to discuss the first and second phases in Sections III.B and III.C, respectively.

A. The Input and Output

The method's inputs and outputs are now discussed. All examples employed below are derived from the HIPAA Privacy Rule [18].

1) Input. Requirements and software engineers provide semantic models of regulations as inputs to the method. Breaux et al. developed a methodology that engineers can use to extract these models from policy and regulation text [7]. The methodology, which was developed from a pilot study [6] and a summative case study [7], employs a process called Semantic Parameterization to derive the models from restricted natural language statements [8].

The semantic models describe rights, obligations and constraints: a *right* is an action a stakeholder is permitted to perform; an *obligation* is an action a stakeholder is required to perform; and a *constraint* describes either an act or state-of-being that is a pre-condition to a right or obligation. In this paper, legal requirements are presented using R_i for a right and O_i for an obligation where i is a unique index. Each legal requirement has a corresponding first-order logic expression consisting of constraints $C_1 \ldots C_n$ and logical connectives (and, or, not).

To illustrate, consider the constraints and obligations below, extracted from the HIPAA Privacy Rule, for the obligation pairs ($O_{4.10}$, $C_1 \wedge C_2$) and ($O_{4.11}$, $C_1 \wedge \neg C_2$) in which a covered entity (CE) provides protected health information (PHI) to the individual:

C_1:	The individual requests to access PHI in a format.
C_2:	The requested format is readily available.
$O_{4.10}$:	The CE must provide the individual with access to PHI in the requested format.
$O_{4.11}$:	The CE must provide the individual with access to PHI in a readable hard copy format.

The obligation $O_{4.10}$ requires the CE to provide access to PHI in the requested format if the individual requests the format (C_1) and the format is available (C_2). Otherwise, the obligation $O_{4.11}$ requires the CE to provide access to PHI in a readable hard-copy format.

Semantic models are expressed in the KTL notation [5], which has been formalized in Description Logic [8] using two relations: the class relation δ(x, y), where the equivalent expressions $x[y]$ and $y{=}x$ read "x is y"; and the property relation $\alpha(x, y)$, where the symmetric expressions $x\{y\}$ and $y : x$ reads "x has y" and "y of x", respectively. Symbols in semantic models are restricted to one part-of-speech from nouns, adjectives, verbs and adverbs; articles and prepositions are not allowed. Symbols preceded by an exclamation point are negated, while symbols preceded by a question mark are query variables. Using unification [2], an expression can be used to query a model [5] — queries are used to identify under-specifications consisting of undefined properties that are required for a particular class in a semantic model [7].

The model for obligation $O_{4.10}$ appears in Figure 2, expressed in the KTL notation. In Figure 2, symbols taken from the obligation statement appear in **bold**; all other symbols comprise part of the reusable *meta-model* that is based upon one of several pre-defined classes with required properties.

```
1   activity [ obligation ] {
2       subject = CE
3       action = provide
4       object = access {
5           subject = individual
6           action = access
7           object = PHI {
8               format [ requested ]
9           }
10      }
11      target = individual
12  }
```

Figure 2: Example Semantic Model

Several patterns have been identified to formalize legal requirements statements using a consistent meta-model [8]. The method discussed herein uses the activity pattern that prescribes a meta-model consisting of an activity class, which requires the properties *subject*, *action* and *object* for the subject who performs an action on the object in an activity [4]. An instance of the activity class appears in Figure 2 for the activity (Line 1), the subject (Lines 2 and 5), action (Lines 3 and 6) and object (Lines 4 and 7) properties. For a legal requirement matching the activity pattern with values assigned from the set of subjects S, actions A, and objects O in the activity, we define the function $T : L \rightarrow S \times A \times O$ that maps the domain of legal requirements L to the range of *SAO*-triples. For example, $T(O_{4.10}) = \langle CE, provide, access \rangle$. The function T is implemented using a static query through existing tool support [5].

2) Output. The method produces finite state machines as output in which each state corresponds to a right or obligation for which stakeholders are accountable. To "reach a state" means to assign a right or obligation to a stakeholder; otherwise, the rights or obligations are considered to be unassigned. From each state, a stakeholder who is assigned a right may invoke that right and a stakeholder who is assigned an obligation must achieve or maintain that obligation. It is considered a *violation* of a right or obligation if the stakeholder cannot invoke an assigned right or cannot achieve or maintain an assigned obligation in a relevant state.

Each state is connected by one or more transitions and each transition coincides with an event, which is an act or state-of-being. The in-transitions to states consist of pre-conditions to rights and obligations, whereas the out-transitions from states consist of the act of invoking a right or achieving and maintaining an obligation. For the obligation pair ($O_{4.10}$, $C_1 \wedge C_2$) from the earlier example in Section III.A.1, we derive the following events:

E_1: The individual requests to access PHI in a format.
E_2: The requested format is readily available.
E_3: The CE provides the individual with access to the PHI in the requested format.

The constraint C_1 maps to the event E_1, the constraint C_2 maps to the event E_2 and the act of achieving the obligation $O_{4.10}$ maps to the event E_3. Since E_1 and E_2 were derived from the constraints, they become in-transitions to the state that corresponds to $O_{4.10}$. The achievement E_3 becomes an out-transition to that state.

The state and transitions are illustrated in Figure 3.

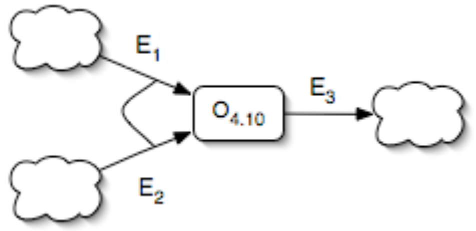

Figure 3: Example State with Transitions

The logical conjunction in the expression $C_1 \wedge C_2$ maps to $E_1 \wedge E_2$ and appears as a bridge between the corresponding in-transitions in Figure 3.

We now describe the individual steps in the first and second phases of the method in Sections III.B and III.C, respectively.

B. *Phase 1: Checking Model Consistency*

In the first phase of the method, semantic models are checked for under-specifications and transformed to balance rights with obligations. The procedure to perform these steps and their contribution to the second phase of the method is discussed in each sub-section that follows.

1) Identify Ambiguities. The method requires the user to resolve ambiguities called under-specifications. The method employs two patterns for doing so: the activity pattern and the verbs masquerading as nouns pattern [8], which both yield an activity that has three co-requisite properties: subject, action and object. These properties must be specified in each activity because they are required to generate events in the second phase. We automatically detect these under-specifications by applying a query algorithm based on unification [2] that proceeds as follows: for each symbol x, if x is a type of *activity* then $\alpha(x, y)$ is true for some symbol $y \in \{subject, action, object\}$; a contradiction indicates an under-specification. The algorithm compares each symbol in a requirement expression for an under-specification and the user must resolve such ambiguities using domain-expert knowledge before proceeding to the next step.

The verbs masquerading as nouns pattern is applied to nouns, including gerunds such as *request*, *denial*, *review*, *agreement*, etc., that are types of activities. For these nouns, an action is implied by the noun (e.g., the action *agree* is implied by the noun *agreement*) and the user maintains a list of these nouns for use in the method. Based on our prior work analyzing policies in healthcare and finance [4, 5], it is reasonable to expect many of these nouns are generalizable across domains. For example, in Figure 2 (above) if the object access (Line 4) were specified without the object PHI (Line 7), the method would detect this ambiguity during this step and require the user to complete the specification.

2) Balance Rights with Obligations. Rights and obligations are balanced by identifying their implied rights and obligations [7]. Implied rights or obligations are not always stated in the regulation text but they are always the logical consequences of expressed rights and obligations and they are needed to improve coverage and identify missing requirements in the model. For this reason, they are logically inferred from expressed rights and obligations and the method provides patterns to do so.

Balancing is guided by four general cases in which rights or obligations are implied by (1) delegations, (2) direct provisions, (3) indirect provisions, and (4) an act where a stakeholder is expressly not obliged, called an *anti-obligation* [7]. Each case uses a *transformation* comprised of a unique query to match the input requirement and identify relevant values that are in turn mapped to an output requirement describing the implied right or obligation. For example, consider the permitted delegation $R_{6.3}$ balanced by implied obligation $O_{R\text{-}6.3}$:

$R_{6.3}$: The CE may require an individual to request in writing that the CE amend their PHI.
$O_{R\text{-}6.3}$: The individual must request that the CE amend their PHI in writing.

In Figure 4, the right $R_{6.3}$ (Lines 1–15) and the implied obligation $O_{R\text{-}6.3}$ (nested in Lines 4–14) is extracted as a separate obligation (Lines 16–25). In general, the transformation uses a unique query to recognize the actions *permit* and *require* as delegation verbs in which the object of the delegation is always the implied right or obligation, respectively. Consequently, for the action *require* (Line 3) the activity (Lines 4–14) is identified as an implied obligation (Line 16–25). Direct and indirect provisions are also balanced using transformations that rely on a unique query to identify and resolve these cases [7].

In the fourth case, *anti-obligations* describe actions that stakeholders are not required to perform. In this case, the stakeholder's implied right is to choose whether or not to perform that action. Anti-obligations are expressed using a negated *obligation* symbol and balanced by replacing the negated *obligation* symbol with a *right* symbol. These symbols appear in square brackets after the *activity* symbol at the head of each expression for anti-obligations.

```
1   activity [ right ] {
2       subject = CE
3       action = require
4       object = activity {
5           subject = individual
6           action = request
7           object = activity {
8               subject = CE
9               action = amend
10              object = PHI : individual
11          }
12          instrument = writing
13          target = CE
14      }
15  }
16  activity [ obligation ] {
17      subject = individual
18      action = request
19      object = activity {
20          subject = CE
21          action = amend
22          object = PHI : individual
23      }
24      instrument = writing
25  }
```

Figure 4: Right Balanced with an Obligation

Balancing rights and obligations requires special handling to map constraints to implied rights or obligations. For delegations and indirect provisions, the balanced right or obligation produces an implied requirement that is pre-conditioned on the invocation of the original delegation or indirect provision. In other words, a stakeholder must first be delegated a right before they can invoke that right. For direct provisions and anti-obligations, the implied requirement inherits the constraints of the direct provision or anti-obligation, because these cases represent the same requirement but from a different stakeholder perspective [7].

In the second phase, we see how events generated from implied rights and obligations correspond to the pre-conditions of other rights and obligations. Balancing rights and obligations ensures these dependent events in pre-conditions are accounted for.

C. *Phase 2: State Machine Generation*

In the method's second phase, two tables are generated: (1) the state-event table is generated by querying the semantic models from the first phase; and (2) the transition table is generated by iterating constraints and entries in the state-event table. Both steps are discussed in detail below.

1) Generate States and Events. In the first step, we populate the state-event table by querying the semantic model. Entries in the state-event table have four fields, including a unique index for the state or event and a *SAO*-triple with subject, action and object.

To populate the table from rights, obligations and their constraints, recall from Section III we introduced the function *T(m)* to identify the *SAO*-triple from a semantic model *m*. In this step, we extend *T* as a recurrence relation *T(o)* for the object $o \in T(m)$, whenever the object *o* is a type of activity. We ensured *T(o)* is well-defined by disambiguating activities in Section III.B.1. In addition, we introduce a similar function *T′(m)* to extract one of two possible *SAO*-triples conveying the regulation's authority over the stakeholder: $T'(m) = \langle Rule, permit, T(m) \rangle$ for a model *m* of a right; and $T'(m) = \langle Rule, require, T(m) \rangle$ for a model *m* of an obligation. In both cases, the subject of the triple is the regulation, identified by *Rule,* whose authority is described by the action, either *permit* or *require*. Consider the example obligation model $O_{6.3}$ in Figure 5, below.

```
1   activity [ obligation ] {
2       subject = CE
3       action = provide
4       object = denial [ written ] {
5           subject = CE
6           action = deny
7           object = request {
8               subject = Individual
9               action = request
10              object = amendment {
11                  subject = CE
12                  action = amend
13                  object = PHI
14              }
15          }
16      }
17      target = Individual
18  }
```

Figure 5: Example Recurrence for *SAO*-triple

The model is an obligation (Line 1) that requires "the CE provide the individual with a written denial to their request for amendment to PHI." Consequently, the function $T'(m) = \langle Rule, require, T(m) \rangle$ and $T(m) = \langle CE, provide, T(denial) \rangle$. Note how the object is an activity (*denial* on Line 4) thus leading to the subsequent recurrence *T(denial)*. For now we ignore properties other than those involved in the *SAO*-triple such as the *target* in Line 13. Applying functions *T′(m)* and *T(m)* yields the entries in Table I from the example obligation $O_{6.3}$.

In the state-event tables, states are entries where the subject is the *Rule* and all other entries are events. Note a state is either a right or an obligation depending on the value in the action field, either *permit* or *require*, respectively. Successive uses of the same subject, action and object fields will reuse the first index to that triple. Table entries for constraints are produced using only the function *T(m)* and the recurrence when applicable. For example, the right $O_{6.3}$ has the constraint "the CE denies an individual's request to amend PHI" in which the function *T* applied to the model yields an event equivalent to event e_2.

TABLE I: EXAMPLE STATE-EVENT TABLE

Index	*Subject*	*Action*	*Object*
$O_{6.3}$	Rule	require	E_1
E_1	CE	provide	E_2
E_2	CE	deny	E_3
E_3	Individual	request	E_4
E_4	CE	amend	PHI

2) Generate Transitions. In the second step, we populate the transition table by generating transitions using events from the first step in phase two. The transition table has four fields: the set number shared by constraints in a conjunction; the source state from which the transition leads out; the event used to generate the transition; and the target state to which the transition leads in.

Each right and obligation state has the following transitions: in-transitions generated for events that were derived from pre-conditions; out-transitions generated from the event in the object field for states in the state table; and, if the state is an obligation, transitions for the negation of the event in the object field of the state table. The negation of the event for obligations always leads to a non-compliant state (NC) equivalent to violating the obligation. This is different from negating the obligation, which is called an anti-obligation. For rights, the target state of this transition is unspecified. Continuing with the example from Section III.C.1, we generate the transitions for obligation $O_{6.3}$ in Table II.

TABLE II: EXAMPLE TRANSITION TABLE

Set	*Source*	*Event*	*Target*
1		E_2	$O_{6.3}$
2	$O_{6.3}$	E_1	
3	$O_{6.3}$	$\neg E_1$	$NC_{6.3}$

The in-transition (first row) to $O_{6.3}$ corresponds to the constraint on $O_{6.3}$ and the out-transition (second row) corresponds to the object from the state $T(O_{6.3}) = \langle Rule, require, e_1 \rangle$ in Table I. The transition to the non-compliant state (third row) must eventually be conjoined with a time-out event or deadline to complete this monitor. The graphic illustration of this monitor appears in Figure 6, where the clouds represent placeholder states that have yet to be specified.

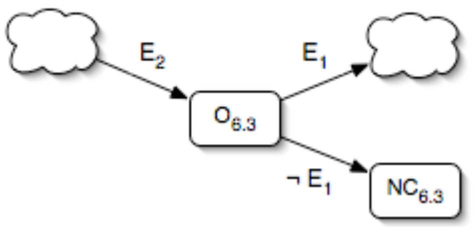

Figure 6: Example Compliance Monitor

After the state-event and transition tables have been generated, one can derive a combined compliance monitor by pairing events from in- and out-transitions to connect states. The combined compliance monitor more effectively illustrates the interactions between rights and obligations. We present such a graphic in Section IV as an application of the entire method.

IV. Results from HIPAA Privacy Rule

In a previous case study [7], we derived semantic models from rights, obligations and constraints that were extracted from the Privacy Rule [18] – a U.S. federal regulation for the HIPAA. The Rule governs use and disclosure of patient healthcare information. Based on discussions with chief security and privacy officers, companies prioritize compliance with those regulations most likely to interface with the public and consumers. For this reason, the method was applied to §164.520 – §164.526 in Subpart E of the Privacy Rule. Results from §164.524, titled "Access of individuals to protected health information," are presented below.

The analysis of §164.524 in the case study yielded a total of 20 rights, 26 obligations and 67 constraints. From these, the following rights and obligations are most relevant to generate the largest combined compliance monitor using the method. The following acronyms are used: covered entity (CE), licensed healthcare professional (LHP), and protected health information (PHI).

$R_{4.1}$: The individual has a right to request access to their PHI.
$R_{4.3}$: The CE may deny an individual access to their PHI. (**C_1**)
$R_{4.5}$: The individual may have a denial of requested access reviewed by an LHP. (**C_2**)
$O_{4.1}$: The CE must permit an individual access to their PHI. (**C_3**)
$O_{4.2}$: The CE must deny an individual access to their PHI. (**C_4**)
$O_{4.3}$: The CE must permit an individual to request access to their PHI.
$O_{4.5}$: The CE must inform the individual that requested access is permitted. (**C_5**)
$O_{4.7}$: The CE must inform the individual that the requested access was denied. (**C_2**)
$O_{4.16}$: The CE must designate an LHP to review a denial of requested access. (**C_6**)
$O_{4.18}$: The LHP must recommend that the CE permit or deny the individual access to PHI. (**$C_7 \wedge C_8$**)
$O_{4.19}$: The CE must inform the individual of the recommendation of the LHP. (**$C_3 \vee C_4$**)

Each right and obligation above is annotated with the logical expression of constraints (in parenthesis) from the following list:

C_1: The individual requests access to their PHI.
C_2: The CE denies requested access to PHI.
C_3: The LHP recommends the CE permit access.
C_4: The LHP recommends the CE deny access.
C_5: The CE permits the requested access to PHI.
C_6: The individual requires an LHP review a denial.
C_7: The CE designates the LHP to review a denial.
C_8: The LHP reviews the denial of access.

For the purpose of this illustration, only those events that form transitions between states are highlighted and constraints that only describe state-of-being as they contribute no such events are ignored. As a result, the following state-event and transition tables are incomplete under the law but sufficient as an exemplar in this paper.

In phase one (consistency checking), step one, the user of the method identified several under-specifications in the original semantic model for rights, obligations and constraints. To resolve these ambiguities, the user is required to specify 37 subjects, 35 actions and 32 objects by coordinating domain expert elicitation with a review of the relevant context in the source legal text.

In step two, the user balanced two rights and one obligation. The rights $R_{4.1}$ and $R_{4.5}$ and the obligation $O_{4.3}$ were balanced with new models $O_{R\text{-}4.1}$, $O_{R\text{-}4.5}$ and $R_{O\text{-}4.3}$, respectively. Since $O_{R\text{-}4.1} \approx O_{4.3}$ and $R_{4.1} \approx R_{O\text{-}4.3}$, the right $R_{4.1}$ balanced directly with $O_{4.3}$, which means $O_{R\text{-}4.1}$ and $R_{O\text{-}4.3}$ were not new contributions to the model. However, the right $R_{4.5}$ only balances with obligation $O_{R\text{-}4.5}$ requiring the LHP to review denials of requested access, so $O_{R\text{-}4.5}$ is a new contribution.

In phase two (FSM generation), step one, the states and events were generated (see Table III). There were 17 instances where events previously entered into the table were duplicated with other states.

The generated transitions from step two appear in Table IV. The in-transitions were generated from constraints (Sets 1–11). The out-transitions were generated from the object value of states (Sets 12–25), the alternate transitions from rights (Sets 26–30) and the transitions from states to non-compliant states (Sets 31–40) all from Table III.

TABLE III: STATE-EVENT TABLE FOR HIPAA §164.524

Index	***Subject***	***Action***	***Object***
$R_{4.1}$	Rule	permit	E_1
E_1	Individual	request	E_2
E_2	Individual	access	PHI
$R_{4.3}$	Rule	permit	E_3
E_3	CE	deny	E_2
$R_{4.5}$	Rule	permit	E_4
E_4	Individual	require	E_5
E_5	LHP	review	E_3
$O_{R\text{-}4.5}$	Rule	require	E_5
$O_{4.1}$	Rule	require	E_6
E_6	CE	permit	E_2
E_7	LHP	recommend	E_6
$O_{4.2}$	Rule	require	E_3
E_8	LHP	recommend	E_3
$O_{4.3}$	Rule	require	E_9
E_9	CE	permit	E_1
$O_{4.5}$	Rule	require	E_{10}
E_{10}	CE	inform	E_6
$O_{4.7}$	Rule	require	E_{11}
E_{11}	CE	inform	E_3
$O_{4.16}$	Rule	require	E_{12}
E_{12}	CE	designate	LHP
$O_{4.18}$	Rule	require	E_7
$O_{4.18}$	Rule	require	E_8
$O_{4.19}$	Rule	require	E_{13}
E_{13}	CE	inform	E_7
$O_{4.19}$	Rule	require	E_{14}
E_{14}	CE	inform	E_8

The combined compliance monitor is illustrated in Figure 7. In Figure 7, the transitions for events produced by invoking rights are illustrated using dotted lines whereas the transitions for events produced by achieving or maintaining obligations are illustrated using solid lines. For easier reading, the figure omits transitions to non-compliant states and alternate transitions for rights that do not align with existing states. Unspecified states appear as clouds with events E_{10}, E_{11}, E_{13} and E_{14} leading to such states.

Figure 7 makes it easier to recognize important aspects of the combined compliance monitor. For example, the *unconditional* rights and obligations such as $O_{4.3}$ have no in-transitions. Rights or obligations that immediately follow unconditional obligations, like right $R_{4.1}$, are consequently unconditional, unless the stakeholder violates the preceding obligation. In addition, loops on states for obligations require stakeholders to maintain that state. For example, based on obligation $O_{4.18}$, if the LHP determines the CE should not provide access (via E_8, bottom center), then the CE must deny access (via E_3 and the loop at $O_{4.2}$).

Rights that provide stakeholders choices are also easier to visualize. For example, in state $R_{4.3}$, the CE has a choice: (1) they can deny the requested access via E_3, in which case they must (a) inform the individual via $O_{4.7}$ and (b) provide the right to review via $R_{4.5}$; or (2) they can permit the requested access, in which case they must inform the individual via $O_{4.5}$.

TABLE IV: TRANSITION TABLE FOR HIPAA §164.524

Set	***Source***	***Event***	***Target***
1		E_1	$R_{4.3}$
2		E_3	$R_{4.5}$
3		E_4	$O_{R\text{-}4.5}$
4		E_3	$O_{4.7}$
5		E_3	$O_{4.16}$
6		E_7	$O_{4.1}$
7		E_7	$O_{4.19}$
8		E_8	$O_{4.2}$
9		E_8	$O_{4.19}$
10		E_6	$O_{4.5}$
11		E_{12}	$O_{4.18}$
11		E_5	$O_{4.18}$
12	$R_{4.1}$	E_1	
13	$R_{4.3}$	E_3	
14	$R_{4.5}$	E_4	
15	$O_{R\text{-}4.5}$	E_5	
16	$O_{4.1}$	E_6	
17	$O_{4.2}$	E_3	
18	$O_{4.3}$	E_9	
19	$O_{4.5}$	E_{10}	
20	$O_{4.7}$	E_{11}	
21	$O_{4.16}$	E_{12}	
22	$O_{4.18}$	E_7	
23	$O_{4.18}$	E_8	
24	$O_{4.19}$	E_{13}	
25	$O_{4.19}$	E_{14}	
26	$R_{4.1}$	$\neg E_1$	
27	$R_{4.3}$	$\neg E_6$	
28	$R_{4.3}$	$\neg E_3$	
29	$R_{4.5}$	$\neg E_4$	
30	$O_{R\text{-}4.5}$	$\neg E_5$	
31	$O_{4.1}$	$\neg E_3$	$NC_{4.1}$
32	$O_{4.2}$	$\neg E_6$	$NC_{4.2}$
33	$O_{4.3}$	$\neg E_9$	$NC_{4.3}$
34	$O_{4.5}$	$\neg E_{10}$	$NC_{4.5}$
35	$O_{4.7}$	$\neg E_{11}$	$NC_{4.7}$
36	$O_{4.16}$	$\neg E_{12}$	$NC_{4.16}$
37	$O_{4.18}$	$\neg E_7$	$NC_{4.18}$
38	$O_{4.18}$	$\neg E_8$	$NC_{4.18}$
39	$O_{4.19}$	$\neg E_{13}$	$NC_{4.19}$
40	$O_{4.19}$	$\neg E_{14}$	$NC_{4.19}$

Rights and obligations assigned through delegation are clearly shown. For example, the obligation requiring the CE to permit the individual to request access (the path $O_{4.3} \rightarrow E_9 \rightarrow R_{4.1} \rightarrow E_1$) is distinct from the right permitting the individual to require an LHP to review a denial (the path $R_{4.5} \rightarrow E_4 \rightarrow O_{R\text{-}4.5} \rightarrow E_5$). However, the graphic does not clearly distinguish between rights to obligate other stakeholders and obligations that follow from invoking a stakeholder right, such as the path $R_{4.3} \rightarrow E_6 \rightarrow O_{4.5} \rightarrow E_{10}$.

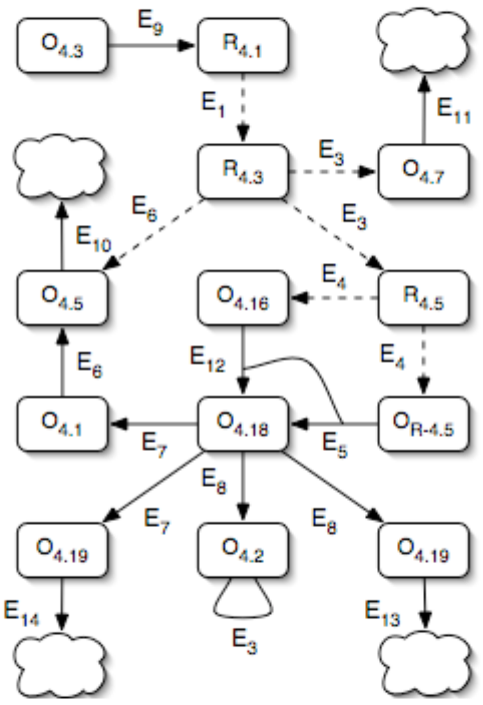

Figure 7: Combined Compliance Monitor

V. Discussion and Future Work

Before concluding with future work, we quickly discuss the current limitations.

In phase two, step one in Section III.C.1, the method must determine if two events are equivalent in order to reuse shared events. In this study, we only used the *SAO*-triple to compare events and identify duplicates; however, two different events can have the same *SAO*-triple. For example, two similar requests to two different recipients or a repeated event with dissimilar temporal constraints could both have the same *subject*, *action*, and *object*. We propose addressing this problem by extending the method to compare the semantic model sub-components used to generate the events instead of the *SAO*-triple, since they will have the necessary information to distinguish these events – including any temporal constraints.

In phase two, step two in Section III.C.2, the set number is sufficient to assign transitions to logical conjunctions or disjunctions. However, we encountered the need to support exclusive-or on the out-transitions for obligations (see $O_{4.18}$ in Section IV) as a convenience to stakeholders. The consequences of exclusive-or on out-transitions impacts how transitions to non-compliant states are generated. For example, for the two events, A and B, the out-transitions for an obligation in the expression $(A \wedge \neg B) \vee (\neg A \wedge B)$ only require a transition to a non-compliant state for $(\neg A \wedge \neg B)$ and not for $\neg A$ independent of $\neg B$.

Future work includes integrating the compliance monitors produced by the method into runtime systems and additional validation to assess the repeatability of the method. Formalization of rights and obligations using FSM is a significant first step towards runtime compliance monitoring, however, additional work is needed to identify the software interfaces and functions responsible for implementing these legal requirements. Regarding validation, the method was rigorously applied to four sections of the HIPAA Privacy Rule and shown to yield compliance monitors for one method user. Because humans have natural differences in ability, however, we do not know the extent to which requirements engineers can apply the method, in general. This is a problem with formal methods in general and requires additional experimental study in a controlled laboratory. That said, understanding the difficulty other users have with the method may suggest directions for future automation and support to improve consistency and coverage in the FSM.

Finally, future work includes the need to evaluate risk and compliance associated with the decision to implement a right or obligation in systems. *Risk* is the probability that a regulation will be violated by a business or system process. For each right or obligation, calculating risk requires knowing the frequencies of real-world events that pre-condition, satisfy or violate the right or obligation. Furthermore, risk must also factor in the penalty or cost of violating a right or obligation. For rights or obligations with a high penalty, frequent violation or with frequent satisfaction of pre-conditions, there is a higher priority to implement system processes to prevent violation and monitor compliance at runtime.

For example, in Figure 7 in Section IV, the individual is given the right to request access to PHI via right $R_{4.1}$ and, if denied that access, they receive the right to have an LHP review the denial via right $R_{4.5}$. The review involves a complex set of stakeholder interactions between the obligations of the LHP and CE and the rights of the individual. If individuals rarely exercise right $R_{4.5}$, the cost of implementing these obligations in systems may be incommensurate with the frequency of violations due to human error in the business process. Since the compliance monitors can be aligned with both business and system processes, stakeholders can use these monitors to design and develop software systems commensurate with risk and compliance costs.

VI. Acknowledgements

This work was funded by NSF ITR Grant #032-5269.

VII. References

[1] J. Atlee, J. Gannon, "State-based Model Checking of Event-driven System Requirements." *Conf. Soft. for Critical Systems*, New Orleans, LA, pp. 16-28, 1991.

[2] F. Baader, J.H. Siekmann, Unification Theory. *Handbook of Logic in AI and Logic Programming*, Oxford University Press, New York, NY, pp. 41-125, 1994.

[3] R. Bharadwaj, C.L. Heitmeyer, "Model Checking Complete Requirements Specifications Using Abstraction." Auto. Soft. Engr., 6(1), pp. 37-68, 1999.

[4] T.D. Breaux, A.I. Antón, "Deriving Semantic Models from Privacy Policies." *IEEE Workshop on Policies for Distributed Sys. & Networks*, Sweden, pp. 67-76, 2005.

[5] T.D. Breaux, A.I. Antón, "Analyzing Goal Semantics for Rights, Permissions, and Obligations." *IEEE Req'ts. Engr. Conf.*, Paris, France, pp. 177-186, 2005.

[6] T.D. Breaux, A.I. Antón, "Mining Rule Semantics to Understand Legislative Compliance." *ACM Workshop on Privacy in Electronic Society*, USA, pp. 51-54, 2005.

[7] T.D. Breaux, M.W. Vail, A.I. Antón, "Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations," *IEEE Int'l Conf. Reqts. Engr,* pp. 49-58, 2006.

[8] T.D. Breaux, A.I. Anton, J. Doyle, "Semantic Parameterization: A Process for Modeling Domain Descriptions," *ACM Trans. on Software Engineering Methodology,* 18(2): 5, Nov. 2008.

[9] T.D. Breaux, A.I. Anton, C.-M. Karat, J. Karat, "Enforceability vs. Accountability in Electronic Policies," *IEEE 7th International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, London, Ontario, pp. 227-230, Jun. 2006

[10] M. Chechik, J. Gannon, "Automatic Analysis of Consistency Between Requirements and Design." *IEEE Trans. Soft. Eng.*, 27(7), pp. 651-672, 2001.

[11] D. Cohen, M.S. Feather, K. Narayanaswamy, S.F. Fickas, "Automatic Monitoring of Software Requirements." *IEEE Int'l Conf. Soft. Eng*, pp. 602-603, 1997.

[12] Ernst & Young, *10th Annual Global Information Security Survey: Achieving A Balance of Risk and Performance*, 2007.

[13] S.F. Fickas, T. Beauchamp, N.A.R. Mamy, "Monitoring Requirements: A Case Study." *IEEE Int'l Conf. Auto. Soft. Eng.* Edinburgh, UK, pp. 299-304, 2002.

[14] B.A. Garner, editor. *Blacks Law Dictionary, 8th ed.* Thompson West, St. Paul, Minnesota, 2004.

[15] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone. "Modeling Security Requirements Through Ownership, Permission and Delegation." *IEEE 13th Req'ts. Eng.. Conf.*, France, pp. 167-176, 2005.

[16] C.L. Heitmeyer, R.D. Jeffords, B.G. Labaw. "Automated Consistency Checking of Requirements Specifications," *ACM Trans. Soft. Eng. Methods,* 5(3), pp. 231-261, 1996.

[17] Health Insurance Portability and Accountability Act, USC H.R. 3103-168, April 2000.

[18] Office for Civil Rights, "Standards for Privacy of Individually Identifiable Health Information." 45 CFR Part 160, Part 164 Subpart E. In Federal Register, vol. 68, no. 34, February 20, 2003, pp. 8334 – 8381

[19] S. Kerrigan, K.H. Law, "Logic-based Regulation Compliance-Assistance." *Int'l Conf. AI and Law,* pp. 126-135, 2003.

[20] R. de Landtsheer, E. Letier, A. van Lamsweerde, "Deriving Tabular Event-based Specifications from Goal-Oriented Requirements Models." *IEEE Req'ts. Eng. Conf.*, Monterrey, CA, pp. 200-210, 2003.

[21] N.A.M. Maiden, "CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements," *Auto. Soft. Eng.* 5(4), pp. 419-446, 1998.

[22] D.K. Peters, D.L. Parnas, "Requirements-based Monitors for Real-time Systems." *IEEE Trans. Soft. Eng.,* 28(2), pp. 146-158, 2002.

[23] W.N. Robinson, "A Requirements Monitoring Framework for Enterprise Systems." *Req'ts. Eng. Journal*, 11(1), pp. 17-41, 2005.

[24] M.J. Sergot, F. Sadri, R.A. Kowalski, F. Kriwaczek, P. Hammond, H.T. Cory, "The British Nationalisty Act as a Logic Program," *Comm. of the ACM*, 29(5), pp. 370-386, 1986.

[25] M.J. Sergot, A.S. Kamble, K.K. Bajaj, "Indian Central Civil Service Pension Rules: A Case Study in Logic Programming." *Int'l Conf. AI & Law,* pp. 118-127, 1991.

[26] D. Sherman, "A Prolog Model of the Income Tax Act of Canada." *Int'l Conf. AI & Law,* pp. 127-136, 1987.

[27] G. Spanoudakis, K. Mahbub, "Requirements Monitoring for Service-based Systems: Towards a Framework Based on Event Calculus." *IEEE Int'l Conf. Auto. Soft. Eng.,* Linz, Austria, pp. 379-384, 2004.

[28] A.G. Sutcliffe, N.A.M. Maiden, S. Minocha, D. Manuel, "Supporting Scenario-based Requirements Engineering," *IEEE Trans. Soft. Eng.*, 24(12), pp. 1072-1088, 1998.