Validating Existing Requirements for Compliance with Law Using a Production Rule Model

Jeremy C. Maxwell and Annie I. Antón
North Carolina State University
{jcmaxwe3, aianton}@ncsu.edu

Abstract

To ensure legal compliance, requirements engineers need tools to validate existing requirements for compliance with relevant law. This paper proposes an approach to aid in validating software requirements for legal compliance. The approach employs a production rule model for the United States Health Insurance Portability and Accountability Act (HIPAA) that can be queried by requirements engineers as they validate and refine software requirements. In this paper, we discuss a case study in which we applied the approach to evaluate the iTrust Medical Records System requirements for HIPAA compliance. We were able to identify 12 new potential requirements beyond the original 63 functional requirements, as well as operationalize one nonfunctional requirement.

1. Introduction

Software engineers are increasingly asked to develop software for regulated environments. The U.S. Sarbanes-Oxley Act regulates financial companies. The U.S. Health Insurance Portability and Accountability Act[1] (HIPAA) regulates the healthcare and health insurance industries. The European Union's Directive 95/46/EC requires that personal data remains protected. When developing software for domains regulated by these and other laws, requirements engineers must verify that existing requirements comply with laws and regulations, as well as extracting new requirements from law.

The cost of noncompliance is high, including fines, cost of court representation, government audits, and workforce training. The Choicepoint data breach cost the company in excess of 27 million dollars [OAB07]. Using studies in The New England Journal of Medicine and the American Hospital Association's 2003 Hospital Statistics, we estimate that hospitals budgeted between 360 million and 1.2 billion dollars for HIPAA compliance in 2003 alone [AHA03, Kil03]. This estimate is only for hospitals, and does not consider doctor's offices, health insurance companies, and other entities affected by HIPAA. Because of the cost of noncompliance, complying with legal regulations must be a focus area for software firms when developing software.

The fundamental problem we address is the communication gap between requirements engineers and legal domain experts. Requirements engineers, software designers, and developers are well versed in the technical aspects of software development; lawyers, policy makers, as well as judicial and legislative officials are well versed in the law. The communication gap between these domains can open up software to noncompliance.

We present an approach to analyze existing requirements' completeness concerning regulatory compliance. We query a production rule model to determine the compliance of existing requirements, elicit potential requirements to improve regulatory compliance, and assist in operationalizing nonfunctional requirements. Production rule models provide a unique advantage when performing compliance analysis: a low amount of knowledge of the regulation is required to perform the analysis. We developed our approach via an exploratory case study, analyzing the requirements of iTrust, an open-source electronic health records system, for HIPAA compliance.

Production rules, a knowledge representation technique used in artificial intelligence [DHL86], are usually stated in Horn clauses connected by logical operators [BL04]. In other words, each rule is an if-then statement. Many such rules combine to create a knowledge base, also called a rules base. To interact with this rules base, a query is presented and viewed as a top-level goal. An inference engine then uses a reasoning strategy, usually backwards chaining, to execute the rules in the rules base. The result is an affirmation or a refutation of the original query [SS94].

[1] 45 CFR Parts 160, 162, and 164

The remainder of this paper is organized as follows: Section 2 reviews work related to production rule modeling of legal texts, rights and obligations in legal texts, and evaluating existing requirements for legal compliance; Section 3 reviews the approach used in the iTrust case study; Section 4 discusses the results of our case study; Section 5 discusses threats to the validity of the case study; and Section 6 provides a summary of this work and outlines areas of future work in the field.

2. Related Work

Prior work has focused on modeling legal texts using production rules, extracting rights and obligations from legal texts, and using traditional requirements engineering techniques to evaluate requirements for legal compliance.

2.1. Production Rule Modeling of Legal Texts

Knowledge representation has been identified as a key activity for requirements engineering [BGM85]. Creating models of the software's target environment encourages communication between stakeholders. Additionally, models allow: for querying the model; the derivation of new knowledge not yet represented in the model; and for simulation [BGM85]. Production rules, among other techniques, have been proposed for use in requirements engineering [DHL86].

We have made use of production rules to model a portion of the HIPAA Privacy Rule [MA09]. We the Production Rule Modeling methodology, a two-activity process for translating a legal text into Prolog. We translated §164.520, §164.522, §164.524 and §164.526 of the HIPAA Privacy Rule into Prolog. We query this model when performing our analysis of the iTrust requirements.

Production rules have been used to model several other legal texts [BMT87, BRR87, SSK86, She87, SKB91]. These works focus on: improving the understanding of law using production rules [BMT87]; knowledge representation research rather than practical uses of production rule models [BRR87, SSK86]; aiding law makers in drafting legislation [SSK86]; and legal reasoning [BMT87, She87, SKB91]. Production rule models have not been used to assist in requirements validation and compliance checking.

Production rule modeling has also been used by Mitchell et al. to create a preliminary version of a HIPAA Compliance Checker[2] [HS08, Mit08]. The tool, written in Prolog and based on a class project, seeks to enable the discovery of inconsistencies in legal texts, while providing real-time message checking for privacy violations. There are several differences between their work and ours. Their Checker determines whether a message can legally be transmitted between two entities; thus, they have yet to address the broad range of queries present in the regulation. For example, a model of the HIPAA Security and Privacy Rule could potentially be queried about access control, the right of notice, security requirements, etc. Second, their work does not make use of the rights, obligations, and permissions, whereas we draw upon our prior work in this area [BA08, BVA06].

2.2. Extracting Rights and Obligations from Legal Texts

The Semantic Parameterization methodology extracts rights and obligations from regulatory texts [BA08, BVA06, BAD09]. This methodology is based on deontic logic, which is concerned with the notions of permission and obligation. Rights, obligations, and permissions are the main query mechanisms of the production rule model we have developed previously [MA09]. As such, in our case study, we query the model to determine an organization's legal obligations that must be fulfilled by the covered entity, and, by extension, the software.

2.3. Evaluating for Legal Compliance using Traditional Requirements Engineering Techniques

Massey et al. examined the iTrust requirements for security and legal concerns [MOH08]. They developed a methodology for tracing requirements back to applicable regulatory sections. Their input was 27 Unified Modeling Language (UML) use cases and four nonfunctional requirements, which comprised the entirety of the original iTrust requirements. They derived 63 functional and 10 nonfunctional requirements, which we use for our case study presented in Section 4. Additionally, a portion of their methodology, mapping terminology in a requirements document to the terminology in a legal text, is similar to the first activity in our approach, presented in Section 3. Our work differs from the work by Massey et al. through our use of Prolog to validate existing requirements and identify new potential requirements—Massey et al. rely on more traditional techniques such as the Inquiry Cycle Model for evaluating requirements for legal compliance [PTA94].

[2] http://crypto.stanford.edu/privacy/HIPAA/

2.4. Prolog Overview

In this section, we provide a brief overview of the Prolog syntax we use in this paper. The syntax of a Prolog rule is:

```
<result> :-
   <condition1>,
   <condition2>,
   ...
   <conditionN>.
```

Where the symbol `:-` is interpreted as the if conditional, the comma symbol is interpreted as logical-and, and the period symbol is interpreted as a full stop (the end of a rule). The result is evaluated to true only if {`condition1, condition2,..., conditionN`} are evaluated to true. The Prolog rule `father(X,Y) :- male(X), child(Y,X)` is read "X is the father of Y if X is male and Y is the child of X." The portion of the rule before the `:-` symbol is called the head of the rule, while the portion following the `:-` is called the body of the rule [SS94].

An atom is a name, quoted string, or a sequence of special characters (`:-` is one example). A term is the basic unit in Prolog; a term can be an atom, an integer value, a variable or a compound term. A variable signifies a single yet unspecified quantity, and are signified by beginning with a capital letter. A compound term consists of a predicate and its arguments, where a predicate is a relationship between atoms. A procedure is a set of rules that have the same predicate as the head [SS94].

A production rule model makes use of two built-in Prolog commands. The `assert(NewFact)` command adds a new fact to the knowledge base, namely the parameter of the command. Similarly, the `retract(Fact)` command removes the first occurrence of the specified fact from the knowledge base [SS94]. We will make use of these commands to build precondition sets from the requirements to query the model.

The strength of Prolog to answer queries comes from two concepts: unification and backtracking [Set90]. Unification occurs when the inference engine attempts to find a single value to bind to multiple occurrences of a variable. The inference engine backtracks to find alternate solutions when they are requested or when one line of reasoning has failed.

2.5. iTrust Overview

iTrust[3] is an open-source electronic health records system used as an instructional tool at North Carolina State University. The system is developed as part of undergraduate and graduate courses; each semester, students develop new functionality and test the code developed from previous semesters.

For our case study, we use the 73 requirements developed by Massey et al. [MOH08]. We focus on analyzing the 63 functional requirements; the 10 nonfunctional requirements, as is usually the case, tend to detail system wide requirements. For example, the nonfunctional requirements specify technology constraints, testing criteria, coding standards, and the web browsers iTrust must be compatible with. One of the nonfunctional requirements is "iTrust shall comply with HIPAA and other laws and regulations."

3. Using Production Rules to Validate Requirements' Compliance

We developed a three activity approach for checking requirements for regulatory compliance. An overview of this approach is presented in Figure 1. A production rule model of the legal text and an existing set of requirements are inputs. The steps are listed as follows:

1. *Map Terminology*. Map requirements document terminology to regulation terminology used in the production rule model.
2. *Identify Precondition Sets*. Preconditions for queries are identified and grouped into sets.
3. *Analyze Requirements*. Identify conflicts, gaps in requirements coverage, and organizational concerns. The identification of new legal preconditions in this step may require the engineer to return to the *Identify Preconditions* step to create new precondition sets.

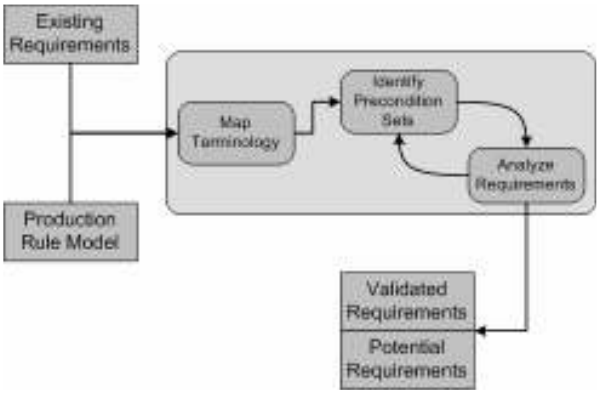

Figure 1. Approach Overview

Applying the steps in our approach, using a production rule model, a set of validated requirements, along with new potential requirements, are produced. In the remainder of this section, we will discuss each step in detail using a concrete example from our case study. Requirement 21 from the iTrust requirements

[3] http://agile.csc.ncsu.edu/iTrust/wiki/doku.php

document is listed in Figure 2. We will identify new compliance requirements using this requirement as a starting point.

Reqt$_{21}$: iTrust shall allow a patient, using his/her authorized account, to read or update his/her demographic information, the demographic information for the patients he/she represents, his/her list of personal representatives, the list of personal representatives for the patients he/she represents, their list of designated physicians, and the list of designated physicians for the patients he/she represents.

Figure 2. An iTrust Requirement

3.1. Map Terminology

Mapping terminology entails mapping the natural language phrases in the requirements document onto the terminology used in the legal text as expressed by the production rule model. As proposed by Massey et al. [MOH08] this is the first phase of checking requirements for regulatory compliance. There are three categories of terminology: actors, objects, and relations. Actors are individuals and organizations that have a right, are constrained by an obligation, or permitted to perform some action [MA09], or appear in the requirements document as a subject. Objects are inanimate things with which actors interact. Relations are actions performed by actors, relationships between actors or objects, or properties of actors or objects in the production rule model.

Ideally, terminology should be mapped early in the requirements engineering process, as regulatory compliance should be a consideration from the outset. Oftentimes, however, engineers are working on existing systems whose requirements have already been specified. These requirements: (a) may predate the regulation, (b) were specified based on an older version of the regulation, or (c) were specified without regard to the regulation at all. Thus, terminology mapping is an essential activity.

A prerequisite to the Map Terminology activity is well-defined, consistent requirements document terminology. This may require discussion and negotiation with stakeholders to address conflicting or vague terminology.

A challenge mentioned but not solved by Massey et al. [MOH08] is the familiarity with the legal text that is required before terminology mapping can take place. A requirements engineer must have a deep knowledge of the legal text in order to accurately map terms. The queriability of production rule models enables us to address this challenge. As a result of the iTrust case study, we have added two Prolog procedures to our model: `glossary` and `whatIs`. Using these two procedures, a requirements engineer can obtain the definitions of terms used in a legal text. This integrated glossary is similar to others used in previous legal knowledge base systems [GMT87, SAA00].

The `glossary` procedure prints a listing of all predicates, derived from the legal text, that are used in the production rule model. The actor terms, object terms, and the relation terms can be printed individually, or at the same time. Currently, there are 24 actors, 19 objects, and 69 relations in the glossary.

The `whatIs` procedure displays the actual definition of the term from the legal text, along with the section the definition appears in, where applicable. For example, to determine the meaning of the term 'protected health information' (PHI), the command `whatIs(phi)` is used, resulting in the output shown in Figure 3. Some terminology, however, does not have a specific definition in the legal text. For example, `receive` is an action preformed from one actor to another, but does not have a precise legal definition in the HIPAA Privacy Rule. For terminology such as `receive`, `whatIs` provides documentation on the uses of the term in the production rule model. The `glossary` and `whatIs` Prolog procedures form the basis of a legal glossary as suggested by Otto and Antón [OA07]. Using these two procedures, an engineer is able to

```
1 ?- whatIs(phi).
Protected Health Information
SOURCE: HIPAA Privacy and Security Rule, section 160.103
means individually identifiable health information:
   (1) Except as provided in paragraph (2) of this definition, that is:
      (i) Transmitted by electronic media;
      (ii) Maintained in electronic media; or
      (iii) Transmitted or maintained in any other form or medium.
   (2) Protected health information excludes individually identifiable health
        information in:
      (i) Education records covered by the Family Educational Rights and Privacy Act,
        as amended, 20 U.S.C. 1232g;
      (ii) Records described at 20 U.S.C. 1232g(a)(4)(B)(iv); and
      (iii) Employment records held by a covered entity in its role as employer.

Yes
```

Figure 3. Example of `whatIs`

determine the terminology used in the production rule model, the best map with the requirements document terminology, and gain confidence that terms are being used consistently. To illustrate, consider Requirement 21, listed in Figure 2. Using the two Prolog procedures, we map the iTrust terminology to the terminology in the production rule model. Table 1 displays the result of mapping the terminology of Requirement 21. We use the notation `CE` to denote a covered entity as defined by the HIPAA Privacy Rule. Examples of covered entities include health care providers, insurance companies, and correction institutions.

Table 1. Mapping the Terminology of $Reqt_{21}$

iTrust Terminology	Production Rule Model Terminology
patient	`individual`
demographic information	`phi`
designated physician	`lhcp`
personal representative	`individualRepresentative`
read demographic information	`receive(individual,CE,phi)`
update demographic information	`requests(individual, CE, amends(CE, phi))`

3.2. Identify Precondition Sets

Identify Precondition Sets entails gathering preconditions derived from the law and individual requirements. A precondition set signifies a set of facts that represent one instance or possibility. These sets of facts will then be used in the Analyze Requirements activity to query the knowledge base. Precondition sets are built from individual requirements, to test the requirement for regulatory compliance, and from law to express legal preconditions. Some requirements may have no preconditions.

Initially, the only precondition sets an engineer is able to identify come from the requirement. For example, Table 2 presents precondition sets identified from Requirement 21 listed in Figure 2. In preparation for the Analyze Requirements activity, each precondition set must be specified as a series of Prolog `assert` statements, with each precondition corresponding to one `assert` statement.

Table 2. Precondition Sets Identified from $Reqt_{21}$

PS1	`assert(coveredUnder(individual, CE)).`
PS2	`assert(coveredUnder(individual, CE)).` `assert(receive(individual,CE,phi)).`
PS3	`assert(coveredUnder(individual, CE)).` `assert(requests(individual, CE, amends(CE, phi))).`
PS4	`assert(coveredUnder(individual, CE)).` `assert(represents( individualRepresentative, individual)).`

3.3. Analyze Requirements

Analyzing Requirements entails using the precondition sets identified in the previous activity to query the production rule model. A requirements engineer then uses the query responses to determine potential areas of noncompliance in the original requirements. The query process is iterative, by which an engineer explores how different precondition sets play out in the model. Invaluable to this is the Prolog `trace` procedure. This procedure instructs the inference engine to print each goal the engine attempts to prove, and the results of each proof. Through this mechanism, and engineer can gain insight to the preconditions for failed rules. These preconditions are derived from the legal text, as they are preconditions to rules in the model. Some preconditions may not have been represented in the initial precondition sets. The engineer then repeats the Analyze Requirements activity for the newly identified precondition set.

An example `trace` transcript is listed in Figure 4. using a precondition set containing a single precondition, expressed in Prolog as: `assert(coveredUnder (individual, hmo)).` This precondition set is a variant of PS1 from Table 2. The query being executing is `may(hmo, Permission, Source).` The first column in Figure 4 is the activity the inference engine is performing: testing a goal (`Call`), exiting a successful goal (`Exit`), failing to prove a goal (`Fail`), or backtracking to try to find other solutions (`Redo`). The second column contains the level of nested calls; the last column lists the goal being examined. Variables the inference engine has not yet unified begin with an underscore, 'G', followed by an

```
1.  Redo: (1) may(hmo, _G11, _G12) ?
2.  Call: (2) areHIPAAdefinitions(hmo, _G21) ?
3.  Call: (3) coveredEntity(hmo) ?
4.  Call: (4) healthPlan(hmo) ?
5.  Exit: (4) healthPlan(hmo) ?
6.  Exit: (3) coveredEntity(hmo) ?
7.  Call: (3) isPHI(_G21) ?
8.  Exit: (3) isPHI(phi) ?
9.  Exit: (2) areHIPAAdefinitions(hmo, phi) ?
10. Call: (2) s164 524 a 1 exception(phi) ?
11. Call: (3) phi for courtProceeding ?
12. Fail: (3) phi for courtProceeding ?
13. Redo: (2) s164_524_a_1_exception(phi) ?
14. Call: (3) subjectToClinicalLab
    Improvements1988_42USC_263a(phi) ?
15. Fail: (3) subjectToClinicalLab
    Improvements1988_42USC_263a(phi) ?
16. Redo: (2) s164_524_a_1_exception(phi) ?
17. Call: (3) exemptFromClinicalLab
    Improvements1988_42CFR_493_3a2(phi)
18. Fail: (3) exemptFromClinicalLab
    Improvements1988 42CFR 493 3a2(phi)
19. Redo: (3) isPHI(_G21) ?
20. Exit: (3) isPHI(psychotherapyNotes) ?
```

Figure 4. Example Trace Execution

integer index (e.g., `_G21`).

We discover preconditions by examining the trace transcript. For example, lines 11-12 in Figure 4, the inference engine attempts and fails to prove the goal `phi for courtProceeding`. We did not express this precondition in our original preconditions sets, so we create a new precondition set, namely:

PS5:
```
assert(coveredUnder(individual, hmo)).
assert(phi for courtProceeding).
```

PS5 asserts the newly identified precondition, along with the original precondition. Preconditions from which similar precondition sets can be constructed are in lines 14-15, and 17-18.

To query the model to validate existing requirements, we first execute the assert statements in a precondition set. Figure 5 displays a transcript for part of an example query execution. For this query, we use precondition set PS5, whose preconditions are asserted in prompts one and two. The query `may(hmo, Permission,Source)` is executed in prompt three. Multiple responses to the query are viewed by pressing the semi-colon (logical-or in Prolog) key. We omit several responses from the model for brevity's sake.

```
1 ?- assert(coveredUnder(individual, hmo)).
Yes

2 ?- assert(phi for courtProceeding).
Yes

3 ?- may(hmo,Permission,Source).
Permission = unreviewable(denies(hmo,
   receives(individual, hmo, phi))),
Source = '164.524(a)(2)(i)' ;

Permission = unreviewable(denies(hmo,
   receives(individual, hmo,
   psychotherapyNotes))),
Source = '164.524(a)(2)(i)' ;

No
```

Figure 5. Sample Query Execution

The engineer must now view the model responses, and determine if they conflict with the existing requirements. The query response indicates that PHI may be withheld from the patient, if it has been prepared for a court proceeding; other queries (not discussed here) have indicated that the HMO has no such permission when the PHI has *not* been prepared for a court proceeding. In the iTrust requirements, however, there is no possibility of preventing the release of such information. This functionality is important, because in certain court cases, information may need to be withheld as evidence. Figure 6 lists two new compliance requirements we identified (with original indexing retained).

All compliance requirements identified during the Analyze Requirements activity need to be verified with

NewReqt$_3$: iTrust shall allow a physician, an administrative assistant, or a medical assistant, using his/her authorized account, to flag diagnostic information or restricted diagnostic information as being used in preparation for court proceedings

NewReqt$_4$: iTrust shall allow a physician, an administrative assistant, or a medical assistant, using his/her authorized account, the option to restrict a patient's access to their diagnostic information or restricted diagnostic information that has been flagged as being used in preparation for court proceedings.

Figure 6. New Potential Requirements

legal domain experts to ensure the law has properly been interpreted. Additionally, stakeholders may choose to change the original requirements in light of the new information, or choose some alternate means of implementing the newly identified requirements other than software, such as business practices.

4. The iTrust Requirements Case Study

In this section, we discuss our case study and findings. Namely, we describe: the materials used in our case study; the terminology mapping activity; the identify precondition sets activity; and our analysis of the iTrust requirements using the production rule model.

4.1. Materials

As mentioned in Section 2.5, we performed our analysis on the 73 iTrust requirements. These requirements are documented in wiki form, which contains a glossary of terms used in the document. Several requirements are annotated with security, legal, and/or engineering concerns.

For requirements validation, we use the production rules model we developed [MA09]. This model covers §164.520, §164.522, §164.524 and §164.526 of the HIPAA Privacy Rule, and is written in SWI-Prolog[4].

Because we did not use a comprehensive production rule model of the HIPAA Security and Privacy Rule, our analysis was governed by the four sections our model covered. Drawing on our knowledge of the Rule, we first analyzed the requirements to determine which sections of the legal text were relevant to each requirement. This analysis is atypical, and is unnecessary when using a comprehensive model of a legal text. Table 3 displays the results of this analysis. Only the sections of the Security and Privacy Rule that have related iTrust requirements are displayed. As indicated in Table 3, our case study focuses on

[4] http://www.swi-prolog.org/

validating the 18 requirements that directly related to the sections our production rule model covers.

Table 3. iTrust Requirements vs. Relevant HIPAA Sections, with Focus Sections Highlighted

HIPAA Section	#of Relevant Requirements
§164.308	3
§164.312	11
§164.506	11
§164.510	1
§164.514	1
§164.520	3
§164.524	14
§164.526	1
§164.528	5

4.2. Mapping iTrust Terminology to Production Rule Model Terminology

Mapping the iTrust terminology to production rule model terminology was easier than the typical case. The iTrust requirements document contains an actor to HIPAA Privacy Rule role mapping, performed by Massey et al. [MOH08]. They used a stakeholder role hierarchy to determine overlap between the iTrust actors and the roles defined in the Rule. We did, however, have to map the objects and relations terminology.

We found the command line interface for the glossary created by the `glossary` and `whatIs` Prolog procedures to be inefficient. For example, if the definition of one term uses another term, we must perform an independent search for the new term. A hyperlinked glossary will be more efficient for such cases. We plan to include such a glossary in our graphical tool discussed in Section 6.

4.3. Identify Precondition Sets from the iTrust Requirements

We identified a total of 14 precondition sets. Precondition set PS1 in Table 2 was our starting point, then we identified the remaining precondition sets as we began to trace query executions. We provided an example of this process in Section 3.2.

Among the 14 precondition sets, we identified three cross-references to external legislation, namely: (a) the Clinical Laboratory Improvement Amendments of 1988[5], (b) the legal text contained at 42 C.F.R. 493.3, and (c) the Privacy Act of 1974[6]. Otto and Antón have identified cross referencing as a difficult problem for regulatory compliance [OA07], because they contain additional compliance requirements. Exploring these cross-references and their implications for legal compliance is an important area for future work.

Trace transcripts are not a perfect method for discovering precondition sets—from our prior experience with the HIPAA Privacy Rule, we knew there are legal preconditions we failed to identify using trace transcripts. Failing to identify precondition sets can lead to noncompliant software. An improved interface will greatly aid in discovering new precondition sets. Specifically, every precondition for a legal rule could be uncovered by a utility for listing all preconditions for a specified goal. We plan on incorporating such a utility into a graphical tool for interaction with production rule models.

4.4. Analyzing the iTrust Requirements for Compliance

As discussed in this subsection, we identified 12 new potential requirements to be added to the iTrust requirements document. In addition, we operationalized an ambiguous nonfunctional requirement.

4.4.1. Identifying New Compliance Requirements

We identified 12 new potential iTrust requirements. To identify these requirements, we make use of rights, obligations, and permissions. These form the principle query mechanisms of our production rule model; each of our queries make use of the rights, obligations, and permission rule patterns [MA09]. The queries we used are listed in Table 4. These queries were paired with the previously identified precondition sets.

Table 4. Queries Used to Identify New Requirements

Q1	`right(individual,CE,receives(individual,CE,phi),Source).`
Q2	`may(CE,X,Source).`
Q3	`may(CE,unreviewable(denies(CE, receives(individual, CE, phi))),Source).`
Q4	`may(correctionalInstitution,unreviewable(denies(correctionalInstitution, receives(individual, correctionalInstitution, phi))),Source).`
Q5	`may(CE,reviewable(X),Source).`
Q6	`must(CE,X,Source)).`

To aid the query process, we are developing tool support for production rule modeling. The tool support is very preliminary. In particular, we employ one feature of the tool support, the use of variable lists in queries. That is, we built an architectural framework that allows for specification of a list of possible values

[5] 42 U.S.C. 263a.
[6] The Privacy Act of 1974, 5 U.S.C. § 552a, 1974.

a variable can hold. The tool support iteratively replaces each of the `CE` variables in Table 4 with each value from a list of covered entities. This mechanism allows us to query the model for each type of covered entity individually, and observe differences between their obligations. We determined, for example, that correctional institutions have several unique obligations, or are released from obligations with which other covered entities must comply.

The majority of the newly identified requirements address exceptions to releasing PHI to a patient. The original iTrust requirements do not include any such exceptions. Two example requirements have been listed in Figure 6, Section 3.3. It is important to note that these requirements may have been identified using other methods. Production rule modeling has an advantage over other methods because familiarity with the legal text is not required.

Traceability to the source section in the legal text is essential, because it aids in demonstrating due diligence [OA07]. For each compliance requirement, we recorded the value unified to the `Source` variable. This variable is the source section in the HIPAA Privacy Rule where the requirement originates from.

4.4.2. Operationalizing Nonfunctional Requirements

Many of the iTrust nonfunctional requirements were not applicable to HIPAA compliance. We were able, however, to operationalize one nonfunctional requirement into a set of functional requirements. Requirement 65 is:

Reqt$_{65}$: iTrust shall have a privacy policy, which is linked off of the login screen.

This requirement is listed as a nonfunctional requirement, because the privacy policy can govern the behavior of the system, but is not an actual part of the system the developers create. That is, the iTrust developers must provide access to the policy via the iTrust login screen, but legal domain experts are responsible for creating the privacy policy. Production rule models, however, can provide insight to the elements of the privacy policy that are required by law.

To operationalize this requirement, we first must map the objects and relations to HIPAA terminology. Using the Prolog procedures specified in Section 3.1, we mapped the iTrust term "privacy policy" to the production rule term "notice", as used in §164.520 of the Privacy Rule.

Using the precondition set and queries listed in Table 5, we were able to determine the potential requirements for the notice a covered entity must maintain. We discovered 18 functional requirements and one nonfunctional requirement. The 18 functional requirements specify the contents of the privacy notice; the one nonfunctional requirement specifies that the privacy policy must be written in plain language.

Table 5. Precondition Set and Queries Used to Operationalize an iTrust Nonfunctional Requirement

Precondition Set	`assert(coveredUnder(individual, healthCareProvider)).`
Query #1	`right(individual, CE, Right, Source).`
Query #2	`must(healthCareProvider, Obligation, Source).`

5. Threats to Validity

Our case study is exploratory and formative; we developed an approach for evaluating existing requirements for legal compliance by analyzing the iTrust requirements. Internal validity is not a concern for exploratory case studies [Yin03]. Internal validity addresses causal relationships. No inferences are made as the result of our case study, so internal validity is not applicable. Construct validity, external validity, and reliability do concern our case study, which we now discuss.

Construct validity addresses the degree to which a case study is in accordance with the theoretical concepts used [CC79, Yin03]. Three ways to reinforce construct validity are: use multiple sources of evidence, establish a chain of evidence, and have key informants review draft case study reports [Yin03]. While we rely on the iTrust requirements document as our single source of evidence, future studies for different sets of requirements will validate and refine our approach developed herein. To establish a chain of evidence, we rigorously followed the steps presented in Section 3 when validating the iTrust requirements for legal compliance, even when we were aware we had neglected to identify legal preconditions and therefore neglected to identify requirements. Finally, our draft case study report has been reviewed by several members of ThePrivacyPlace[7].

External validity addresses the ability of a case study's findings to be generalized to other domains under different settings [Yin03]. We recognize several threats to the external validity of our case study. iTrust is intended for eventual deployment in industry, but is currently only an academic project. We only examine one legal text, the HIPAA Privacy Rule, which regulates only one domain, the healthcare industry. Furthermore, we used a portion of a production rule model, and not a comprehensive model when

[7] www.theprivacyplace.org

performing our analysis, and only analyzed a subset of the iTrust requirements.

Mitigating these threats, the iTrust requirements stakeholders included a physician and a healthcare information professional [MOH08], and thus approximate requirements gathered in industrial settings. Further studies with different requirements artifacts, in other domains, regulated by other legal texts will serve to further validate and refine our approach.

Reliability addresses the ability to repeat a case study and reproduce similar results [Yin03]. A threat to the reliability of our case study is the authors' familiarity with the HIPAA Privacy Rule [BA08, BVA06, MA09]. Allowing previous knowledge of the regulation affect or even guide the query process is a threat to the repeatability of our case study, especially concerning the discovery of new precondition sets. To address this threat, we were extremely cautious to strictly adhere to and follow our approach as a novice would, even when we were aware precondition sets had been missed, as discussed in Section 4.3.

6. Conclusion

This paper an approach to analyze existing requirements for regulatory compliance using queries to a production rule model. These queries enable a requirements engineer to determine whether existing requirements comply with law, as well as aid in identifying new requirements to improve legal compliance. We validated our approach on a set of requirements for the iTrust electronic health records system.

We are currently developing a user-friendly tool support with a graphical user interface (GUI) to support analysis efforts. A strength of production rule modeling is engineers are not required to have intimate knowledge of the legal text. As discussed in Section 4.3, there were several preconditions we did not discover while using the SWI-Prolog command line interface. Utilities we are planning to include in the tool are: a text editor; an interface for authoring, executing, and saving queries; an interface for conducting automated unit testing of the production rule model to aid in maintenance and construction; a viewer to view the text of the regulation, and the production rules that map to each section in the legal text; and a glossary builder and viewer.

Another avenue of future work is to comprehensively model a regulatory text. We cannot be certain the existing or new potential requirements are noncompliant with some portion of the legal text that is not yet modeled. A comprehensive production rule model of the legal text will address this current limitation.

Cross-references pose significant challenges to requirements engineers in determining regulatory compliance [OA07]. For now, we rely on environmental flags to resolve cross-references, but this places the onus on the user to check the external legislation for compliance. The affect of cross-referencing on legal compliance and production rule modeling is necessary to consider when comprehensively modeling a legal text.

Finally, we are designing a human subject experiment to measure the effectiveness of our approach for analyzing existing requirements for legal compliance and identifying potential requirements. The experiment will be carried out by individuals with little familiarity with the HIPAA Privacy Rule, allowing us to validate the claim that production rule models aid engineers with little legal domain knowledge in extracting compliance requirements. This experiment will address the authors' familiarity with the legal text being a threat to reliability, as discussed in Section 6.3.

Acknowledgements

This work was partially funded by NSF grant #0325269. We thank Paul Otto, Aaron Massey, Jessica Young, and Gurleen Kaur for their comments.

References

[AHA03] American Hospitals Association, *AHA Hospital Statistics*, Health Forum LLC, 2009.

[BA08] Breaux, T.D. and Antón, A.I., "Analyzing Regulatory Rules for Privacy and Security Requirements", IEEE Trans. on Software Engineering, Vol. 34, No. 1, Jan.-Feb. 2008, pp. 5-20.

[BAD09] Breaux, T.D., Antón, A.I., and Doyle, J., "Semantic Parameterization: A Process for Modeling Domain Descriptions", ACM Trans. on Soft. Eng. Methodologies, (In Press) 2009.

[BGM85] Borgida, A., Greenspan, S., Mylopoulos, J., "Knowledge Representation as the Basis for Requirements Specifications", *IEEE Computer*, Vol. 18, No. 4, Apr. 1985, pp. 82-91.

[BMT87] Biagioli, C., Mariani, P., and Tiscornia, D., "Esplex: A Rule and Conceptual Model for Representing Statutes", *Proc. of the 1st ACM Intl. Conf. on Artificial Intelligence and Law*, Boston, 1987, pp. 240-251.

[BRR87] Bench-Capon, T.J.M., Robinson, G.O., Routen, T.W., and Sergot, M.J., "Logic Programming for Large Scale Applications in Law: A Formalisation of Supplementary Benefit Legislation", *Proc. of the 1st ACM Intl. Conf. on Artificial Intelligence and Law*, Boston, May 1987, pp. 190-198.

[BVA06] Breaux, T.D., Vail, M.W., and Antón, A.I., "Towards Regulatory Compliance: Extracting Rights and

Obligations to Align Requirements with Regulations", *Proc. of the 14th IEEE Intl. Requirements Engineering Conf.*, Minneapolis, Sep. 11-15, 2006, pp. 46-55.

[CC79] Cook, T.D., Campbell, D.T., *Quasi-Experimentation: Design & Analysis Issues for Field Settings*, Houghton Mifflin: Boston, 1979.

[DHL86] Dubois, E., Hagelstein, J., Lahou, E., Ponsaert, F., and Rifaut, A., "A Knowledge Representation Language for Requirements Engineering", *Transactions of the IEEE*, vol. 74, no. 10. Oct. 1986, pp. 1431-1444.

[GMT87] Greenleaf, G., Mowbray, A., and Tyree, A.A., "Expert Systems in Law: The Datalex Project", *Proc. of the 1st ACM Intl. Conf. on Artificial Intelligence and Law*, Boston, May 1987, pp. 9-17.

[HS08] Ho, Anthony, and Sundaram, Sharada, A Prolog Based HIPAA Online Compliance Auditor, Unpublished class report, Mar. 20, 2008, <www.stanford.edu/class/cs259/projects/ cs259-final-Sharada%20Sundaram%20Anthony%20 Ho/report.pdf>.

[Kil03] P. Kilbridge, "The Cost of HIPAA Compliance", *The New England Journal of Medicine*, Vol. 348, No. 15, pp. 1423-1424, Apr. 10, 2003.

[MA09] Maxwell, J.C., and Antón, A.I., "Developing Production Rule Models to Aid in Acquiring Requirements from Legal Texts", North Carolina State University, Tech. Rep. TR-2008-25, 2008. In submission to: *Proc. of the 17th IEEE Requirements Engineering Conference*, Atlanta, Aug. 31-Sep. 4, 2009.

[Mit08] Mitchell, J.C., Medical Privacy and Business Process Design, Presentation, Stanford Computer Forum, March 17, 2008, <http://forum.stanford.edu/events/2008slides /Security%20 Workshop%20Slides/John%20Mitchell-forum-workshop-08.pdf>.

[MOH08] Massey, A.K., Otto, P.N., Hayward, L.J., and Antón, A.I., "Evaluating Existing Security and Privacy Requirements for Legal Compliance", Under revision for: *Requirements Engineering Journal*, Springer Verlag, 2008.

[OA07] Otto, P.N., and Antón, A.I. "Addressing Legal Requirements in Requirements Engineering", *Proc. of the 15th IEEE International Requirements Engineering Conference*, New Dehli, Oct. 15-19, 2007, pp. 5-14.

[OAB07] Otto, P.N., Antón, A.I., Baumer, D.L., "The Choicepoint Dilemma: How Data Brokers Should Handle the Privacy of Personal Information", *IEEE Security and Privacy*, vol. 5, no. 5, Sep.-Oct. 2007, pp. 15-23.

[Peek97] Peek, N., "Representing Law in Partial information Structures", *Artificial Intelligence and Law*, Vol. 5, No. 4, Dec. 1997, pp. 263-290.

[PTA94] Potts, C., Takahashi, K., and Antón, A.I., "Inquiry-based Requirements Analysis", *IEEE Software*, Vol. 11, No. 2, Mar. 1994, pp. 21-32.

[SAA00] Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W., and Wielinga, B., *Knowledge Engineering and Management: The CommonKADS Methodology*, Cambridge, Mass.: MIT Press, 2000.

[She87] Sherman, D.M. "A Prolog Model of the Income Tax Act of Canada", *Proc. of the 1st ACM Intl. Conf. on Artificial Intelligence and Law*, Boston, May 1987, pp. 127-136.

[SKB91] Sergot, M.J., Kamble, A.S., and Bajaj, K.K., "Indian Central Civil Service Pension Rules: A Case Study in Logic Programming Applied to Regulations", *Proc. of the 3rd ACM Intl. Conf. on Artificial Intelligence and Law*, Oxford, 1991, pp. 118-127.

[SS94] Sterling, L., and Shapiro, E., *The Art of Prolog: Advanced Programming Techniques*, Cambridge, Mass.: MIT Press, 1994, 2nd ed.

[SSK86] Sergot, M.J., Sadri, F., Kowalski, A., Kriwaczek, F., Hammond, P., and Cory, H.T., "The British Nationality Act as a Logic Program", *Comm. of the ACM*, Vol. 29, No. 5, May 1986, pp. 370-386.

[Yin03] Yin, R.K., *Case Study Research: Design and Methods*, in Applied Social Research Methods Series, Vol. 5, Thousand Oaks, CA: Sage Publications, 2003, 3rd ed.