Assessing the Accuracy of Legal Implementation Readiness Decisions

Aaron K. Massey, Ben Smith, Paul N. Otto, and Annie I. Antón
North Carolina State University
Raleigh, NC, USA
{akmassey, ben_smith, pnotto, aianton}@ncsu.edu

Abstract—**Software engineers regularly build systems that are required to comply with laws and regulations. To this end, software engineers must determine which requirements have met or exceeded their legal obligations and which requirements have not. Requirements that have met or exceeded their legal obligations are legally implementation ready, whereas requirements that have not met or exceeded their legal obligations need further refinement. Research is needed to better understand how to support software engineers in making these determinations. In this paper, we describe a case study in which we asked graduate-level software engineering students to assess whether a set of software requirements for an electronic health record system met or exceeded their corresponding legal obligations as expressed in regulations created pursuant to the U.S. Health Insurance Portability and Accountability Act (HIPAA). We compare the assessment made by graduate students with an assessment made by HIPAA compliance subject matter experts. Additionally, we contrast these results with those generated by a legal requirements triage algorithm. Our findings suggest that the average graduate-level software engineering student is ill-prepared to write legally compliant software with any confidence and that domain experts are an absolute necessity. Our findings also indicate the potential utility of legal requirements metrics in aiding software engineers as they make legal compliance decisions.**

Keywords-legal requirements, requirements metrics

I. INTRODUCTION

How do software engineers make legal compliance decisions? The common approach to domain-specific decisions in requirements engineering is to hire or consult with domain experts [26]. Lawyers serve as extremely capable domain experts for legal compliance concerns, but their services are expensive. As a result, lawyers need to be consulted only in situations where software engineers cannot provide accurate or reliable answers. To our knowledge, no researchers have empirically examined if software engineers are able to accurately identify whether software requirements meet or exceed their legal obligations.

The increasingly regulated environments in which software systems must operate makes legal compliance a critical concern for software engineers. In the United States, for example, there are extensive laws and regulations governing software used in areas such as healthcare, corporate accounting, and financial systems. The Health Insurance Portability and Accountability Act of 1996 (HIPAA)[1] prompted the adoption of regulations governing electronic health records (EHR) systems. Three years later, passage of the Graham-Leach-Bliley Act (GLBA) [2] spurred regulations governing the financial services industry. The Sarbanes-Oxley Act of 2002 (SOX)[3] regulates corporate accounting practices. Each of these laws, along with their accompanying regulations, governs a myriad of software systems that must be updated and maintained by software engineers.

When new laws and regulations take effect, or existing laws and regulations are amended, software engineers need to quickly assess the ways in which those legal changes will impact software system requirements. Otto and Antón highlight the need to manage the evolution of laws and regulations over time in order to facilitate legal compliance efforts [30]. For example, in 2009 Congress amended HIPAA with the HITECH Act.[4] Among the HITECH Act's provisions were new data breach notification requirements. In addition, administrative agencies update regulations regularly in response to new laws and changing conditions. Again using the HITECH Act as an example, both the Department of Health & Human Services and the Federal Trade Commission promulgated new regulations pursuant to the HITECH Act within six months of the act becoming law.

In summary, requirements for a software system must comply with governing laws and regulations, which also requires that software engineers stay current with changes in those laws and regulations. As a result, software engineers must be able to discern those requirements that are implementation-ready from those that require further disambiguation and/or elaboration.

[1] Pub. L. No. 104-191, 110 Stat. 1936 (1996).
[2] Pub. L. No. 106-102, 113 Stat. 1338 (1999).
[3] Pub. L. No. 107-204, 116 Stat. 745 (2002).
[4] Pub. L. No. 111-5, tit. XIII, 123 Stat 115, 226 (2009).

978-1-4577-0924-1/11/$26.00 © 2011 IEEE

A requirement is considered to be **Legally Implementation Ready (LIR)** *if it meets or exceeds its legal obligations as expressed in relevant regulations*. In this paper, we discuss a case study in which we employ a set of EHR system requirements, coupled with a particular HIPAA regulatory section with which the system must comply. In this study, we examine three independent assessments of legal implementation readiness for EHR system requirements. First, thirty-two graduate-level computer science students assessed the EHR requirements for legal implementation readiness. These students had no prior experience making such assessments. Second, subject matter experts—individuals with both software engineering and HIPAA compliance expertise—assessed the same set of requirements for legal implementation readiness. Third, we employed our legal requirements triage algorithm [22, 24], which uses legal requirements metrics to assess the same set of requirements for legal implementation readiness and compare the algorithm's results to both the student and expert results. Our legal requirements metrics comprise simple, consistent attributes of a legal text that measure the dependency, complexity, and maturity of each attribute as it relates to the requirements for the software system. We also created a statistical model using our legal requirements metrics to determine whether or not they aid legal compliance decisions.

Our study reveals that graduate-level computer science students exhibit little consensus about LIR requirements, and they poorly identified which requirements were LIR. Our study validates the need for subject matter experts or additional guidance for individuals making legal compliance decisions. Finally, our study reveals that the metrics used in our legal requirements triage algorithm are useful in identifying LIR requirements; the strengths and limitations of the metrics and algorithm are presented herein.

The remainder of this paper is organized as follows. In Section II, we discuss relevant related work. In Section III, we describe our case study design. In Section IV, we present the results of our study. In Section V, we describe potential threats to the validity of our study. Finally, in Section VI, we discuss the meaning of our results and potential future work.

II. Related Work

We examine two distinct areas of related work: (*A*) Legal Requirements and (*B*) Metrics.

A. Legal Requirements

Numerous researchers are actively working with legal requirements as evidenced by the creation of a Workshop on Requirements Engineering and Law (RELAW) [1]. Barth et al. employ a first-order temporal logic approach to model HIPAA, GLBA, and the Children's Online Privacy Protection Act of 1998 (COPPA)[5]—they use this model to demonstrate legal compliance [2]. May et al. employ an access control approach to model laws and regulations, also using HIPAA as an example [20]. Massacci et al. use goal-oriented modeling to analyze Italian data protection laws [21]. Maxwell and Antón employ production rules, an artificial intelligence technique, to model regulations for requirements engineers and check requirements for compliance [26, 27]. In their work, requirements engineers query a production rules model and receive specific answers to questions regarding a modeled regulation [26, 27]. Maxwell and Antón also use HIPAA regulations to illustrate their approach [26, 27].

Researchers are also exploring automated processes for generating traceability links for regulatory compliance [5, 9]. Cleland-Huang et al. use a machine learning approach to automatically generate traceability links for regulatory compliance [9, 10]. Berenbach et al. describe techniques for just in time regulatory traceability [5]. These approaches may reduce the effort required to trace requirements to specific subsections of a legal text, which is a required element of the case study presented in this paper, but they do not currently produce such a mapping. To produce the traceability mapping required for this work, we applied methods described in our prior work [24].

In our prior work, we defined and demonstrated a method for evaluating existing security and privacy requirements for legal compliance [24]. The case study described in this paper differs from that work in three key ways. First, our prior work outlines a generalizable legal compliance methodology, whereas in this work we examine a specific part of this process: assessing the legal implementation readiness of requirements. Second, this work is designed explicitly for iterative software development, whereas our prior work was not fully adapted for use in an iterative software development process [24]. Third, our methodology in our prior work required manual examination of each requirement [24]. In this work, we examine the accuracy of automated assessments using legal requirements metrics.

B. Metrics

Metrics are used in many domains such as insurance contracts and policy documents to empirically and objectively analyze natural language text. These metrics are useful for accurately examining aspects of natural language documents without requiring extensive analysis. Consider the Flesh-Kincaid Grade Level model [12]. In this model, the readability of a document can be accurately calculated without consulting numerous linguistics experts by measuring the readability of texts written in English by examining, for example, the average syllables per word and the average length of a sentence [12]. Despite the simple nature of their construction, metrics like the Flesch-Kincaid Grade Level metric prove useful in practice. Flesh-Kincaid, in particular, is included in numerous word processing systems and is a United States government standard [32]. Other models also exist to achieve this end, including the Gunning-Fog Index [13], the SMOG Index [19], and the Automated Readability Index [16].

[5] Pub. L. No. 105-277, tit. XIII, 112 Stat. 2681, 2681-728.

Metrics have also been used to measure software readability. Buse and Weimer's metric model for software readability is based on simple constructs albeit within source code rather than natural language text [8]. For example, they measure the average number of identifiers per line of code and the average number of arithmetic operations per line of code. Using these and other metrics, their model of software readability is useful in examining software quality and provides insight on the design of programming languages.

Metrics must be validated to be empirically useful [17]. Meneely et al. conducted a systematic literature review of metrics validation in software engineering [28]. Their examination identified forty-seven criteria for validating metrics, which sometimes conflicted or overlapped with one another [28]. They concluded that no standard currently exists for validating metrics in software engineering, and therefore researchers should choose a strategy that ensures the metrics they use or develop are validated for their intended use. In this paper, we attempt to do precisely this—confirm that the legal requirements metrics are valid for use in determining whether or not a requirement is LIR.

In this paper, we examine metrics introduced in our own prior analysis of legal requirements triage [22]. Legal requirements triage allows requirements engineers to focus on requirements with legal implications early in the engineering process, avoid expensive and unnecessary refactoring, and demonstrate legal due diligence by incorporating laws and regulations efficiently throughout the engineering effort [22]. Our triage process subdivides a requirements specification into three sets: (a) legal requirements that are ready for design and implementation; (b) legal requirements that require further refinement; and (c) non-legal requirements [22]. The process depends upon eight separate metrics for measuring aspects of a legal text as it relates to a requirements specification [22]. Although we performed a preliminary validation in our prior work, we did not compare the effectiveness of our legal requirements triage with software engineers or domain experts, which is the focus of this paper.

III. Case Study Design

Our case study design employs the Goal/Question/Metric (GQM) approach [3, 4]. The GQM approach is based on the idea that measurement is useful for making intelligent decisions and improving those decisions over time, but that measurement must be focused, and based on goals and models. The GQM approach proposes that every measurement activity have one or more *goals*. Each goal must then have at least one *question* that helps achieve the goal. Finally, each question is answered with one or more *measures* (or metrics).

Our research goal as formulated using the GQM template is as follows:

> Analyze **empirical observations**
> for the purpose of **characterizing legal implementation readiness**
> with respect to **software requirements**
> from the viewpoint of **software engineers**
> in the context of **an EHR system that must comply with HIPAA regulations**.

Given this research goal, we formulate the following six research questions:

Q1. Is there a consensus among subject matter experts on which requirements are LIR?

Q2. Is there a consensus among graduate students on which requirements are LIR?

Q3. Can graduate students accurately [6] assess which requirements are LIR?

Q4. Can we predict which requirements are LIR using attributes of those requirements?

Q5. How do the categories for the legal requirements metrics affect whether a given requirement is LIR? In other words, when the metrics are grouped in the categories defined by our legal requirements triage algorithm, do they remain valid measures of whether or not a requirement is LIR?

Q6. Can we use our legal requirements triage algorithm for automating the process of predicting whether a requirement is LIR?

We introduce our measures for answering these six questions in Section IV-C.

The remainder of this Section details our case study design. In Section III-A, we describe the materials for our research study. In Section III-B, we discuss our participant population. In Section III-C, we describe how we created our canonical set of responses by achieving consensus among experts. In Section III-D, we describe how we used our legal requirements triage algorithm and the legal requirements metrics from which it is built. In Section III-E, we discuss our analysis methodology and the statistical tools we used in our study.

A. Materials

Each case in our case study (described in Section B below) received three inputs: (1) a sample legal text; (2) a requirements specification that includes a glossary of terms; and (3) a traceability mapping of the individual requirements to the legal text. The legal text chosen for this study is 45 C.F.R. § 164.312, a HIPAA regulatory section governing technical safeguards. We selected this section for several reasons: First, it is a part of HIPAA, with which we have extensive experience [7, 22, 24, 25, 26, 27]. Second, it is focused on technical measures for protecting healthcare information, which is something that software engineers are more likely to be familiar with than a less technical HIPAA section. If there is a gap between the results from the experts' canonical classification and that of the graduate students, then we can reasonably infer that this gap would only widen if we were to perform the

[6] Objectively accurate assessment of LIR requirements is impossible due to the inherently subjective nature of interpreting laws and regulations. Thus, we compare against a canonical set of requirements created by experts as described in Section III-C.

study again on more ambiguous or less technically oriented HIPAA sections. Third, it is a relatively short, self-contained section, and selecting it allowed us to test it in its entirety rather than testing an excerpt of a longer HIPAA section.

The requirements used for this case study are from the iTrust Medical Record System requirements specification, an open source EHR system designed by students at North Carolina State University. We selected this system for two reasons: First, we have examined the iTrust system in past studies [22, 24, 25]. Second, the iTrust system shares many characteristics with other existing software systems that must comply with new laws and regulations. Such systems must be evaluated, updated, and improved in order to achieve compliance with each enactment or amendment of relevant laws and regulations.

For purposes of our case study, we modified the iTrust requirements specification. Instead of including all 75 iTrust system requirements, we started with the 15 requirements with legal obligations outlined in HIPAA § 164.312, then iteratively applied our methodology for evaluating requirements for legal compliance [24] to the other iTrust system requirements. After three iterations, we identified an additional 17 requirements for inclusion in our study. We did not completely cover all elements of HIPAA § 164.312, however, and we also included some requirements that only partially covered the elements of the legal text to ensure that some requirements were not LIR. Additionally, we modified our selected requirements to remove terminology and references to other aspects of iTrust not relevant to HIPAA § 164.312.

Our goal in creating this document was not to create a set of requirements that were already perfectly aligned with the law; instead, we sought to examine a variety of legal compliance situations ranging from relatively easy to relatively challenging decisions. We selected one of our 32 requirements to be used as a part of the tutorial for providing basic training to participants about how software engineers may reason about legal compliance. As a result, we had 31 remaining requirements to use in our case study.

The materials also included a traceability mapping of the 31 selected requirements to the specific subsection(s) of HIPAA § 164.312 to which they applied. We constructed the traceability links iteratively using the techniques outlined in our prior work [24].

B. Software Engineering Participants

The participants in our case study are 32 computer science graduate students who have taken or are taking the graduate-level software engineering course at North Carolina State University. Additionally, because they have taken a course in software engineering, they are familiar with the basic practices and responsibilities of a software engineer working as a part of a larger team. Before participating in the study, the participants attended 150 minutes worth of lectures in requirements engineering, and 75 minutes of lecture in regulatory and policy compliance. On the day of the study, they received a brief tutorial on legal compliance in software systems. Prior to the brief tutorial they had never assessed LIR requirements. This tutorial explained some basic concepts in legal compliance for software requirements. It consisted of an introduction to the importance of legal compliance, as well as an explanation of how we constructed the traceability mapping of requirements to the legal text and how that mapping might be useful in evaluating legal compliance.

The tutorial included the following example legal compliance scenario:

> Consider Requirement A:
>
> ***Requirement A:*** *iTrust shall generate a unique user ID and default password upon account creation by a system administrator. [Traces to §164.312(a)(1) and §164.312(a)(2)(i)]*
>
> For reference, here are the two subsections of the legal text to which Requirement A is traced:
>
> (a) (1) **Standard: Access control.** Implement technical policies and procedures for electronic information systems that maintain electronic protected health information to allow access only to those persons or software programs that have been granted access rights as specified in § 164.308(a)(4).
>
> (2) **Implementation specifications:** (i) **Unique user identification (Required).** Assign a unique name and/or number for identifying and tracking user identity.
>
> Requirement A meets or exceeds its legal obligations outlined in §164.312 because no element of the regulation related to the requirement describes different or additional obligations than those described in the requirement.
>
> In contrast, consider Requirement B:
>
> ***Requirement B:*** *iTrust shall allow an authenticated user to change their user ID and password. [Traces to §164.312(a)(1) and §164.312(a)(2)(i)]*
>
> Note that Requirement B traces to the same two subsections of legal text as Requirement A. Requirement B does not meet or exceed its legal obligations outlined in §164.312 because it describes a situation where it is possible for users to end up with the same identifying name or number, which violates §164.312(a)(2)(i).

After walking the participants through this example, we verbally described another potentially conflicting requirement that is a variation on Requirement B. We asked the participants to consider if Requirement B would become LIR if the phrase "so long as the user ID remains unique" were added. After allowing the participants to consider this for a few moments, we showed that it would be possible for a user to adopt a previously discarded user ID as their own. In this situation, a user could change their ID to something totally unique, but their old ID would then become available for another user to use as their ID. This could result in access logs that have a single user ID that represents two separate users.

C. Subject Matter Experts

Canonical sets of LIR requirements can be used to check other requirements sets for legal compliance [23]. To generate a canonical set of LIR requirements, we recruited three subject matter experts: three of the authors of this paper. All three experts have varying years of academic experience with HIPAA legal compliance in software engineering; additionally, one expert has a law degree.

We employed the Wideband Delphi estimation method [6, 33] to determine the consensus subject matter expert opinion to identify the LIR requirements and those requirements needing further refinement. First, we gave each expert the materials described in Section III-A. Next, we asked the experts to individually identify both the LIR requirements and those needing further refinement, recording their rationale for each decision. This individual analysis was used to address our first research question. Third, we held a one-hour-long coordination meeting in which the experts discussed areas of disagreement and worked to arrive at a consensus opinion for each requirement. Their final consensus opinions serve as the canonical LIR requirements set against which we compare responses from the graduate-level computer science participants and the legal requirements triage algorithm.

D. Legal Requirements Triage Algorithm

Our legal requirements triage algorithm uses legal requirements metrics to classify requirements as either LIR or needing further refinement [22, 25]. Legal requirements metrics comprise simple, consistent attributes of a legal text and a set of requirements that must comply with that legal text, such as the number of sections or subsections within a particular regulation. Other attributes include the number of words, the number of cross-references to other sections of legislation, and the number of exceptions within a subsection of a legal text.

We organize our eight legal requirements metrics[7] into three categories of metrics: dependency, complexity, and maturity [22, 25]. *Dependency metrics* estimate the extent to which a requirement may be dependent on other, unknown requirements due to, for example, cross-references to other previously unanalyzed regulations. If a requirement has numerous legal dependencies, then it should be refined into more requirements each with fewer dependencies. *Complexity metrics* estimate the engineering effort required to implement a particular requirement. If a requirement has too much legal complexity, it should be refined to address these complexities. *Maturity metrics* estimate a requirement's ability to be refined to meet its legal obligations. If a requirement is mature, then it has already addressed the legal obligations outlined in the legal text well. Maturity metrics allow requirements traced to inherently complex or ambiguous legal obligations to still be considered LIR since it may not be possible to refine these requirements without considering additional software engineering concerns.

We implemented our legal requirements triage algorithm in Red Wolf [22, 25], a requirements management tool, for the purposes of automating the classification of requirements as either LIR or needing further refinement. Because our algorithm requires the legal text, a set of requirements, and a traceability mapping between the requirements and the legal text, it had the same inputs as the materials given to both the graduate-level software engineering students and the group of experts and practitioners.

E. Analysis Methodology

Weka[8] is a tool used for modeling and data mining. We employed Weka to produce a logistic regression model [15] to predict whether or not each requirement is LIR. Weka performs a regression fit on each data set (the individual student responses, the individual expert responses, and the legal requirements metrics) using the consensus expert results from the Wideband Delphi study for reference and produces coefficients for each logistic regression model. When we substitute values for a given requirement's dependency, complexity, and maturity into the model, the model suggests the likelihood that the requirement is LIR. Weka uses this likelihood to produce a threshold at which a requirement is considered to be LIR. Weka uses this threshold to produce a result that we compare to the canonical set of LIR requirements created by the subject matter experts, which provides values that we use to calculate specificity, sensitivity, and a Kappa statistic for the model. We used the R Project[9], a statistical modeling package, to calculate Fleiss Kappa statistics for the data in our study.

All of our logistic regression prediction models were built using 10-fold cross validation, which is a method for simulating a prediction scenario using smaller amounts of data. In 10-fold cross validation, the original sample is partitioned into 10 subsamples [29]. The model is then trained on 9 of the subsamples, and tested on the remaining subsample. This process is repeated 10 times for each division of subsamples. The results can then be averaged to reach a single estimation of the model's performance.

We formed a consensus using Wideband Delphi [6] for the three subject matter experts. In Wideband Delphi, consensus is formed in rounds with discussion at the end of each round. First, each participant provides answers independently and records their rationale for the answers they have chosen. Next, participants share their answers and reconsider their positions. Then, participants meet to achieve a final consensus. Although we completed the Wideband Delphi technique to produce our canonical set of LIR requirements, we also chose to analyze the results of our independently recorded first analyses to see what

[7] Due to space constraints, we are unable to detail each metric individually in this paper. However, a complete discussion of these metrics can be found in our prior work [22, 25].

[8] http://www.cs.waikato.ac.nz/ml/weka/

[9] http://www.r-project.org/

level of agreement existed among the subject matter experts at the beginning of the process.

IV. Results

First, we describe reactions and questions from our participants. Next, we discuss some of the points of disagreement found during our consensus meeting for the subject matter experts. Then, we analyze the data according to our analysis methodology.

A. Participant Experiences

We conducted our case study with 32 graduate students. The majority (21) of these participants completed the study in the same room at the same time. The remaining participants completed the study individually or in small groups over the course of the next week. Participants were trained using the same training materials, which took about five minutes to introduce and explain at the beginning of the study. No participant had prior experience assessing legal implementation readiness for software requirements. Each participant was then given 45 minutes to analyze 31 requirements and determine whether each requirement was LIR or not. Training was conducted in a group setting when possible, but all participants worked on their own during the 45-minute analysis portion of the study. Although we did not record completion time for individual participants, we observed that most participants completed their study five or more minutes prior to the deadline. In addition, no participant failed to indicate whether they believed a requirement was LIR or not for any of the 31 requirements.

We answered clarifying questions during each session, and for each question asked we repeated the question and the answer for every participant. Because the first session was the largest group to participate, we assembled all of the questions that could not be answered by referring to other material within the requirements specification or legal text provided to the participants; we then provided those questions and corresponding answers to subsequent groups as part of their training material. For example, a question about the definition of an invalid employee account was not repeated because it is listed as a term in the glossary provided. However, a question about whether or not an email would be considered part of an access control log (answer: they are not) was included in subsequent training material.

B. Subject Matter Expert Discussion

As described in Section IV-C, our subject matter experts entered the Wideband Delphi consensus meeting already having achieved a moderate level of consensus based on their individual responses. However, there were still 12 out of 31 requirements for which the subject matter experts did not achieve universal agreement after the first round of Wideband Delphi.

In some cases, two subject matter experts immediately accepted the rational used by one subject matter expert to denote a non-LIR requirement. Consider the following requirement:

R-18: *Whenever a new employee account is created, iTrust shall create a record of the employee account created and the system administrator who created it in the access control log. [Traces to §164.312(b)]*

Note that HIPAA § 164.312(b) reads as follows:

> **Standard: Audit controls.** Implement hardware, software, and/or procedural mechanisms that record and examine activity in information systems that contain or use electronic protected health information.

Two of the subject matter experts believed that this requirement met or exceeded its legal obligations. However, one subject matter expert believed that a record should also be created in the access control log for each unsuccessful attempt to create a new employee account as well as the successfully created employee accounts, pursuant to the broad language of HIPAA § 164.312(b). The other experts agreed with this rationale, and since no other requirement in the requirement set describes this activity, R-18 was found to be in need of further refinement in the canonical requirements set.

In other cases, the subject matter experts took some time to debate whether or not a requirement should be considered LIR. In iTrust, any employee is allowed to generate a printable summary of a patient's medical record for the purposes of handling emergency medical scenarios. Part of this functionality is described by the following requirement:

R-26: *Each time a printable emergency report for a medical record is generated, iTrust shall email a notification to the patient to whom the record pertains and to all of that patient's personal representatives. [Traces to §164.312(b)]*

This requirement traces to the same section of HIPAA as our previous example, but the discussion of whether or not it is a LIR requirement is more nuanced. The experts debated several considerations: First, they determined that email notification does not qualify as a record in an access control log. Since R-25 describes the creation of such a record in the exact same printable emergency report scenario, R-26 could not be found in need of this refinement. Second, the experts debated whether or not an email notification qualified as a "procedural mechanism that records or examines activity" as prescribed by HIPAA § 164.312(b). On one hand, email notification may allow a patient some oversight through the ability to examine the activity of generating a printable emergency report. On the other hand, email notification does not ensure that a patient *will* examine the activity. This second position is a broader interpretation of HIPAA § 164.312(b), and the experts agreed that assuming the broader standard was the safer course of action to ensure legal compliance. Third, the subject matter experts determined that R-26 should not be traced to HIPAA § 164.312(b) because notification is a separate action from either 'recording' or 'examining activity.' Ultimately, the experts reached consensus that R-26 is LIR because it does not describe any action that violates any legal obligation.

C. Data Analysis

To analyze our results, we built statistical models of the subjects' results, the experts' results, and the algorithm's results. These predictive models allow us to compare the results from each group. Since any predictive model for classifying requirements as LIR or not LIR will either do so either correctly or incorrectly, we can compare the rates of correct responses versus the rates of incorrect responses. For a given test of each model, some responses will be *true positives*, where the model correctly classifies a requirement as LIR, and other responses will be *true negatives*, where the model correctly classifies the requirement as not LIR. When the model is wrong, some responses will be *false positives*, where the model classifies the requirement as LIR when it was not, and other responses will be *false negatives*, where the model failed to identify an LIR requirement as LIR.

The performance of a given model to classify a requirement as being one of two binary options is often evaluated using two measurements: sensitivity and specificity [31]. Sensitivity measures the ability of a given model to predict *positives.* In the case of LIR, this is a weighted measurement of how many LIR requirements the model classifies as not LIR. From a software engineering standpoint, this situation describes a scenario in which requirements that already meet or exceed their legal obligations are considered for further refinement. Sensitivity is defined in Equation 1, where **tp** is the number of true positives and **fn** is the number of false negatives.

$$Sensitivity = \frac{tp}{tp + fn} \quad (1)$$

Specificity measures the ability of a given model to predict *negatives*. In the case of LIR, this is a weighted measurement of how many non-LIR requirements the model classifies as LIR. From a legal compliance standpoint, this is a dangerous situation because it may result in requirements that have not met or exceeded their legal obligations being implemented. Specificity is defined in Equation 2, where **tn** is the number of true negatives and **fp** is the number of false positives.

$$Specificity = \frac{tn}{tn + fp} \quad (2)$$

The Fleiss Kappa statistic [11], **κ**, measures level of agreement between raters on a given set of subjects. In this paper, we employ the Fleiss Kappa statistic to calculate the level of agreement between subjects' determinations about whether a requirement is LIR or not. The Fleiss Kappa statistic ranges from –1 to 1. The value 1 reflects perfect agreement between raters, the value –1 indicates perfect disagreement, and the value 0 indicates the amount of agreement that would be expected by random chance. Every value between 0 and 1 reflects some level of agreement, with larger values indicating more agreement.[10]

We now discuss the results of our case study for each of our research questions identified in Section III.

Q1. Is there a consensus among subject matter experts on which requirements are LIR?
Measure: *Fleiss Kappa statistic for three subject matter experts' results requirements.*

The Fleiss Kappa statistic for our three experts was **κ** = 0.517 ($p < 0.0001$). This result indicates that with a high level of statistical significance, our experts moderately agreed on their first assessment of the 31 requirements before completing the Wideband Delphi session to reach consensus.

Answer: *We can say that there is a moderate consensus among experts regarding LIR.*

Q2. Is there a consensus among graduate students on which requirements are LIR?
Measure: *Fleiss Kappa statistic for 32 graduate students on 31 requirements.*

The Fleiss Kappa statistic for our 32 students was **κ** = 0.0792 ($p < 0.0001$). This result indicates that with a high level of statistical significance, the students in our case study had only a slight agreement on their assessment of the 31 requirements. Because Fleiss Kappa accounts for random chance, this level of agreement is only slightly better than what would be expected from a random set of responses. Due to the high level of significance in both Kappa scores for Q1 and Q2, we can say that there was a higher level of consensus among experts than students.

Answer: *We can say that there is little consensus among students regarding LIR.*

Q3. Can graduate students accurately assess which requirements are LIR?
Measures: *Sensitivity, specificity, and Fleiss Kappa statistic for graduate students' and subject matter experts' results for 31 requirements.*

To answer Q3, we formed a consensus among the 32 students' votes on whether each of the 31 requirements was LIR or not. We used a simple majority to form consensus between the students: if more than 50% of the students said that a given requirement was LIR, the consensus was that the requirement was LIR. Otherwise, the consensus was that the requirement was not LIR. We used these consensus values to measure the ability of the

[10] In this paper, we use Landis and Koch's definitions for levels of agreement (e.g. poor, slight, fair, moderate, substantial, and almost perfect) for the Fleiss Kappa statistic [18], although we recognize their description has been contested [14]. Regardless, we report numerical values for each use of the Fleiss Kappa statistic.

students to accurately predict whether the requirement was LIR, using the experts' consensus as an oracle. Using this technique, the students' consensus had sensitivity of 0.875 and specificity of 0.20, with $\kappa = 0.076$ ($p < 0.0001$). These values mean that the students were better at predicting a requirement as LIR and likely to miss a requirement that is not LIR.

Answer: *We can say that students cannot accurately assess the LIR status of a requirement and are more likely to miss requirements that are not LIR.*

Q4. Can we predict which requirements are LIR using attributes of those requirements?

Measures: *Sensitivity, specificity, and Fleiss Kappa statistic for a statistical model built based on the legal requirements triage algorithm's results for 31 requirements.*

We used our legal requirements metrics to produce a logistic regression model that predicts whether or not requirement is LIR. Using this model to predict the experts' opinions, we achieved sensitivity of 0.625 and specificity of 0.80, with $\kappa = 0.35$ ($p < 0.0001$). These values mean that the model was more likely to miss a requirement that was LIR, sending a requirement to experts for evaluation that required little to no more evaluation, than to say a requirement was LIR that was not. The model also had a higher specificity than the students, meaning it was more likely to catch requirements that were not LIR than the students. The Fleiss Kappa score for this comparison also indicates a fair agreement between the model and the experts, with the model agreeing with the experts more than the students do.

Answer: *We can say that our legal requirements metrics can be used to make a model that is useful in predicting whether a requirement is LIR status.*

Q5. How do the categories for the legal requirements metrics affect whether a given requirement is LIR? In other words, when the metrics are grouped in the categories defined by our legal requirements triage algorithm, do they remain valid measures of whether or not a requirement is LIR?

Measures: *Complexity, Maturity, and Dependency for 31 requirements.*

As described in Section III-E, Weka fits a logistic regression function to the dataset to create the model we used to predict the LIR status of each requirement. The coefficients of this resultant logistic regression function tell us how each metric affects the probability that a requirement is LIR for this dataset.

If the coefficient for the metric's term in the model is *negative*, higher values of the metric mean it is less likely that metric is LIR. If the coefficient for the metric's term in the model is *positive*, higher values of the metric mean it is more likely that the metric is LIR. The table below presents the signs for the coefficients in our model.

Term	Coefficient Sign
Dependency	Negative
Complexity	Negative
Maturity	Positive

Answer: *These results indicate that Dependency and Complexity make the metric less likely to be LIR, and Maturity makes the metric more likely to be LIR.*

Q6. Can we use our legal requirements triage algorithm for automating the process of predicting whether a requirement is LIR?

Measures: *Sensitivity, specificity, and Fleiss Kappa statistic for algorithm and experts on 31 requirements.*

To answer Q6, we executed the algorithm on the 31 requirements and obtained its classifications for whether each requirement was LIR or not. We used these values to measure the ability of the algorithm to accurately predict whether the requirement was LIR, using the experts' consensus as an oracle. Using this technique, the algorithm had sensitivity of 0.5 and specificity of 0.466, with $\kappa = -0.03$ ($p < 0.0001$). These values indicate that the algorithm was slightly better at predicting requirements that were LIR than it was at predicting requirements that were not LIR. The Fleiss Kappa value indicates that the model often disagreed with the expert opinion. However, the algorithm was better at predicting requirements that were not LIR than the students, although not as good at predicting either LIR requirements or not LIR requirements as the subject matter experts.

Answer: *We can say that the algorithm is not useful for predicting LIR in its current form; however, the algorithm would be less likely to miss a requirement that is not LIR than the students.*

V. THREATS TO VALIDITY

In this section, we present the issues and concerns that may threaten the validity of this research. **Internal validity** refers to the causal inferences made based on experimental data [34]. Computer science graduate students may identify different requirements as LIR than software engineers working in industry. In fact, we did not ask students to report their level of previous industry experience, which may affect inferences in this study. We plan to conduct a similar study using industry practitioners to further validate the inferences made in this research.

Construct validity refers to the appropriateness and accuracy of the measures and metrics used for the concepts studied [34]. The primary measure we use is the canonical set of LIR requirements generated by the subject matter experts using a Wideband Delphi procedure. The use of a canonical set of LIR requirements to test other requirements for legal compliance is an accepted practice

[23]. In addition, Wideband Delphi is an accepted practice for achieving consensus in software engineering activities, such as formally reasoning about specifications or assessing natural language texts for requirements [6]. However, the three subject matter experts worked together previously on the concept of LIR requirements. As a result, they are not entirely independent despite conducting the first phase of our Wideband Delphi consensus procedure separately. Also, our subject matter experts have not developed a commercial EHR system.

Another construct validity concern is the choice of statistical tools and modeling techniques. Our choice of sensitivity and specificity was based on the fact that a false negative (identifying a requirement as needing refinement when it is actually LIR) has a low penalty whereas a false positive (identifying a requirement as LIR when it really needs further refinement to meet its legal obligations) has a high penalty. Precision and recall [31] are similar statistical tools that may be better estimators, but these statistics treat false positives and false negatives as equally problematic.

A modeling technique other than logistic regression might perform better at predicting whether or not requirements are LIR. We considered many alternative modeling techniques and used Weka to compare the sensitivity and specificity for each technique. Based on these scores, Weka reveals that logistic regression modeling consistently outperform the other choices available in the Weka modeling toolkit.

The answers we provided to participants' questions during the study are another source of potential construct validity concerns. For example, one participant asked about the difference between *Required* and *Addressable* as used in the legal text. We explained that *Required* indicates a legal obligation to implement all elements described in that subsection as closely as possible, whereas *Addressable* indicates that the general concept must be present but does leave some implementation details up to the specific software engineer. Our explanation for *Addressable*, however, was an inadvertent misrepresentation of the term's precise legal meaning, provided in HIPAA § 164.306(d)(3). *Addressable* subsections allow entities to assess whether the subsection's requirements are "reasonable and appropriate" for their systems; if not, then entities may instead implement equivalent measures that are in fact "reasonable and appropriate" for their system, or explain why no measures are required. Our misrepresentation was not inconsistent with the legal text, and our explanation implied that all *Addressable* subsections had been predetermined to be "reasonable and appropriate" for implementation in iTrust.

External validity refers to the ability to generalize findings and results to other domains [34]. By selecting a highly technical subsection of HIPAA that should have played to the technical background of our participants, we believe that we have established a valid baseline from which we can generalize to other sections of HIPAA or other pieces of legislation. In other words, because our participants exhibited little consensus when examining a piece of legislation with an emphasis on technology standards, we believe that they would exhibit even less consensus on a less technical and more legally oriented piece of legislation. In addition, we plan to conduct additional studies on broader sections of HIPAA.

Another external validity concern is that graduate students may perform differently if they were in a genuine legal compliance scenario. For purposes of this case study, we had to limit the time and resources that our participants were allowed to use while reviewing the requirements for legal compliance.

Reliability refers to the ability of other researchers to repeat a case study [34]. We assiduously documented our process and procedures while conducting our case study. In addition, we can provide exact copies of all materials used in our study for other researchers interested in repeating it. As a result, we do not believe reliability is a concern for this study.

VI. Discussion

The need to better understand how we can support software engineers in developing legally compliant software systems is clear. It is increasingly important for software engineers to manage laws and regulations during development. Legal compliance may ultimately become the single most important non-functional requirement for a large number of software systems.

This research supports the goal of understanding legal compliance in software systems by providing an empirical baseline for the accuracy of legal compliance decisions made by software engineers. As we have shown in this paper, graduate-level computer science students with a background in software engineering are ill-prepared to make legal compliance decisions with any confidence. These students exhibit little consensus regarding whether or not requirements are LIR, and they disagreed with our subject matter experts on most decisions. To our knowledge, no other research exists on this topic. If our participant population of computer science graduate students does not differ substantively from the average entry-level software engineer, then this is a crucial finding for organizations that must develop software that complies with laws and regulations.

Our research further illustrates the value of involving subject matter experts in evaluating whether or not requirements are LIR. The subject matter experts bested the study's participants even before conducting the consensus meeting. We note, however, that 12 out of 31 requirements did not enjoy universal agreement entering the consensus meeting. The experts' initial differences illustrate the specific difficulty of determining whether or not requirements are LIR and the general challenge of dealing with ambiguity-laden legal texts. We suspect that even subject matter experts would benefit from structured, metric-based support in making their assessments of whether or not a given requirement is LIR.

Additionally, we have shown that legal requirements metrics are a promising area of research for supporting

software engineers as they face legal compliance decisions. Metrics are used in numerous domains to quickly and accurately examine aspects of natural language and source code. Despite the fact that the legal requirements triage algorithm performed poorly, we were able to show that prediction models based on the same metrics the algorithm was using performed quite well at predicting whether a requirement is LIR. We plan to use the results of this research to improve the algorithm in our future work. For example, the coefficients developed by the prediction model for the *Dependency*, *Complexity*, and *Maturity* terms could replace the weighting factors we discussed when creating the legal requirements triage algorithm [22]. Even if this approach fails to significantly improve the results, we could simply build a prediction model as a part of the legal requirements triage process.

ACKNOWLEDGMENT

This work was partially supported by NSF ITR Grant #522931 and NSF Cyber Trust Grant #0430166. The authors would like to thank the members of The Privacy Place for their feedback on early drafts of this paper.

REFERENCES

[1] A.I. Anton, T.D. Breaux, D. Karagiannis, and J. Mylopoulos. *First International Workshop on Requirements Engineering and Law (RELAW)*, pp. i –iv, 2008.

[2] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. "Privacy and Contextual Integrity: Framework and Applications," *IEEE Symposium on Security and Privacy*, pp 184–198, 2006.

[3] V. R. Basili, "Software Modeling and Measurement: The Goal/Question/Metric Paradigm," *University of Maryland, CS-TR-2956*, UMIACS-TR-92-96, September 1992.

[4] V. R. Basili, "Applying the Goal/Question/Metric Paradigm in the Experience Factory," *Software Quality Assurance and Measurement: Worldwide Perspective*, Fenton, Whitty, and Lizuka, eds., Thomson Computer Press, pp. 21-44, 1995

[5] B. Berenbach, D. Grusemann, and J. Cleland-Huang, "The Application of Just In Time Tracing to Regulatory Codes and Standards," *8th Conf. on Systems Engineering Research*, 2010.

[6] Barry W. Boehm. "Software Engineering Economics," *Prentice Hall PTR*, Upper Saddle River, NJ, USA, 1981.

[7] T. D. Breaux and A. I. Antón. "Analyzing Regulatory Rules for Privacy and Security Requirements," *IEEE Transactions on Software Engineering*, 34(1):5–20, Jan. 2008.

[8] Raymond P.L. Buse and Westley R. Weimer. "A Metric for Software Readability," *In Proceedings of the 2008 International Symposium on Software Testing and Analysis (ISSTA '08)*, pages 121–130, New York, NY, USA, 2008. ACM.

[9] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, "A Machine Learning Approach for Tracing Regulatory Requirements Codes to Product Specific Requirements," *32nd International Conference on Software Engineering*, May 2-8 2010.

[10] Chuan Duan, Paula Laurent, Jane Cleland-Huang, and Charles Kwiatkowski. "Towards Automated Requirements Prioritization and Triage," *Requirements Engineering*, 14(2):73–89, 06 2009.

[11] J. L. Fleiss, and J. Cohen, "The Equivalence of Weighted Kappa and the Intraclass Correlation Coefficient as Measures of Reliability," *Educational and Psychological Measurement*, 33:613–619, 1973.

[12] R. F. Flesch. "A New Readability Yardstick," *Journal of Applied Psychology*, 32:221–233, 1948.

[13] R. Gunning. "The Technique of Clear Writing," *McGraw-Hill International Book Co*, New York, 1952.

[14] K.L. Gwet, "Handbook of Inter-Rater Reliability" 2nd ed, Gaithersburg : Advanced Analytics, LLC, 2010.

[15] D.W. Hosmer, and S. Lemeshow. "Applied Logistic Regression," 2nd ed. *Wiley*, 2000.

[16] J. P. Kinciad and E. A. Smith. "Derivation and Validation of the Automated Readability Index for use with Technical Materials," *Human Factors*, 12:457–464, 1970.

[17] B. Kitchenham, S.L. Pfleeger, et al. "Towards a Framework for Software Measurement Validation," *IEEE Transactions on Sofware Engineering*, 21(12): 929-944.

[18] J.R. Landis, and G.G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*. 33:159–174, 1977.

[19] G. H. McLaughlin. "SMOG grading – a new readability," *Journal of Reading*, May 1969.

[20] M. J. May, C. A. Gunter, and I. Lee, "Privacy APIs: Access Control Techniques to Analyze and Verify Legal Privacy Policies," *Proceedings of the Computer Security Foundations Workshop*, pp. 85 – 97, 2006.

[21] F. Massacci, M. Prest, and N. Zannone, "Using a security requirements engineering methodology in practice: The compliance with the Italian data protection legislation," *Computer Standards & Interfaces*, vol. 27, no. 5, pp. 445–455, 2005.

[22] A.K. Massey and A.I. Antón. "Triage for Legal Requirements," *Technical Report (TR-2010-22), North Carolina State University*, 2010.

[23] J.C. Maxwell, A.I. Antón, "The Production Rule Framework: Developing a Canonical Set of Software Requirements for Compliance with Law", *1st ACM Intl. Health Informatics Symposium*, 2010.

[24] Aaron Massey, Paul Otto, Lauren Hayward, and Annie Antón. "Evaluating Existing Security and Privacy Requirements for Legal Compliance." *Requirements Engineering*, 15:119–137, 2010.

[25] A.K. Massey, P.N. Otto, and A.I. Anton. "Prioritizing Legal Requirements," *In Second International Workshop on Requirements Engineering and Law (RELAW)*, pages 27–32, 2009.

[26] J. C. Maxwell and A. I. Antón, "Developing Production Rule Models to Aid in Acquiring Requirements from Legal Texts," *17th IEEE International Requirements Engineering Conference*, pp. 101 –110, 31 2009-Sept. 4 2009.

[27] J. C. Maxwell and A. I. Antón, "Validating Existing Requirements for Compliance with Law Using a Production Rule Model," *Proc. of the 2nd Intl. IEEE Workshop on Requirements Engineering and the Law (RELAW)*, pp. 1–6, 2009.

[28] A. Meneely, B. Smith, and L. Williams. "Software Metrics Validation Criteria: A Systematic Literature Review," *Technical Report (TR-2010-2), North Carolina State University*, 2010.

[29] G.J, McLachlan, K.A. Do, and C. Ambroise. "Analyzing Microarray Gene Expression Data," *John Wiley & Sons, Inc.*, Hoboken, New Jersey, 2004.

[30] Paul N. Otto and Annie I. Antón. "Addressing Legal Requirements in Requirements Engineering," *15th IEEE International Requirements Engineering Conference*, pages 5–14, 15-19 Oct. 2007.

[31] D. L. Olson and D. Delen, "Advanced Data Mining Techniques," Berlin Heidelberg: *Springer*, 2008.

[32] L. Si and J.Callan. "A Statistical Model for Scientific Readability," *10th Int'l Conf. on Information and Knowledge Management*, pp. 574–576, 2001.

[33] Andrew Stellman and Jennifer Greene. "Applied Software Management," *O'Reilly Media*, November 2005.

[34] R. K. Yin. "Case Study Research: Design and Methods," Volume 5 of Applied Social Research Methods Series. *Sage Publications*, 3rd edition, 2003.