Identifying and Classifying Ambiguity for Regulatory Requirements

Aaron K. Massey*, Richard L. Rutledge*, Annie I. Antón*, Peter P. Swire†
*School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA, USA
{akmassey, rrutledge, aianton}@gatech.edu
†Scheller College of Business, Georgia Institute of Technology, Atlanta, GA, USA
{Peter.Swire}@scheller.gatech.edu

***Abstract*—Software engineers build software systems in increasingly regulated environments, and must therefore ensure that software requirements accurately represent obligations described in laws and regulations. Prior research has shown that graduate-level software engineering students are not able to reliably determine whether software requirements meet or exceed their legal obligations and that professional software engineers are unable to accurately classify cross-references in legal texts. However, no research has determined whether software engineers are able to identify and classify important ambiguities in laws and regulations. Ambiguities in legal texts can make the difference between requirements compliance and non-compliance. Herein, we develop a ambiguity taxonomy based on software engineering, legal, and linguistic understandings of ambiguity. We examine how 17 technologists and policy analysts in a graduate-level course use this taxonomy to identify ambiguity in a legal text. We also examine the types of ambiguities they found and whether they believe those ambiguities should prevent software engineers from implementing software that complies with the legal text. Our research suggests that ambiguity is prevalent in legal texts. In 50 minutes of examination, participants in our case study identified on average 33.47 ambiguities in 104 lines of legal text using our ambiguity taxonomy as a guideline. Our analysis suggests (a) that participants used the taxonomy as intended: as a guide and (b) that the taxonomy provides adequate coverage (97.5%) of the ambiguities found in the legal text.**

I. Introduction

> Most people who bother with the matter at all would admit that the English language is in a bad way, but it is generally assumed that we cannot by conscious action do anything about it.
>
> – George Orwell

Orwell's *Politics and the English Language* details ways authors conceal their actual meaning behind vague or ambiguous language. He believed much of this was due to sloppiness, and that writers could actually do something about it, but for readers of ambiguous language, rewriting is not an option. More importantly, ambiguity sometimes accurately conveys an authors intent. Legal texts are sometimes intentionally ambiguous [1]. Requirements engineers have long recognized that natural language is often ambiguous [2]. Resolving ambiguities in source documents for requirements remains an area of active research. In particular, researchers have not focused on identifying ambiguities in legal texts that govern software systems, which is critical because ambiguities in legal texts can neither be ignored nor easily removed. Many approaches to resolving ambiguity in software requirements rely on disambiguation or removal of the ambiguity. These may simply not be an option for software engineers addressing ambiguity in a legal text. This paper explores ambiguity in a legal text from the U.S. healthcare domain and whether software engineers can actually do something about it.

Our prior research focused on compliance with the Health Insurance Portability and Accountability Act (HIPAA)[1] [3], [4]. Non-compliance with HIPAA can result in significant fines. The U.S. Department of Health and Human Services (HHS) fined WellPoint $1.7 million[2], a Massachusetts healthcare provider $1.5 million[3], and Cignet Health $4.3 million[4] for non-compliance with HIPAA. In 2009, Congress amended HIPAA with the HITECH Act, which was passed as a part of the American Recovery and Reinvestment Act[5]. HITECH outlines a set of objectives that incentivize Electronic Health Record (EHR) systems development by providing payments to healthcare providers using EHRs with certain "meaningful uses," which are further detailed by the U.S. Department of Health and Human Services (HHS), the federal agency charged with regulating healthcare in the United States [5].

The first step for engineers building HITECH-regulated systems is examining the text of the regulation and extracting requirements from it. Unfortunately, extracting software requirements from regulations is extremely challenging [1], [6], [7]. Even reading and understanding these documents may be beyond the capability of professional engineers [8]. Identifying ambiguous statements and understanding why those statements are ambiguous are critical skills for requirements engineers reading legal texts. Even outside of the legal domain, too much unrecognized ambiguity is considered one of the five most important reasons for failure in requirements analysis [9]. To our knowledge, this paper is the first to examine identification and classification of ambiguities in a legal text for the purpose of software requirements analysis.

Many types of ambiguities exist, and each type must be

[1]Pub. L. No. 104–191, 110 Stat. 1936 (1996)
[2]http://www.hhs.gov/ocr/privacy/hipaa/enforcement/examples/wellpoint-agreement.html
[3]http://www.hhs.gov/ocr/privacy/hipaa/enforcement/examples/meei-agreement.html
[4]https://www.huntonprivacyblog.com/2011/02/articles/hhs-fines-cignet-health-4-3-million-for-violation-of-hipaa-privacy-rule/
[5]Pub. L. No. 111–5, 123 Stat. 115 (2009)

978-1-4799-3032-6/14/$31.00 © 2014 IEEE

RE 2014, Karlskrona, Sweden

disambiguated differently by requirements engineers. Herein, we define an ambiguity taxonomy consisting of six broad ambiguity types based on definitions used in requirements engineering, law, and linguistics. *Lexical* ambiguity refers to a word or phrase with multiple valid meanings. *Syntactic* ambiguity refers to a sequence of words with multiple valid grammatical interpretations regardless of context. *Semantic* ambiguity refers to a sentence with more than one interpretation in its provided context. *Vagueness* refers to a statement that admits borderline cases or relative interpretation. *Incompleteness* is a grammatically correct sentence that provides too little detail to convey a specific or needed meaning. *Referential* ambiguity refers to a grammatically correct sentence with a reference that confuses the reader based on the context provided. Each of these types of ambiguity are described in more detail in Section III-A. Some types of ambiguity require additional analysis or disambiguation before implementation can begin. In Section II, we compare and contrast the approaches that prior researchers have taken to different types of ambiguity.

Ambiguities complicate reading, understanding, and examining legal texts for software requirements. Herein, we conduct a case study to determine how prevalent ambiguity is in legal texts. Our findings suggest that ambiguity is prevalent in legal text. In 50 minutes of examination, participants in our case study identified on average 33.47 ambiguities in 104 non-blank lines of legal text[6] using our ambiguity taxonomy as a guideline. Participants did not, however, achieve a strong level of agreement on the exact number and type of ambiguity, regardless of whether we measured agreement over all participants or just over the two groups we examined (technologists and policy analysts).

The remainder of this paper is organized as follows. Section II introduces related work and background information. Section III describes our case study methodology. Section IV details the results of our case study. In Section V, we discuss the implications of this work. Section VI presents potential threats to the validity of this work. Finally, in Section VII, we summarize our work and provide directions for future work in this area.

II. Related Work

The majority of software requirements specifications are written in natural language, which is inherently ambiguous and imprecise [9]. However, software engineers do not yet have a single, comprehensive, accepted definition for ambiguity [10]. Ambiguity has been defined as a statement with more than one interpretation [11]. The IEEE Recommended Practice for Software Requirements Specifications states that a requirements specification is unambiguous only when each requirement has a single interpretation [12]. Lawyers, on the other hand, depend on ambiguity to ensure that laws and regulations are not dependent on transient standards [10]. For example, lawyers might require "reasonable" encryption practices rather than specifying a particular encryption algorithm or standard that might be outdated in a few years. Linguists have created detailed ambiguity classifications. In this section, we present related work on ambiguity in requirements engineering, focusing on legal requirements, and in linguistics.

A. Ambiguity in Requirements Engineering

Common sense suggests that an unambiguous statement would have only a single, clear interpretation. But how should we classify statements that have no interpretations? Vague or incomplete statements may not have a valid interpretation. For a requirements engineer, a statement that depends heavily on domain knowledge may also, at first, appear uninterpretable. Herein, we consider vague or incomplete statements to be ambiguous because they are not unambiguous. That is, we consider them to be ambiguous because they do not have a single, clear interpretation.

Requirements engineers may tolerate requirements with multiple interpretations early in the development of a new set of software requirements [13]. In addition, some statements may be *innocuous* because only one possible interpretation would be reasonable, and these statements are unlikely to lead to misunderstandings [11], [14]. Requirements with statements having more than one reasonable interpretation are *nocuous* and likely to lead to misunderstandings if not clarified [11], [14]. Legal domain knowledge would be required to differentiate between innocuous and nocuous requirements in this study. Since we do not assume our case study participants have the necessary background, we do not consider the difference between nocuous and innocuous to be meaningful. Chantree et al. make an additional distinction between *acknowledged* ambiguities, which are known to engineers, and *unacknowledged* ambiguities, which are unknown to engineers [11]. Our case study focuses only on identifying (i.e. acknowledging) ambiguity in legal texts. We consider unacknowledged ambiguity to be outside the scope of our work.

Many software engineering approaches to ambiguity involve the development of tools or techniques for recognizing or eliminating ambiguity in software requirements. For example, Gordon and Breaux use refinements to resolve potential conflicts between regulations from multiple jurisdictions [15]. Researchers have used natural language processing to detect and resolve ambiguity in software requirements [16]–[18]. Van Bussel developed a machine learning approach to detecting ambiguity in requirements specifications [19]. Popescu et al. developed a semi-automated process for reducing ambiguity in software requirements using object-oriented modeling [20]. None of these approaches focused exclusively on identifying and classifying ambiguity in legal texts to which software systems must comply.

Antón et al. examine conflicts between policy documents and software requirements [21]. Although conflicts between policy documents, legal texts, and software requirements may not necessarily be a form of ambiguity, these conflicts inspired our work in two primary ways. First, Antón et al. state that alignment between policies and software requirements must

[6]We had 16 lines of legal text in our tutorial and 121 total lines in our survey. Seventeen of the lines in our survey were blank.

be flawless to avoid conflicts [21]. Even potential conflict should be addressed [21]. These assertions support the use of a broad definition of ambiguity. Second, although linguists view vagueness or generality as having a single, albeit broad, meaning [22] that is sometimes used to force readers to come to their own understanding or interpretation [23], Antón et al. explicitly state that incompleteness is a form of engineering ambiguity that must be addressed for policy compliance.

B. Ambiguity in Linguistics

Empson's book on literary criticism identifies seven types of ambiguity [23]. This book led to our ambiguity taxonomy for the purposes of evaluation or criticism. Many authors use language simply to provoke a reaction in the reader, and some authors use Empson's ambiguity types for that purpose. We chose not to map Empson's concepts of discovery, incoherence, and division [23] to our taxonomy because their primary utility is for literary criticism or interpretation.

Berry et al. identified linguistic types of ambiguities [10], which they classify according to six broad types, some of which have sub-types. For example, pragmatic ambiguity includes referential ambiguity and deictic ambiguity. Their classification is similar to other classifications of linguistic ambiguity [22]. Berry et al. also examine legal ambiguity [10]. They describe the legal principles used to interpret ambiguity when encountered rather than defining ambiguity. Consider the following legal principle:

> AMBIGUUM FACTUM CONTRA VENDITOREM INTERPRETANDUM EST: An ambiguous contract is to be interpreted against the seller.

This principle does not define ambiguity, rather it provides a mechanism for resolving it in contract law. This principle supports intentionally ambiguous language in legal writing by providing an context in which it can be disambiguated. Unfortunately, requirements engineers do not have the appropriate domain knowledge to interpret this language clearly, and they cannot simply ignore it or remove it. Thus, they must learn to recognize it and seek help from a legal domain expert.

Linguists and philosophers often classify ambiguity in a finer granularity than we do herein. For example, Sennet's syntactic classification ambiguity includes the subtypes phrasal, quantifier and operator scope, and pronouns [22]. Similarly, lexical ambiguity could be classified as either homonymy or polysemy [10]. Linguists and philosophers continue to debate the nature of ambiguity and correct usage of natural language [24]. In particular, classifying types of ambiguity is itself often ambiguous [25]. Even seemingly simple grammatical corrections can quickly balloon into fundamental arguments. Attempting to define what constitutes an arbitrator for "correct" usage in English is extremely challenging [26]. A discussion of the nuance involved in interpreting or correcting language use is outside the scope of this investigation.

III. Case Study Methodology

Our case study methodology is based upon the Goal/Question/Metric (GQM) model [27]–[29]. The GQM model starts with a set of goals. Each goal is addressed by at least one question, with each measured by at least one metric. Following this paradigm focuses the case study on the research questions and minimizes extraneous test participant tasks. Our research goal formulated using the GQM template is:

> Analyze **empirical observations** for the purpose of **characterizing ambiguity identification and classification** with respect to **legal texts** from the viewpoint of **students in a graduate-level Privacy course** in the context of **§ 170.302 in the HITECH Act**.

Given this research goal, we formulate the following questions:

Q1: Does the taxonomy provide adequate coverage of the ambiguities found in § 170.302?

Q2: Do participants agree on the number and types of ambiguities they identify in § 170.302?

Q3: Do participants agree on the number and types of intentional ambiguities they identify in § 170.302?

Q4: Do participants agree on whether software engineers should be able to build software that complies with each paragraph of § 170.302?

Q5: Does an identified ambiguity affect whether participants believe that software engineers should be able to build software that complies with each paragraph of § 170.302?

The remainder of this Section is organized as follows: We first discuss important terminology, providing definitions for each ambiguity type in our taxonomy. Subsection III-B details our participant selection criteria and all materials used for this study. We introduce measures to evaluate each of these questions in Subsection III-C.

A. Terminology

Case study participants were asked to identify ambiguity in the HITECH Act, 45 CFR Subtitle A, § 170.302. We provided participants with a taxonomy that defines six separate types of ambiguity. Table I outlines these ambiguity types. Note that they are not mutually exclusive: a single sentence from a legal text may exhibit more than one ambiguity type. Although this ambiguity taxonomy is designed to be broadly applicable, it is not guaranteed to be comprehensive. Sentences may be ambiguous in ways that do not fall into one of these six types. To introduce our ambiguity taxonomy, we employ example ambiguities identified by our study participants rather than the examples used in our study tutorial, shown in Table I.

Lexical ambiguity occurs when a word or phrase has multiple valid meanings. Consider § 170.302(d): "Enable a user to electronically record, modify, and retrieve a patient's active medication list as well as medication history for longitudinal care." A medication history for longitudinal care could mean either a complete medication history in a particular arrangement or an abbreviated medication history used only for a particular purpose. A requirements engineer must disambiguate this prior to implementation. Another example: "Melissa walked to the bank." This could mean that Melissa walked to a financial institution or she walked to the edge of a river.

TABLE I
CASE STUDY AMBIGUITY TAXONOMY

Ambiguity Type	Definition	Example
Lexical	A word or phrase with multiple valid meanings	Melissa walked to the bank.
Syntactic	A sequence of words with multiple valid grammatical interpretations regardless of context	Quickly read and discuss this tutorial.
Semantic	A sentence with more than one interpretation in its provided context	Fred and Ethel are married.
Vagueness	A statement that admits borderline cases or relative interpretation	Fred is tall.
Incompleteness	A grammatically correct sentence that provides too little detail to convey a specific or needed meaning	Combine flour, eggs, and salt to make fresh pasta.
Referential	A grammatically correct sentence with a reference that confuses the reader based on the context	The boy told his father about the damage. He was very upset.

Syntactic ambiguity occurs when a sequence of words has multiple valid grammatical parsings. Consider § 170.302(f): "Enable a user to electronically record, modify, and retrieve a patient's vital signs…" Here, "electronically" may refer to all the verbs "record, modify, and retrieve" or only to "record." It seems unlikely that the U.S. government wants EHR vendors to "electronically modify a patient's vital signs." But, electronic recording or retrieving seem like reasonable requirements. Again, a requirements engineer must disambiguate prior to implementation. Also: "Quickly read and discuss this paragraph."

Semantic ambiguity occurs when a sentence has more than one interpretation based entirely on the surrounding context. Each word in the sentence has a distinct meaning and the sentence has a single parse tree, but the correct interpretation of the sentence requires more context. Consider § 170.302(j): "Enable a user to electronically compare two or more medication lists." Comparing two lists is reasonably clear if a context for the comparison is provided. These lists could be compared for length, cost, drug interaction, or any number of other factors. In addition, these lists could belong to the same patient or different patients, depending on the comparison's purpose. Other examples: "Fred and Ethel are married." and "Fred kissed his wife, and so did Bob." Further context is needed to determine if Fred and Ethel are married to each other or separately. Nor do we know if Fred has cause to be annoyed.

Vagueness occurs when a term or statement admits borderline cases or relative interpretation. Consider § 170.302(h)(3): "Electronically attribute, associate, or link a laboratory test result to a laboratory order or patient record." What constitutes attributing, associating, or linking? Must these records always be displayed together or would simply having an identifier and allowing a physician to find one given the other suffice? Similarly, consider: "Fred is tall." If Fred was a North American male and 5'2" tall, then the claim is not true. If Fred was 7'0" tall, then the claim is supported. Somewhere in between lie heights that reasonable people might disagree as to constituting "tall."

Incompleteness occurs when a statement fails to provide enough information to have a single clear interpretation. Consider § 170.302(a)(2): "Provide certain users with the ability to adjust notifications provided for drug-drug and drug-allergy interaction checks." This sentence omits information that would allow requirements engineers to identify which users should have this ability or what options they would have to adjust notifications. Incompleteness must be resolved for the requirements to be implemented. Similarly, "Combine flour, eggs, and salt to make fresh pasta." omits some necessary information such as quantity of materials and techniques to be employed.

Referential ambiguity occurs when a word or phrase in a sentence cannot be said to have a clear reference. Consider § 170.302(n): "For each meaningful use objective with a percentage-based measure, electronically record the numerator and denominator…" The meaningful use objectives that use a percentage-based measure are not referenced directly, which leaves the requirements engineer to determine which objectives must comply with this legal obligation. Other examples include pronouns and their antecedents. "The boy told his father about the damage. He was very upset." The pronoun 'he' could refer to either the boy or the father. Also: "There are many reasons why lawyers lie. Some are better than others."

We created our ambiguity taxonomy based on those ambiguity types that are relevant for regulatory compliance. It is not intended to be comprehensive with respect to all types of ambiguity. A word, phrase, sentence, or paragraph with more than one meaning may not fit in our taxonomy. For this case study, participants were instructed to classify such sentence as an **Other ambiguity**.

Because we chose a section of legal text from the HITECH Act primarily for its important implications for software development, this study was not designed to guarantee that all types of ambiguities appear in the legal text. For example, there may be no referential ambiguity in the legal text. It is also possible that a paragraph from this legal text has a single interpretation with a single, clear meaning. In our taxonomy, such statements are called **Unambiguous statements**.

Requirements engineers can use our ambiguity taxonomy as a guide when evaluating legal text for ambiguity. Each statement could be evaluated for each of the six ambiguity types in sequence from the beginning to the end of the text. If no ambiguity can be found in those six categories, then the requirements engineer could consider the statement to

be unambiguous. Each discovered ambiguity could then be examined for intent. Requirements engineers may be able to disambiguate an intentional ambiguity. For example, the legal phrase "reasonable security practices" is vague, but it could be clarified by a specific government or industry security standard. Unresolved intentional ambiguities and all unintentional ambiguities must be disambiguated by a legal expert.

B. Study Participants and Materials

We selected participants for our case study from a population of all students enrolled in a graduate-level class at the Georgia Institute of Technology, entitled Privacy Technology, Policy, and Law. This course was held during the 2014 spring semester and jointly listed by the College of Computing (CoC) and the Scheller College of Business (CoB). Eighteen students elected to participate.

Our case study materials consisted of a tutorial and a survey. We conducted the tutorial in the class session prior to the survey. During the tutorial, we briefly described the motivation for this research, explained the ambiguity taxonomy, and defined each ambiguity type using illustrative examples for each ambiguity type. After a short question and answer period about the ambiguity types, we presented a worked example of a legal text similar to what the students would be asked to analyze in the survey. The example legal text consisted of a paragraph from the HIPAA[7]. This example provided participants with an experience as similar as possible to that of the survey itself and allowed us to demonstrate each of the types of annotations that might be required of the participants during the survey. During the tutorial, we did not tell the participants which section of legal text would be covered in the survey.

We chose to conduct the tutorial and the survey in consecutive class sessions to allow participants more time to understand our ambiguity taxonomy. At the beginning of the class session during which we conducted the survey, we briefly recapped the tutorial and described two examples for each ambiguity type in our taxonomy. We provided study participants 50 minutes to complete the study. The first question asked the participant to self-identify as one of the following roles:

1) I am a **technologist**, and I am more interested in creating, building, or engineering software systems than I am in legal compliance or business analysis.
2) I am a **business analyst**, and I am more interested in creating a business based on technologies than I am in building technologies.
3) I am a **legal analyst**, and I am more interested in regulatory compliance than I am in building technologies or in business analytics.

We selected the HITECH Act, 45 CFR Subtitle A, § 170.302, which contains 23 paragraphs, as the legal text for this study. This section specifies the certification criteria for EHRs under Meaningful Use Stage 1. Compliance with this regulation is a required qualification for the HITECH incentives that depend upon the use of a certified EHR. Non-compliance with this regulation would result in both regulatory penalties and loss of marketplace reputation. For each paragraph, participants identified ambiguities using a response block. The response block allowed participants to identify ambiguity type(s) identified, the line number on which it was identified, and whether the participant believed it to be intentional (i.e. an ambiguity the author meant to include) or unintentional (i.e. an ambiguity that was accidentally included). The distinction between intentional and unintentional ambiguities is one of the ways that requirements engineers can determine when they must consult a legal expert to resolve the ambiguity.

We created line numbers to simplify the annotation of ambiguities in the legal text. When a paragraph within § 170.302 contained a cross reference to another section of legal text within HITECH, we provided the referenced legal text without line numbers both to provide participants additional context to disambiguate the target legal text and to indicate that cross referenced legal texts were simply provided for context. The response block contained space for each of the six ambiguity types, a space labeled 'Other' for ambiguities that did not fall into one of our six types, and a space labeled 'None' for participants to indicate that the paragraph was unambiguous. If participants identified an ambiguity in the legal text, they wrote the respective line numbers in the appropriate space. If participants believed that the ambiguity was intentional, they also circled the line numbers after writing them. Finally, for each paragraph, participants were asked to agree or disagree with the following statement: "Software engineers should be able to build software that complies with this legal text." We call paragraphs for which participants agree with this statement "implementable," and we call those for which participants disagree "unimplementable." We use responses to this question to determine whether identified ambiguities in § 170.302 affect participants' beliefs about building compliant software.

C. Study Analysis

In addition to the standard mean and variance statistics, we employ two specialized measures for describing group participant agreement. The intraclass correlation coefficient (ICC) measures the variability of a set of responses with quantitative values across N participants [30]. Because we gave the same survey to each participant and performed calculations directly upon responses without first averaging, we employ ICC with the oneway effects model and with a single measure of interest as recommended by McGraw and Wong [30]. In cases where the responses were categorical instead of quantitative, we employ Fleiss' kappa [31]. Both measures compute inter-rater reliability for a fixed number of participants and range from inverse-correlated (-1.0) to un-correlated (0.0) to perfectly correlation (1.0). To perform the statistical computations, we used the R Project[8] with the Interrater Reliability (IRR) package,[9] which supports both ICC

[7]The exact paragraph used in the tutorial was 45 CFR Subtitle A, § 164.312(a).

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

[9]http://cran.r-project.org/web/packages/irr/

and Fleiss' kappa. We analyzed the collected data to answer the questions identified above in Section III as follows:

Q1 Measures: An affirmative answer to this question requires (1) high coverage of identified ambiguities by the taxonomy and (2) minimal use of the "Other" type.

Q2 Measures: We counted the number of ambiguities each participant identified per paragraph and the number and type of each ambiguity found. Since this measure is quantitative, we measured agreement with ICC.

Q3 Measures: We employed the same statistics as with **Q2** with responses restricted to intentional ambiguities. That is, we counted the number of intentional ambiguities each participant identified per paragraph and the number and type of each intentional ambiguity found. Because this measure is quantitative, we measured agreement with ICC.

Q4 Measures: We tabulated participant responses to our question of whether software engineers should be able to build compliant software for each legal paragraph. Because this data is categorical, agreement was measured with Fleiss' Kappa.

Q5 Measures: For paragraphs participants believe to be unimplementable, we calculated the percentage containing identified ambiguities.

IV. Case Study Results

Eighteen students volunteered for our case study. Of these eighteen participants, one provided complete responses to only five of the 23 paragraphs, and we excluded those results. We accepted responses from the remaining seventeen participants, including one participant who identified 55 ambiguities in the first eleven paragraphs and none in the remaining twelve. Some participants failed to provide a response for all parts of the response block for some questions. In each case, we removed those responses from our analysis where appropriate.

As previously mentioned, we asked participants to self-identify as either a: (1) *technologist*, (2) *business analyst*, or (3) *legal analyst*. Because only one participant self-identified as a legal analyst, we combined groups (2) and (3) to produce a new group that we refer to as *policy analysts*. This resulted in a roughly equal division of our seventeen participants with nine in the technologist group and eight in the policy analyst group.

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

Q1: Does the taxonomy provide adequate coverage of the ambiguities found in § 170.302?

Answer: Yes, on average, the participants identified 33.47 ambiguities for the paragraphs in § 170.302, including ambiguities from each type in the taxonomy. The least frequently identified ambiguity type is Semantic with an average of 1.59. The most frequently identified type was Vagueness with an average of 9.82. The 'Other' type had an average of 0.82.

Both technologists and policy analysts identified ambiguities from every type in the taxonomy. Figure 1 shows the total ambiguities identified by participants for each paragraph in § 170.302.[10] It also shows the relative totals for each ambiguity type. Note that paragraph § 170.302(a), in which participants found the most ambiguities, also includes the preamble text that appears at the beginning of § 170.302 and prior to the paragraph.

Table II shows the ambiguities (mean and standard deviation) identified in each paragraph by (1) all participants, (2) technologists, or (3) policy analysts. Intentional ambiguities were relatively rare compared to unintentional ambiguities. Some paragraphs are also appear to be less ambiguous than others. For example, in Table II, paragraphs (b), (j), (o), (r), and (w) each had less than one ambiguity on average for all participants, whereas paragraphs (a), (c), (h), (f), and (n) all had over two ambiguities on average.

The taxonomy ambiguity types overlap and the decision to select one or more types is inherently subjective. Participants used all six ambiguity types more frequently than the "Other" type. Subsection III-A provides examples of each ambiguity type as identified by the participants. Our analysis suggests (a) that participants used the taxonomy as intended: as a guide and (b) that the taxonomy provides adequate coverage (97.5%) of the ambiguities found in §170.302.

In our prior work, we sought to compare participants' understanding of the law to a consensus expert opinion on the law [7]. This comparison worked well as an evaluation technique because rules exist for interpreting laws and regulations. Thus, a correct interpretation can be differentiated from an incorrect interpretation. Unfortunately, conducting a similar comparison to assess the "correctness" of ambiguities as identified and classified by participants in this work is not possible and would be misleading if conducted. Ambiguity is subjective [26]. No absolute authority exists to interpret ambiguity [26], so there is no way to evaluate objective correctness.

Because no correct interpretation of ambiguity exists, any comparison to a consensus expert opinion would misleadingly imply that a correct interpretation exists. In the absence of an objectively correct assessment, expertise is subjective. If a reader finds a statement ambiguous, how can an "expert" prove it is clear? Similarly, if a reader fails to interpret a statement as ambiguous, how can an "expert" prove it to be so?

Analogies to other forms of communication may help illuminate the nature of ambiguity. Comedians do not get to blame their audience for not laughing at their jokes. There is no such thing as an objectively humorous statement. Writers do not get to blame their readers for not understanding their point. Clear communication is the burden of the sender, not the recipient. Thus, in the only sense possible, when participants determine a statement was ambiguous to them, it is ambiguous.

Q2: Do participants agree on the number and types of ambiguities they identified in § 170.302?

Answer: We evaluated agreement using intraclass correlation finding only slight to fair agreement between all partici-

[10] One participant found a total of 55 ambiguities through paragraph § 170.302(k) and none in the remainder of the legal text.

TABLE II
AMBIGUITIES IDENTIFIED IN EACH PARAGRAPH OF THE HITECH ACT, 45 CFR SUBTITLE A, § 170.302

Type	Intent	Paragraphs (a) - (l) (mean, std dev)											
		(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)
Tech	U	2.7, 1.8	0.9, 0.7	2.3, 0.5	1.1, 0.7	1.2, 0.8	2.0, 1.1	1.0, 0.9	1.6, 1.2	0.9, 0.9	0.8, 0.9	1.3, 1.4	1.2, 1.0
	I	0.6, 0.5	0.1, 0.3	0.1, 0.3	0.0, 0.0	0.0, 0.0	0.1, 0.3	0.1, 0.3	0.7, 0.7	0.1, 0.3	0.1, 0.3	0.2, 0.4	0.1, 0.3
	C	3.2, 1.7	1.0, 0.8	2.4, 0.5	1.1, 0.7	1.2, 0.8	2.1, 1.1	1.1, 0.9	2.2, 0.8	1.0, 0.8	0.9, 0.9	1.6, 1.3	1.3, 1.1
Policy	U	3.5, 3.5	0.9, 0.6	1.6, 1.5	1.5, 1.0	1.3, 0.8	2.6, 2.0	1.4, 1.9	2.5, 2.7	1.9, 1.8	0.9, 0.8	1.3, 1.6	0.9, 0.9
	I	0.5, 0.7	0.0, 0.0	0.0, 0.0	0.3, 0.4	0.3, 0.4	0.1, 0.3	0.0, 0.0	0.8, 0.8	0.0, 0.0	0.1, 0.3	0.1, 0.3	0.0, 0.0
	C	4.0, 3.3	0.9, 0.6	1.6, 1.5	1.8, 1.1	1.5, 1.0	2.8, 2.0	1.4, 1.9	3.3, 2.4	1.9, 1.8	1.0, 0.9	1.4, 1.6	0.9, 0.9
Combined	U	3.1, 2.8	0.9, 0.7	2.0, 1.1	1.3, 0.9	1.2, 0.8	2.3, 1.6	1.2, 1.5	2.0, 2.1	1.4, 1.5	0.8, 0.9	1.3, 1.5	1.1, 1.0
	I	0.5, 0.6	0.1, 0.2	0.1, 0.2	0.1, 0.3	0.1, 0.3	0.1, 0.3	0.1, 0.2	0.7, 0.7	0.1, 0.2	0.1, 0.3	0.2, 0.4	0.1, 0.2
	C	3.6, 2.6	0.9, 0.7	2.1, 1.2	1.4, 1.0	1.4, 0.9	2.4, 1.6	1.2, 1.5	2.7, 1.8	1.4, 1.4	0.9, 0.9	1.5, 1.5	1.1, 1.0
		Paragraphs (m) - (w) (mean, std dev)											
		(m)	(n)	(o)	(p)	(q)	(r)	(s)	(t)	(u)	(v)	(w)	
Tech	U	1.3, 0.8	2.2, 1.2	0.6, 0.7	0.7, 1.1	1.0, 0.9	0.4, 0.5	1.0, 0.8	0.2, 0.4	1.3, 0.8	1.0, 0.8	0.7, 0.9	
	I	0.1, 0.3	0.3, 0.7	0.3, 0.5	1.0, 1.1	0.3, 0.5	0.1, 0.3	0.4, 0.7	0.2, 0.4	0.8, 0.4	0.3, 0.7	0.0, 0.0	
	C	1.4, 0.8	2.6, 1.3	0.9, 1.0	1.7, 0.9	1.3, 0.9	0.6, 0.7	1.4, 1.2	0.4, 0.5	2.1, 0.6	1.3, 0.7	0.7, 0.9	
Policy	U	1.0, 1.2	1.5, 1.2	0.5, 0.5	0.6, 0.7	0.6, 0.7	0.8, 0.8	1.0, 1.3	0.8, 1.4	0.9, 0.9	0.4, 0.7	0.5, 0.9	
	I	0.3, 0.4	0.0, 0.0	0.4, 0.7	0.5, 0.7	0.3, 0.4	0.0, 0.0	0.4, 0.5	0.1, 0.3	0.4, 0.5	0.3, 0.4	0.0, 0.0	
	C	1.3, 1.3	1.5, 1.2	0.9, 0.9	1.1, 1.1	0.9, 0.6	0.8, 0.8	1.4, 1.2	0.9, 1.4	1.3, 1.0	0.6, 0.7	0.5, 0.9	
Combined	U	1.2, 1.0	1.9, 1.3	0.5, 0.6	0.6, 0.9	0.8, 0.9	0.6, 0.7	1.0, 1.1	0.5, 1.0	1.1, 0.9	0.7, 0.8	0.6, 0.9	
	I	0.2, 0.4	0.2, 0.5	0.4, 0.6	0.8, 0.9	0.3, 0.5	0.1, 0.2	0.4, 0.6	0.2, 0.4	0.6, 0.5	0.3, 0.6	0.0, 0.0	
	C	1.4, 1.1	2.1, 1.3	0.9, 1.0	1.4, 1.0	1.1, 0.8	0.6, 0.8	1.4, 1.2	0.6, 1.0	1.7, 0.9	1.0, 0.8	0.6, 0.9	

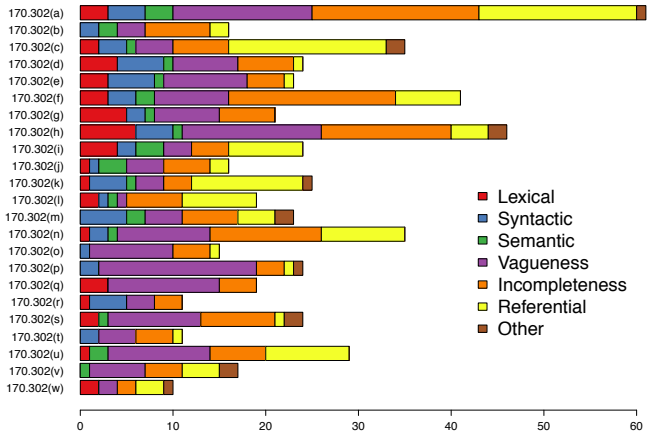

Fig. 1. Ambiguities identified in each paragraph of the HITECH Act, 45 CFR Subtitle A, § 170.302

pants on the number and types of ambiguities identified for each paragraph in § 170.302.

When examining our results according to ambiguity type, the participants demonstrate fair agreement (ICC: 0.316, $p < 0.0001$). This indicates that participants successfully identified different ambiguity types according to our taxonomy classifications. Although there is clearly room for improvement, we believe these results are encouraging given the training, time, and conditions we were able to provide our participants. If we examine participant agreement regarding whether or not each paragraph was unambiguous, we find that participants demonstrate slight agreement (FK: 0.0446, $p = 0.00288$). Participants unanimously agreed that paragraph § 170.302(h) was ambiguous and every participant except one rated § 170.302(a) as ambiguous. For the other 21 paragraphs, participants exhibited little agreement.

Figure 2 shows which ambiguity types participants identified most. Each type (labeled on the x-axis) has two bars. The bar on the left (with hash marks) represents the number of ambiguities identified by technologists, and the one on the right (without hash marks) represents the number identified by policy analysts. Each bar is divided into two parts. The lower part (with a lighter shade) represents the proportion of the total that are unintentional ambiguities identified, and the upper part (with a darker shade) represents intentional ambiguities. For example, both technologists and policy makers identified roughly the same number of Syntactic ambiguities. In contrast, technologists and policy analysts differ in their identification of Incompleteness. Technologists identified over 100 Incompletenesses, with about a quarter of those being intentional, whereas policy analysts only identified about 50 Incompletenesses, most of which were unintentional.

The largest disagreement between technologists and policy analysts occurred in the Lexical and Incompleteness ambiguity types. Policy analysts found on average 4.4 times more lexical ambiguity than technologists, and technologists found 1.8 times more incompletenesses than policy analysts. This may be indicative of their respective professional training and background. Lexical ambiguities are more commonly associated with grammar, writing, and linguistics, whereas Incompleteness comes primarily from software engineering. Table III details additional examples of both agreement and disagreement. Note that the number of Vaguenesses identified differs greatly, which may also be a result of training because Vagueness and Incompleteness are similar, overlapping ambiguity types. Technologists are trained to identify Incompleteness and Vagueness in formal specifications, which may carry over into identifying those ambiguity types in § 170.302.

Q3: Do participants agree on the number and types of intentional ambiguities they identified in § 170.302?

Answer: We evaluated agreement using intraclass correlation and found a slight level of agreement between participants

TABLE III
AMBIGUITIES IDENTIFIED BY TYPE

Type	Intent	Lexical	Syntactic	Semantic	Vagueness	Incompleteness	Referential	Other
Tech	U	1.0, 1.9	2.9, 3.5	1.4, 2.2	5.6, 3.1	8.2, 5.5	7.3, 4.6	1.0, 1.2
	I	0.0, 0.0	0.0, 0.0	0.0, 0.0	3.0, 2.8	3.2, 5.7	0.0, 0.0	0.0, 0.0
	C	1.0, 1.9	2.9, 3.5	1.4, 2.2	8.6, 3.1	11.4, 10.3	7.3, 4.6	1.0, 1.2
Policy	U	3.9, 5.0	3.3, 3.3	1.6, 3.0	7.8, 8.0	6.1, 2.7	5.4, 4.7	0.6, 1.0
	I	0.5, 1.0	0.0, 0.0	0.1, 0.3	3.5, 4.2	0.1, 0.3	0.4, 1.0	0.0, 0.0
	C	4.4, 5.0	3.3, 3.3	1.8, 2.9	11.3, 9.0	6.3, 2.7	5.8, 4.4	0.6, 1.0
Combined	U	2.4, 4.0	3.1, 3.4	1.5, 2.6	6.6, 6.0	7.2, 4.5	6.4, 4.8	0.8, 1.1
	I	0.2, 0.7	0.0, 0.0	0.1, 0.2	3.2, 3.5	1.8, 4.5	0.2, 0.7	0.0, 0.0
	C	2.6, 4.1	3.1, 3.4	1.6, 2.6	9.8, 6.7	9.0, 8.2	6.6, 4.6	0.8, 1.1

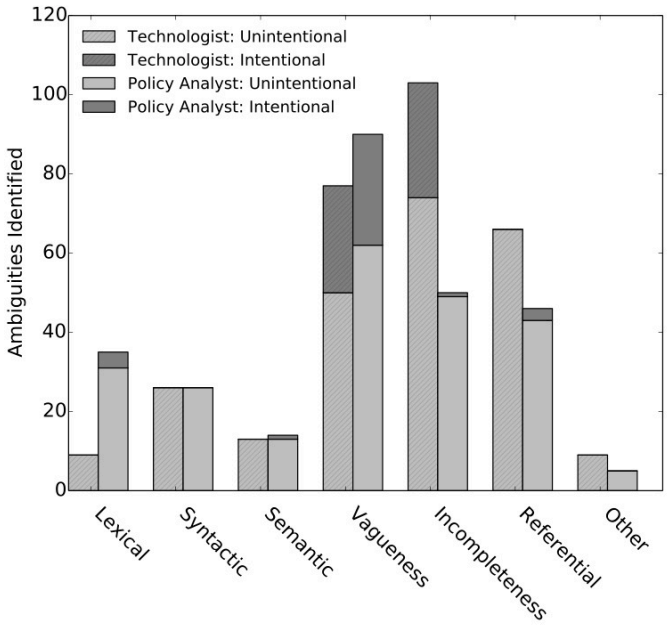

Fig. 2. Ambiguities Identified by Type

on the number and types of intentional ambiguities identified for each paragraph in § 170.302.

Participants agreed less on the number and type of intentional ambiguities than they did on the number and type of total ambiguities. Participants exhibited slight agreement on intentional ambiguities, whether measured by number (ICC: 0.141, $p < 0.0001$) or type (ICC: 0.201, $p < 0.0001$). The set of intentional Incompletenesses identified by the participants drove this difference. Table III shows that technologists identified an average of 3.2 intentional Incompletenesses compared to a 0.1 average for policy analysts. If we remove all incompletenesses from the calculation, the level of agreement for the number of identified ambiguities is roughly the same as before (ICC 0.134, $p < 0.0001$) and the level of agreement for the ambiguity type increases (ICC: 0.39, $p < 0.0001$).

Regardless of the agreement level, the fact that participants of both groups were able to identify intentional ambiguities at all is important because intentional ambiguity is a fundamental part of legal texts [1]. Intentional ambiguity holds important implications for software requirements that must comply with laws and regulations [1], [3], [6], [7], [10], [32]. Consider § 170.302(p), which reads as follows:

> (p) Emergency access. Permit authorized users (who are authorized for emergency situations) to access electronic health information during an emergency.

This paragraph describes the "break the glass" scenario in which physicians otherwise unable to access certain health records would be allowed access. The definition of an "emergency situation" or what it means to be "authorized for emergency situations" is not provided. Thirteen participants flagged this as an intentional ambiguity. This recognition is important because intentional ambiguities must first be identified before they may be disambiguated. Moreover, they must be periodically reevaluated regarding ambiguity and the author's intent.

Q4: Do participants agree on whether software engineers should be able to build software that complies with each paragraph in § 170.302?

Answer: We evaluated agreement using Fleiss' kappa and did not find agreement between participants on whether paragraphs from § 170.302 were implementable.

Participant agreement was not statistically significant for the group as a whole (FK: 0.0052, $p = 0.788$) or for the technologists as a group (0.0455, $p = 0.116$). The policy analysts disagreed slightly on the legal text's implementability (FK: -0.124, $p = 0.0111$). This is consistent with other findings for similar tasks involving the evaluation of legal texts for software engineering purposes. Prior research notes that determining whether software requirements have met or exceeded their legal obligations is challenging [7]. Maxwell found identification and classification of legal cross references to be similarly challenging for professional engineers [6].

Q5: Does an identified ambiguity affect whether participants believe that software engineers should be able to build software that complies with each paragraph in § 170.302?

Answer: Yes, 89% of unimplementable paragraphs contained an unintended ambiguity, whereas only 48% of implementable paragraphs contained an ambiguity.

Of the 83 paragraphs found to be unimplementable by the participants, 74 contained unintentional ambiguities. Of the 216 paragraphs found to be implementable, 104 contained unintentional ambiguities. We expected that those paragraphs that Participants identified as implementable would only rarely

contain unintentional ambiguities, but our results indicate that 48% of the implementable paragraphs were deemed to contain unintended ambiguities. Prior research has shown for a similar task (identifying legally implementation-ready requirements) that individuals working alone tend to be too liberal (i.e. accept as implementation-ready requirements that need further refinement) and groups working together tend to be too conservative (i.e. reject requirements that have actually met their legal obligations) [7], [33].

V. Discussion

Perhaps the most interesting results for this case study are the qualitative results. When preparing the materials for this case study, we examined § 170.302 many times to identify and classify its ambiguities. Our participants were given 50 minutes to accomplish the same task, yet they found several subtle ambiguities that eluded us. For example, consider § 170.302(q):

```
(q) Automatic log-off. Terminate an electronic
session after a predetermined time of inactivity.
```

We found the phrase "predetermined time of inactivity" to be incomplete or perhaps vague because no purpose is stated; making it a challenge to determine how much inactivity is allowable. It could be considered vague because "after" admits borderline or relative cases: does it mean immediately after or at some point after? Most participants identified the statement in the same way, but one participant identified "time" as lexically ambiguous. It could mean either duration or a time of day. If interpreted as the latter, then it could be interpreted as requiring EHRs to terminate an electronic session after closing time.

Paragraph § 170.302(o) provides another interesting example:

```
(o) Access control. Assign a unique name and/or
number for identifying and tracking user identity
and establish controls that permit only authorized
users to access electronic health information.
```

Two respondents found this paragraph to be both unambiguous and also not implementable. How could an unambiguous statement be unimplementable? This may seem unintuitive at first, but the halting problem can be stated unambiguously and cannot be implemented. Similarly, the absolute nature of the phrase "permit only authorized users to access electronic health information" could be interpreted as impossible to implement because it is not a wholly technological problem.

VI. Threats to Validity

Case study research is incomplete without a discussion of concerns that may threaten results validity. *Internal validity* refers to the causal inferences made based on experimental data [34]. Herein, we do not attempt to determine causality for any part of this research. Our goal is simply to determine whether and how participants identify and classify ambiguity in legal texts.

Construct validity refers to the appropriate use of evaluation metrics and measures [34]. We specifically avoided the use of absolute measures of ambiguity to conform with the term as expressed in accepted IEEE standards [12]. To calculate other statistical measures, we used accepted statistics for agreement (ICC and Fleiss' kappa) and scrupulously followed recommended practices in applying them. Our case study participants may have become fatigued and stopped responding to the questions in our survey. To mitigate the impact of survey fatigue, we adjusted our statistical measures to account for the three surveys that contained unanswered questions.

Providing participants with only a single section of the HITECH Act and the text of cross-references contained within it is another threat to construct validity. The complete text would have unreasonably increased participant fatigue. Note that providing additional text could allow participants to either disambiguate ambiguities identified in our study or discover additional ambiguities resulting from potentially conflicting material.

External validity refers to the ability to generalize the findings to other domains [34]. We have mitigated threats to external validity by selecting a section in the HITECH Act that is representative of the style, tone, and wording of obligations found in the rest of the act. In addition, we chose a participant population with as many different backgrounds as possible rather than limiting our research to stakeholders with an engineering background. Unfortunately, two important threats of this type remain. First, our study employs a small population of students rather than a large population of practitioners. Although the findings of our study align with similar case studies that examine legal texts for engineering purposes [6], [7], students enrolled in a graduate class may not be representative of practicing engineers, lawyers, managers, and policy makers. Second, we selected a legal text from a single domain. Healthcare is a popular domain for regulatory compliance software engineering research, but other domains, like finance, also have extensive regulatory requirements. To address these threats, we plan to adapt what we have learned from this study for a broader, web-based examination of ambiguity identification and classification for multiple legal domains in the future.

Reliability refers to the ability of other researchers to replicate this methodology. We assiduously detailed both our methods and our evaluation techniques. In addition, we have made our case study tutorial and survey materials available online for researchers interested in replicating our results.[11] We do not believe reliability is a serious concern for this research.

VII. Summary and Future Work

The development of methods to improve and demonstrate legal compliance with federal privacy and security regulations in software systems is critical. Stakeholders of regulated software systems, and in particular requirements engineers, must be able to identify ambiguities in legal text and understand their implications for software systems. To this end, we created a taxonomy with six ambiguity types intended to encompass a broad definition of ambiguity within the context of legal texts. We conducted a case study to examine how students

[11] http://www.cc.gatech.edu/~akmassey/documents/ambiguity-case-study-materials.pdf

in a graduate privacy class identify and classify ambiguity for § 170.302 in the HITECH Act. Our research suggests that ambiguity is prevalent in legal texts. In 50 minutes of examination, participants in our case study identified on average 33.47 ambiguities in 104 lines of legal text using our ambiguity taxonomy as a guide.

Participants did not exhibit strong agreement on the number and type of ambiguities present in the legal text. This may be due to the 50-minute time limit or to the complexity of the task. Our analysis suggests (a) that participants used the taxonomy as intended: as a guide and (b) that the taxonomy provides adequate coverage (97.5%) of the ambiguities found in §170.302. This suggests that the ambiguity taxonomy is sufficient for analyzing this particular legal text.

Participants were willing to accept paragraphs with unintentional ambiguities as implementable (i.e. as something for which software engineers should be able to build compliant software). Prior research has shown that software engineers are ill-equipped to perform similar tasks [6], [7]. Further research is needed in this area to provide better guidance and improve decision-making in this area.

We plan to conduct additional case studies on larger populations to better understand ambiguity in legal texts and its implications for software engineering. In particular, we seek to conduct a larger online case study covering healthcare, finance, and other regulated domains. A larger study would allow us to evaluate multiple possible aids for identifying and classifying ambiguity in legal text. In addition, we plan to examine whether identifying and classifying ambiguity improves software engineering assessments of legal implementation readiness, which our prior work has shown to be extremely challenging for engineers to do with accuracy [7].

REFERENCES

[1] P. N. Otto and A. I. Antón, "Addressing Legal Requirements in Requirements Engineering," *15th IEEE International Requirements Engineering Conference*, pp. 5–14, 15-19 Oct. 2007.

[2] G. Roman, "A taxonomy of current issues in requirements engineering," *Computer*, vol. 18, no. 4, pp. 14–23, Apr. 1985.

[3] A. K. Massey, P. N. Otto, L. J. Hayward, and A. I. Antón, "Evaluating Existing Security and Privacy Requirements for Legal Compliance," *Requirements Engineering*, 2010.

[4] A. K. Massey, P. N. Otto, and A. I. Antón, "Legal Requirements Prioritization," *Proc. of the 2nd Intl. IEEE Workshop on Requirements Engineering and the Law*, 2009.

[5] Department of Health and Human Services, "Medicare and Medicaid Programs; Electronic Health Record Incentive Program; Final Rule," *Federal Register*, vol. 75, no. 8, July 28 2010.

[6] J. C. Maxwell, "Reasoning about legal text evolution for regulatory compliance in software systems," Ph.D. dissertation, North Carolina State University, 2013.

[7] A. Massey, B. Smith, P. Otto, and A. Anton, "Assessing the Accuracy of Legal Implementation Readiness Decisions," in *19th IEEE International Requirements Engineering Conference (RE)*, September 2011, pp. 207–216.

[8] A. K. Massey, J. Eisenstein, A. I. Antón, and P. Swire, "Automated Text Mining for Requirements Analysis of Policy Documents," *21st International Conference on Requirements Engineering*, 2013.

[9] D. M. Berry and E. Kamsties, "Ambiguity in requirements specification," in *Perspectives on Software Requirements*, ser. The Springer International Series in Engineering and Computer Science, P. Leite, J. C. Sampaio, and J. H. Doorn, Eds. Springer US, 2004, vol. 753, pp. 7–44.

[10] D. M. Berry, E. Kamsties, and M. M. Krieger, "From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity," School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, Tech. Rep., November 2003.

[11] F. Chantree, B. Nuseibeh, A. De Roeck, and A. Willis, "Identifying nocuous ambiguities in natural language requirements," in *Requirements Engineering, 14th IEEE International Conference*, 2006, pp. 59–68.

[12] IEEE, *ANSI/IEEE Standard 830-1993: Recommended Practice for Software Requirements Specifications*. Institute of Electrical and Electronics Engineering, New York, NY, 1993.

[13] S. Easterbrook and B. Nuseibeh, "Managing inconsistencies in an evolving specification," *Second International Symposium on Requirements Engineering*, 1995.

[14] H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Extending nocuous ambiguity analysis for anaphora in natural language requirements," in *18th IEEE International Requirements Engineering Conference*, 2010, pp. 25–34.

[15] D. Gordon and T. Breaux, "A cross-domain empirical study and legal evaluation of the requirements water marking method," *Requirements Engineering*, vol. 18, no. 2, pp. 147–173, 2013.

[16] M. Osborne and C. MacNish, "Processing natural language software requirement specifications," in *Requirements Engineering, 1996., Proceedings of the Second International Conference on*, 1996, pp. 229–236.

[17] A. Umber and I. Bajwa, "Minimizing ambiguity in natural language software requirements specification," in *Sixth International Conference on Digital Information Management*, 2011, pp. 102–107.

[18] A. Nigam, N. Arya, B. Nigam, and D. Jain, "Tool for Automatic Discovery of Ambiguity in Requirements," *International Journal of Computer Science*, vol. 9, no. 5, September 2012.

[19] D. van Bussel, "Detecting ambiguity in requirements specifications," Master's thesis, Tilburg University, August 2009.

[20] D. Popescu, S. Rugaber, N. Medvidovic, and D. Berry, "Reducing ambiguities in requirements specifications via automatically created object-oriented models," in *Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs*, ser. Lecture Notes in Computer Science, B. Paech and C. Martell, Eds. Springer Berlin Heidelberg, 2008, vol. 5320, pp. 103–124.

[21] A. I. Antón, J. B. Earp, and R. A. Carter, "Precluding incongruous behavior by aligning software requirements with security and privacy policies," *Information and Software Technology*, vol. 45, no. 14, pp. 967–977, 2003.

[22] A. Sennet, "Ambiguity," in *The Stanford Encyclopedia of Philosophy*, summer 2011 ed., E. N. Zalta, Ed., 2011.

[23] W. Empson, *Seven Types of Ambiguity*. New Directions, 1966.

[24] T. Wasow, A. Perfors, and D. Beaver, *Morphology and The Web of Grammar: Essays in Memory of Steven G. Lapointe*. CSLI Publications, 2005, ch. The Puzzle of Ambiguity.

[25] T. Wasow, "Ambiguity avoidance is overrated," *to appear in a volume edited by Susanne Winkler, http://www.stanford.edu/~wasow/Ambiguity.pdf*, 2014.

[26] D. F. Wallace, "Tense present," *Harper's Magazine*, April 2001.

[27] V. Basili and H. Rombach, "The tame project: towards improvement-oriented software environments," *IEEE Transactions on Software Engineering*, vol. 14, pp. 758–773, 1988.

[28] V. Basili, G. Caldiera, and D. Rombach, "The Goal Question Metric Approach," *Encyclopedia of Software Engineering*, vol. 2, pp. 528–532, 1994.

[29] R. V. Solingen and E. Berghout, *The Goal/Question/Metric Method*. McGraw-Hill Education, 1999.

[30] K. O. McGraw and S. P. Wong, "Forming inferences about some intraclass correlation coefficients." *Psychological Methods*, vol. 1, pp. 30–46, 1996.

[31] J. L. Fleiss and J. Cohen, "The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability," *Educational and Psychological Measurement*, vol. 33, pp. 613–619, 1973.

[32] J. C. Maxwell and A. I. Antón, "Discovering Conflicting Software Requirements by Analyzing Legal Cross-References," *(In Submission) IEEE International Requirements Engineering Conference*, 2010.

[33] A. K. Massey, "Legal requirements metrics for compliance analysis," Ph.D. dissertation, North Carolina State University, 2012.

[34] R. K. Yin, *Case Study Research: Design and Methods*, 3rd ed., ser. Applied Social Research Methods Series, L. Bickman and D. J. Rog, Eds. Sage Publications, 2003, vol. 5.