North Carolina State University Computer Science Technical Report TR-2007-26

Managing Ambiguity and Traceability in Regulatory Requirements: A Tool-supported Frame-based Approach

Travis D. Breaux and Annie I. Antón
Department of Computer Science
North Carolina State University
{tdbreaux,aianton}@ncsu.edu

Abstract. *Government laws and regulations impose requirements on software-intensive information systems. To comply, organizations need to evaluate current and future software systems early in the software development and procurement process by using a set of regulatory requirements. Acquiring requirements from regulations is complex because regulations contain ambiguity and because maintaining traceability is essential to demonstrate due diligence in adhering to the law. To further address the traceability challenge, we extend the Frame-Based Requirements Analysis Method (FBRAM); a method to systematically acquire requirements specifications from regulations. This extension allows analysts to derive dependencies from cross-references and maintain these dependencies in a requirements model. This tool-supported method constructs the requirements model from an annotated regulation text and generates a requirements document, expressed in HTML, from the model. This document is visually inspected by analysts and domain experts who determine the correctness of the resulting requirements based on their collective interpretations of the regulation.*

1. Introduction

National and international standards, regulations and policies impose requirements on industries and business practices that affect a software system's non-functional properties (e.g., accessibility, privacy, safety, security, etc.). Because these requirements are broadly written to govern one or more industries and are not restricted to a single system with a well-defined set of stakeholders, analysts must strategically address the ambiguous syntax in regulatory language while maintaining traceability from regulations to requirements.

The Frame-Based Requirements Analysis Method (FBRAM) assists analysts in extracting software requirements from regulations [7]. It has been applied to two regulations: the Health Insurance Portability and Accountability Act[1] (HIPAA) Privacy Rule and the Telecommunications Act[2] of 1996, Section 508. In this paper, we extend the FBRAM and illustrate how it is applied within the context of the HIPAA Privacy Rule. The HIPAA Privacy Rule governs the privacy of patient medical information and affects some 545,000 different establishments in the U.S. who employ over 13.5 million people [9].

This paper is organized as follows: Section 2 summarizes related work; Section 3 discusses several challenges to resolving ambiguity and improving traceability in regulatory requirements; Section 4 presents the extended FBRAM; Section 5 presents the results of a HIPAA case study; Section 6 presents a comparative evaluation with another method; and Section 7 concludes with a discussion and summary.

2. Related Work

Zave and Jackson discuss the challenge in acquiring formal specifications of systems from informal descriptions of the environment [27]. They declare that "all statements in the course of requirements engineering are statements about the environment" [27] and they distinguish statements expressed in the *indicative* mood, which includes definitions and facts, from statements expressed in the *optative* mood, which typically include requirements [27]. The FBRAM's requirements meta-model includes both types of statements [7]; optative statements are further refined into normative statements about rights, permissions, obligations and refrainments. *Rights* and *permissions* describe actions that stakeholders are permitted to perform, whereas *obligations* and *refrainments* describe actions that stakeholders are required to perform or required to avoid, respectively. These concepts have been formalized in Deontic Logic [19].

Prior work in requirements modeling includes goal-oriented methods and models, such as KAOS [12], Tropos [15, 18], GBRAM [1] and User Requirements Notation (URN) [16]. *Goals* represent states that a system must achieve, maintain or avoid. *Normative goals* elaborate upon the traditional goal concept by

[1] U.S. Public Law No. 104-191, 110 Stat. 1936 (1996)

[2] U.S. Public Law No. 104-104, 110 Stat. 56 (1996)

expressing goals as permissions, obligations and refrainments [2, 18]. Herein, we adapt a frame-based representation to normative goals in which a *frame* corresponds to a concept, comprising *slots* that represent stereotypical properties of that concept [14, 25]. Frames formalize the deep structure, or semantics, of regulatory language [10]. The FBRAM is a step forward in automating an earlier, entirely manual methodology to acquire normative goals from regulations [4, 6].

Lexicons and natural language (NL) patterns are used to analyze requirements. Wasson et al. [26], Overmeyer et al. [23] and Cysneiros and Leite [11] employ lexicons to improve natural language (NL) requirements analysis. In addition, NL patterns are used to identify critical real-time properties [21] and improve requirements quality for embedded systems [13]. FBRAM complements this work by providing a means to formalize NL patterns using a standard lexicon or ontology. Lee et al. describe an ontological approach to acquire requirements from regulations [22]. They highlight a need for new methods to systematically decompose verbose regulatory statements into requirements; a contribution of the FBRAM, presented herein.

The Requirements Apprentice (RA) tool acquires formal specifications from informal descriptions using a specification language based upon Common Lisp and *clichés*, which are used to codify requirements knowledge in frame-like structures [24]. Analysts employ the RA to construct a requirements model through requirements elicitation with domain experts. The FBRAM is specifically designed to systematically account for how analysts interpret and manipulate legal language in regulatory documents during requirements acquisition. Similar to the RA, domain expert feedback is required to evaluate the correctness of an analyst's formalized interpretation of a regulation.

Ghanavati et al. describe a goal-oriented requirements framework that they applied to the Canadian Personal Health Information Privacy Act (PHIPA) [16]. The framework expresses goal models in the Goal-oriented Requirements Language (GRL) and maintains "source links" from regulatory documents to goals acquired from those documents. They note that our manual method [4] "could facilitate the extraction of [their] privacy GRL model" from the PHIPA [16]. The FBRAM improves upon the manual method by automating the maintenance of traceability links from goals to their originating sections, paragraphs, statements and phrases. Furthermore, the frame-based requirements generated by the FBRAM can be expressed in GRL to provide inputs directly to their framework.

3. Ambiguity and Traceability

This section discusses the two primary challenges faced by analysts who extract requirements from regulatory texts: ambiguity and traceability.

Ambiguity. U.S. Federal and state regulations contain ambiguities that are intended by law makers to be re-interpreted as business practices emerge and as capabilities to comply with regulations change over time. For example, HIPAA §164.512(e)(1)(iv) states that an entity must make "reasonable" efforts to notify individuals of certain requests for their protected health information. The word "reasonable" is an intended ambiguity: which mechanisms are considered *reasonable*, (e.g., postal mail, secure electronic mail or websites, etc.) varies depending on the type of communities served and the prevalence of relevant, existing technologies.

Law makers also define governed entities using terms that are open to interpretation. For example, in HIPAA §164.304, *workstation* is exemplified by "a laptop or desktop computer, or any other device that performs similar functions." Compliance officers must decide if this definition is intended to cover handheld Personal Digital Assistants (PDAs). As PDAs are integrated into routine business practices, organizations may need to re-interpret this ambiguity to achieve compliance.

Regulations also contain unintended ambiguities that are inherent to natural language syntax — or English. We distinguish (and address) three types of ambiguity: logical, attributive, and referential. Kamsties classifies these ambiguities as *requirements document ambiguity* [20].

Logical ambiguity refers to English words that can be mapped to different logical interpretations. Herein, we only consider how English conjunctions (and, or) can be assigned conflicting logical connectives; see Berry and Kamsties for a separate discussion of universal and existential qualification-related ambiguity [8]. For example, in HIPAA §164.524(a)(1), an individual has "a right of access to inspect and obtain" a copy of their protected health information. While this statement uses the English conjunction "and," presumably an individual can obtain a copy of their information without needing to inspect the information; e.g., the conjunction can be interpreted as a logical-or. In contrast, interpreting this conjunction as a logical-and may lead to systems that provide the information such that an inspection is required and confirmable.

Attributive ambiguity is found in phrases that may be reasonably ascribed to more than one phrase within a sentence. For example, in HIPAA §164.520(b)(1)(vii), "The [privacy] notice must contain the name or title and telephone number of a person or office" may be construed to mean the notice contains one of: (1) the name of the person or office; (2) the title and telephone number of the person or office; or (3) the name and telephone number of the person or office. Because the phrase "and telephone number" can be attributed to the "name and title" or only the "title," the analyst may interpret either options (1) and (2) or options (2) and (3) as valid interpretations. The former interpretation permits the organization to withhold the telephone number from the policy, making it more difficult for recipients of the notice to contact the person or office.

Referential ambiguity occurs when a word or phrase has multiple meanings; this includes intensional and extensional polysemy [5]. We consider a type of extensional polysemy in which words have an anaphoric (backward-referencing) or cataphoric (forward-referencing) function. These words include pronouns (this, that, they), noun phrases that use definite articles (the) and some adjectives (such). A statement that contains the phrase "must provide such notices" refers to notices that are elaborated upon in the broader context of the statement or paragraph. The analyst must identify additional implications or constraints on the "notices," that appear in the broader context before determining which notices must be provided.

Traceability. Regulations present traditional and novel traceability challenges to analysts. Similar to other requirements sources (e.g. interviews, scenarios and use cases), the loss of original context also affects requirements that are extracted from regulations. Unlike these other sources, the "context" of a regulatory statement is distributed across multiple sections, paragraphs and sub-paragraphs in the source document. Analysts must reconstruct this context by employing knowledge of the regulation document structure and the cross-reference syntax. For example, a regulatory statement can start in one paragraph and end in a sub-paragraph; this break is called a *continuation*. Consider the following continuation in HIPAA §164.520(a)(2)(i)(B)(ii) that describes two requirements (obligations) to maintain and provide a privacy notice to patients:

```
(ii) A group health plan… must:
    (A) Maintain a notice under this
        section; and
    (B) Provide such notice to any person…
```

During requirements acquisition, traceability must be maintained between unique paragraph indices and corresponding requirements to map paragraph cross-references back to those requirements [4]. Therefore, the paragraph index (ii) should trace to both requirements (and vice versa), whereas the paragraph index (ii)(A) should only trace to the maintenance requirement and paragraph index (ii)(B) should only trace to the provision requirement.

Regulations also contain *internal cross-references* to sections and paragraphs within the regulation and *external cross-references* to other regulations. These cross-references express dependencies between regulatory rules. Consider the following statement (a fact) from HIPAA §164.522(a)(1)(v) that describes an exception; the cross-reference phrases appear in **bold**:

> `A restriction agreed to by a covered entity` **`under paragraph (a) of this section`**`, is not effective under this subpart to prevent uses or disclosures` **`permitted or required under §164.502(a)(2)(i), 164.510(a) or 164.512`**`.`

The fact contains four cross-references: a *relative* reference to paragraph (a) in §164.522, and three *fully qualified* references to other Privacy Rule sections and paragraphs. From each cross-reference, we can derive an asymmetric dependency between the fact and one or more other statements in the cross-referenced paragraphs. For example, the first cross-reference excludes the effect of "agreed restrictions" (the domain) from certain uses and disclosures stated by this fact (the range). In contrast, the remaining three cross-references refine these uses and disclosures (the domain) through this exception (the range). When analysts extract requirements from these paragraphs, they must propagate these derived dependencies to requirements specifications derived from those cross-referenced paragraphs to preserve the original context.

4. Frame-Based Requirements Analysis

In the Frame-Based Requirements Analysis Method (FBRAM), analysts use specialized knowledge about requirements and law to manually annotate a regulatory document and an accompanying tool parses the annotations to extract regulatory requirements (Fig. 1).

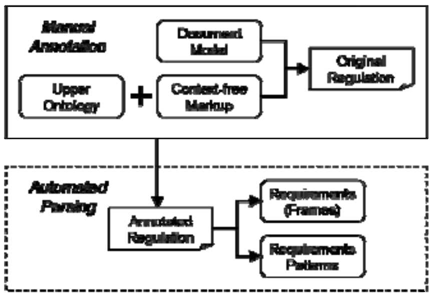

Figure 1: Overview of the FBRAM

The manual annotation process uses the following three artifacts:

1. A reusable *upper ontology* comprised of domain-independent requirements concepts and properties, used to classify regulatory statements independently of any single regulatory domain (e.g., privacy, accessibility).

2. A *context-free markup* language that describes the deep structure of natural language [9] using concepts in the upper ontology and logical connectives.
3. A *document model* that describes the structural organization of a regulatory document in terms of hierarchical divisions (e.g., sections, paragraphs, sub-paragraphs, etc.)

During the manual annotation process (top of Fig. 1), analysts use upper ontology concepts and the context-free markup language to assign an interpretation to a regulation text; this can remove logical, attributive and referential ambiguity. This manual process yields an annotated regulation that is then parsed by our tool to produce the following two types of artifacts:

1. A r*equirement* that is represented as a frame in which original, unedited phrases from the regulation text are assigned to slots in the frame and cross-references are formalized into typed dependencies between requirements.
2. For each requirement, a *requirement pattern* is generalized from the requirement's originating natural language syntax in the regulation.

During parsing, the tool identifies and reports syntax and semantic errors in the markup. The parsed frame objects and patterns are converted into a W3C eXtensible Markup Language (XML) representation; a process called *serialization*. The serialized objects can then be manipulated using the eXtensible Stylesheet Language Transformations (XSLT). To date, from the annotated text, we generate a requirements document expressed in the Hypertext Markup Language (HTML) and an exception and refinement hierarchy expressed in the Graph Markup Language (GraphML). These artifacts help analysts validate whether the semantics of the applied annotations match their intended interpretation of the regulations.

4.1. The Upper Ontology

The upper ontology or *meta-model* [12] describes domain-independent knowledge about the semantic structure of regulatory requirements. Fig. 2 presents the upper ontology using the Unified Modeling Language (UML); this ontology has been validated in two case studies in the accessibility and privacy domains. The concepts in the upper ontology are connected by two types of arrows: arrows that terminate with dark triangles and lead from sub-classes to super-classes; and arrows that terminate with white diamonds and lead from properties to classes, containing those properties. There are three types of concepts in the upper ontology:

1. *Statement-level concepts* (represented by boxes with bold-line borders) classify individual regulatory statements;
2. *Phrase-level concepts* (represented by grayed boxes) classify individual phrases in a regulatory statement; and
3. *Abstract placeholder concepts* (represented by boxes with dotted-line borders) classify statement and phrase-level concepts for analysts.

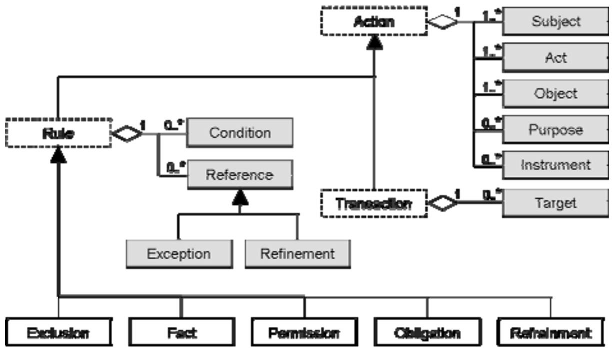

Figure 2: Regulatory Requirements Upper Ontology

The statement-level concepts are defined below. Note that an *entity* is any stakeholder, system or component, including software or hardware:

- *Exclusion* means any state that an entity is not required to achieve, maintain or avoid or any act that an entity is not required to perform.
- *Fact* means any state or act that is assumed true.
- *Permission* means any state that an entity is permitted to achieve, maintain or avoid or any act that an entity is permitted to perform; permissions include stakeholder *rights* [4].
- *Obligation* means any state that an entity is required to achieve or maintain or any act that an entity is required to perform.
- *Refrainment* means any state that an entity is required to avoid or any act that an entity is required to not perform.

The phrase-level concepts are defined as follows:

- *Subject* is the entity that performs an action.
- *Act* is the act performed by an entity.
- *Object* is the object on which an action is performed by an entity.
- *Purpose* is the purpose for which, or why, an action is performed by an entity.
- *Instrument* is the method by which, or how, an action is performed by an entity.
- *Target* is the recipient in a transaction.
- *Condition* is the pre-condition(s) that must be true before an entity acts.

- *Reference* is a phrase that describes rules in another section or paragraph.
- *Exception* is a reference that states that the rule has an exception (another rule).
  - *Inverse exception* is a reference that states one rule is an exception to another rule.
- *Refinement* is a reference that states that the rule has a refinement (another rule).
  - *Inverse refinement* is a reference that states one rule is a refinement to another rule.

The concepts in the upper ontology have been acquired across multiple case studies that include an analysis of privacy policies [2], a HIPAA consumer fact sheet [3] and the HIPAA Privacy Rule [4, 6]. The upper ontology has been formalized in Description Logic for the purpose of reasoning about and comparing normative goals using subsumption [5].

4.2. The Document Model

The document model enables forward and reverse-mapping between requirements and the indices of sections, paragraphs and sub-paragraphs in the regulation that contain the originating statements for these requirements. The indices are used in cross-references that appear in reference phrases, which can be formalized as exceptions [6] or refinements to requirements. Regulatory statements may begin in one paragraph and end in a sub-paragraph. Usually, these continuations presents a set of shared constraints (e.g., subjects, actions, conditions, etc.) in the leading paragraph followed by alternative permissions, obligations and refrainments in sub-paragraphs that reuse the shared constraints. Consider the division syntax in HIPAA Privacy Rule excerpt §164.520(a)(2)(i)(B)(ii); it describes two obligations to notify patients of their privacy practices and shares the same subject constraint (a group health plan) for these obligations:

```
(ii) A group health plan… must:
    (A) Maintain a notice under this
        section; and
    (B) Provide such notice to any person…
```

To support traceability, the document model formalizes the divisions within the regulatory text. The document model semantics are expressed in the W3C eXtensible Schema Language (XSL). The analyst applies the model to the regulation text by replacing division headers with an XML `<div>` tag that maps the header index and sub-title, if any, to corresponding attributes *index* and *title* in the tag; the analyst adds the XML `</div>` tag at end of the division. The XSL is used to debug syntax errors when applying the document model. The above excerpt appears in Fig. 3, annotated with the document model.

```
<document>
   <!-- 164.520(a)(2)(i)(B) -->
     ...
     <div index="(ii)">
        A group health…, must:
        <div index="(A)">
           Maintain a notice under this
           section; and
        </div>
        <div index="(B)">
           Provide such notice to any person…
        </div>
        ...
     </div><!-- end of (ii) -->
</document>
```

Figure 3: The Document Model Applied to the HIPAA §164.520(a)(2)(i)(B)(ii) Excerpt

Because a regulation text's indentation and font styles may be lost or corrupted when the text is transferred to a plain text format, the analyst manually applies the document model to the regulation plain text. However, once applied, the supporting tools will generate an HTML representation that is an indented and stylized version of the original regulation text.

4.3. The Context-free Markup

Analysts use the context-free markup language to codify their interpretation of a regulation text. To codify this interpretation, analysts must align concepts from the upper ontology with regulation sentences and phrases, removing logical, attributive and referential ambiguities and formalizing cross-references as dependencies between requirements. The extended context-free grammar for the markup appears in Appendix A. Table 1 presents concept codes employed in the markup example below to align sentences and phrases with concepts in the upper ontology.

Table 1: Codes Corresponding to Upper Ontology Concepts

Code	Concept	Code	Concept
a	Act	o	Object
c	Condition	R	Refinement
F	Fact	s	Subject
O	Obligation	t	Target

The running example from HIPAA Privacy Rule §164.520(a)(2)(i)(B)(ii) appears below with the markup in **bold**:

```
1 (ii) {#O [#s/1 A group health plan [that
2    provides health benefits solely through
3    an insurance contract with a health
4    insurance issuer or HMO, & and that
5    creates or receives [protected health
6    information in addition to [summary
7    health information [!R164.504/(a) as
8    defined in §164.504(a)]] | or information
9    on whether the individual is
10   participating in the group health
11   plan, or is enrolled in or has
12   disenrolled from a health insurance
13   issuer or HMO offered by the plan]]],
14   {\2 must}:
15   (A) {{#a {*2} [Maintain]} [#o/3 a notice
16       under this section]; & and
17   (B) {#a {*2} [Provide]} [#o*3 such
18       notice] {#c upon [request]} {#t to
19       [any person]}}}. {#F [#s The
20       provisions of paragraph (c)(1) of
21       this section] {#a do not [apply]}
22       {#o to [*1 such group health plan]}}.
```

The markup structures regulatory text into two types of nested blocks denoted by opening and closing brackets: *pattern blocks*, denoted by curly "{}" brackets, indicate the start of a requirements pattern or sub-pattern; and *value blocks*, denoted by square "[ ]" brackets, indicate spans of text that will be mapped to slot values by the parser. A block is *typed* if the opening bracket is followed by a number sign "#" and a letter. Within a block, the English conjunctions "and" and "or" can be mapped to logical connectives using the operators "&" and "|" for logical-and and logical-or, respectively (see lines 4, 8, 16).

To resolve attributive and referential context-sensitive ambiguities, the analyst uses the copy "/" operator, cut "\" operator and "*" paste operator followed by a numbered clipboard location. Recall that referential ambiguity includes words that have an anaphoric or cataphoric function. The phrases "such notice" (lines 17-18) and "such group health plan" (line 22) introduce referential ambiguity. To preserve this original context, we replace these referential ambiguities with the phrases to which they refer. If the paste operator is applied to a block that contains text, as is the case in this example, the text in the block will be replaced by the pasted text.

To formalize cross-references, analysts demarcate cross-reference phrases by using a value sub-block that begins with a cross-reference "!" operator and is followed by a reference code and path (see line 7). The reference code corresponds to a reference concept in the upper ontology and the path is a forward-slash delimited list of section and paragraph indices that appear in the document model. The tool parses the reference path, identifies the rules that are located in the referenced document divisions, and creates dependencies from the rule that contains the reference to the rules referenced by the path. For example, the reference on line 7 denotes a dependency from both obligations that start in paragraph (ii) and continue into sub-paragraphs (A) and (B). The code "R" defines this dependency as a refinement relation and the path "164.504/(a)" identifies the division as paragraph (a) of section 164.504 that contains the target rules. The phrase "summary health information" is the phrase refined by this dependency.

Some cross-references are *relative* to their immediately encapsulating paragraph. For example, a reference "(a)" in paragraph (a)(1) refers to the parent paragraph of sub-paragraph (1); whereas a reference "(b)" in paragraph (a) refers to the sibling paragraph (b). While analysts may supply a fully qualify reference when stating the path from a relative reference, they may also let the parser attempt to systematically resolve these relative references, directly. The parser detects syntax and semantic errors, such as missing brackets, cycles that occur in the copy/ cut/ paste operations, unknown concept codes, dangling (possibly external) cross-references, etc., and alerts the analyst who must then resolve these errors.

4.4. Requirements

Parsing the annotated regulation text yields requirement specifications that are formalized as frame objects. Requirements knowledge structured by these frames is serialized using XML and transformed into a requirements document and exception/ refinement graph using XSLT. The requirements document is expressed in HTML and contains specifications that are presented in a table format. Parsing the example markup from Section 4.3 yields two requirements due to case splitting [2]; the second requirement is presented in Fig. 4 using the same table format that is used in practice.

Frame Type: Obligation	
Pattern: [*subject*] {must [*act*]} [*object*] {upon [*condition*]} {to [*target*]}	
Trace: ID 5, Line 1:0, Source: 164.520(a)(2)(i)(B)(ii)	
Slots	**Values**
condition	*upon…* request
subject	A group health plan that provides health benefits solely through an insurance contract with a health insurance issuer or HMO *A group health plan that creates or receives* protected health information in addition to summary health information (**Refinement**: see §164.504(a)) *A group health plan that creates or receives* information on whether the individual is participating in the group health plan, or is enrolled in or has dis-enrolled from a health insurance issuer or HMO offered by the plan
act	*must…* Provide
object	a notice under this section
target	*to…* any person

Figure 4: Example HTML Table Generated from Annotated Regulation

The table in Figure 4 begins with the statement frame type (Frame Type), the requirements pattern (Pattern), the traceability information (Trace) that contains the requirement ID, the line number and line index and the corresponding paragraph number in the regulation text. Each slot is listed with the slot type (a phrase-level concept from the upper ontology) and the slot value. Because the slot values may be expressed using logical connectives (e.g. see the subject slot value in Fig. 4), the values are presented as trees comprised of logical-and branches (solid line) and logical-or branches (dotted line). Dependencies derived from internal cross-references are presented as hypertext links (underlined) to the corresponding rules that appear in the referenced paragraphs; the link is appended to the phrases from which the cross-reference originated. If the dependency is derived from an external cross-reference, the text description of the section or paragraph is appended in place of the hypertext link.

5. Case Study

The FBRAM was applied to four sections of the HIPAA Privacy Rule §164.520–§164.526 that govern privacy notices, individual rights to request access, access restrictions and amendments to protected health information. We chose these four sections for our case study because they describe publicly visible, consumer-related activities and to compare the FBRAM results with other results that were manually-acquired from these same sections [4]. We discuss the comparative evaluation in Section 6.

The first author applied the FBRAM to the aforementioned sections of the HIPAA Privacy Rule to yield 146 requirements statements. These statements consist of 34 permissions, 100 obligations, 3 refrainments, 2 exclusions and 7 facts. In addition, this study yielded 76 requirements patterns, similar to the pattern in Fig. 4, from the 146 statements; 52% of the acquired statements use only seven of the 76 patterns.

5.1. Ambiguities

The HIPAA study identified logical, attributive and referential ambiguities as discussed in Section 3. Because this study lacks a sufficient number of participants to evaluate alternate interpretations of logical and attributive ambiguities, we limit this discussion to referential ambiguity. In this study, we found it was not always necessary to resolve each referential ambiguity. For example, a referential ambiguity that forward or backward-references a phrase within the same statement (see "such group health plans" in Section 4.2) is less likely to be confusing than a referential ambiguity that refers to a phrase in a different statement. This is because statements are often parsed into separate requirements. Consider the following obligation from HIPAA §164.524(d)(4):

```
1   {#O [#s The covered entity] [#m must]
2   [#a promptly refer] [#o a request for
3   review] {#t to [*12 such designated
4   reviewing official]}}
```

The referential ambiguity "such designated reviewing official" (lines 3-4) refers to an official who is described in another obligation, earlier in paragraph (d)(4). Because this obligation stands alone, this reference is confusing. To preserve original context, we extended the parsing tool to create a refinement dependency from the prior obligation, from which the source text is copied, to the target obligation on lines 1-4 to which the text is pasted. This refinement dependency is similar to explicit cross-references that use phrases similar to "as defined in §164.524;" however, in this case, the act of the analyst copying text between statements provides sufficient information to determine the source and target of this implied dependency.

5.2. Dependencies and Cross-References

The study revealed 92 cross-references, each of which was formalized as either a refinement or exception dependency. The uniqueness of cross-reference phrases affords maintaining a list of corresponding regular expressions that are used by the tool to assist analysts in identifying these dependencies. The list is iteratively extended as new cross-references patterns are identified. Table 2 presents the final list obtained from this study that was used to check for missed cross-references. The *italicized* regular expressions correspond to dependencies that were acquired from sub-paragraphs (vs. other paragraphs).

Table 2: Cross-Reference Regular Expressions

Freq.	Phrase
3	*as follows*
40	paragraph (\(.+?\))+ of this section
1	paragraph (\(.+?\))+ or (\(.+?\))+ of this section
1	paragraph (\(.+?\))+ through (\(.+?\))+ of this section
4	*the following requirements*
12	this (paragraph | section)
27	§\d+.\d+(\(.+?\))*

From the 91 cross-references, we identified 168 exception and refinement dependencies. Table 3 presents the total number of dependencies derived from external and internal cross-references; these numbers are further sub-divided into the different types of dependency (as defined in Section 4.1).

After the annotated regulation text is parsed, the tool generates a dependency graph from the serialized frames using XSLT. The dependency graph helps analysts visualize the broader context of normative goals and simultaneously reason about refinement and exceptions across multiple goals. Fig. 5 presents the largest, connected dependency graph from all four sections §164.520-164.526 in the HIPAA case study. This graph connects 122 statements or 83% of the total number of acquired statements. A significant challenge for analysts is coordinating this extensive volume of data in a focused and concentrated effort to evaluate software products and designs for regulatory compliance; a topic of our ongoing research.

Table 3: Frequency and Types of Dependencies in HIPAA §164.520-164.526

Dependency	164.520	164.522	164.524	164.526
External	39	6	7	10
Internal	47	6	28	25
Exception	2	3	3	0
Inv. Exc.	6	2	2	0
Refinement	59	0	11	2
Inv. Ref.	19	7	19	33

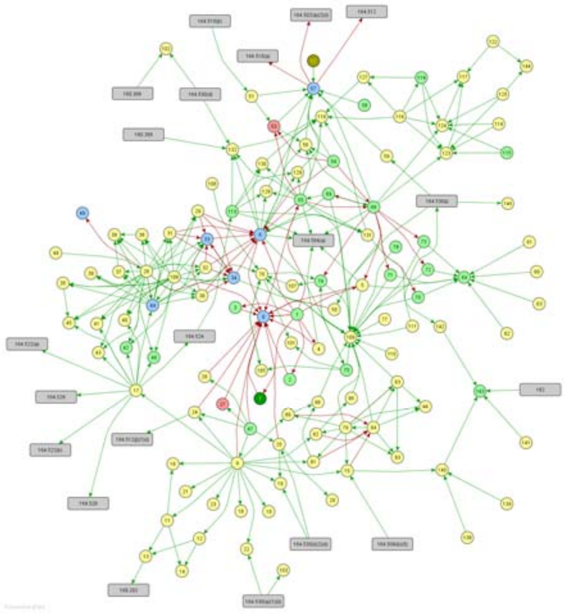

Figure 5: Refinement and Exception Dependency Graph of HIPAA §164.520-164.526

To better understand this challenge, consider a manageable subset of this graph that was derived from the following summarized rules from §164.522(a) and numbered in the order that they were parsed by the tool.

1. **Obligation**: A covered entity must permit an individual to request a restriction to uses and disclosures *permitted under §164.510(b)*.
2. **Exclusion**: A covered entity is not required to agree to a restriction.
3. **Refrainment**: A covered entity may not use or disclose information restricted *under paragraph (a)(1)(i)*.
4. **Permission**: A covered entity may use restricted information to provide emergency treatment.
5. **Permission**: A covered entity may disclose restricted information to provide emergency treatment.
6. **Obligation**: A covered entity must request the restricted information *disclosed under paragraph (a)(1)(iii)* is not further disclosed.
7. **Fact**: A restriction agreed to by a covered entity *under paragraph (a)* is not effective to prevent uses and disclosures *under §164.502(a)(2)(i),164.510(a) or 164.512*.
8. **Permission**: A covered entity may terminate its agreement to a restriction.
9. **Obligation**: A covered entity must document an agreed restriction *in accordance with §164.530(j)*.

The sub-graph appears in Fig. 6: circular nodes represent statements; the node labels correspond to the above numbered statements; the rectangular nodes are cross references to sections that are external to §164.522; the dotted-line arrows point from a statement to one of its exceptions; the solid arrows point from a statement to one of its refinements.

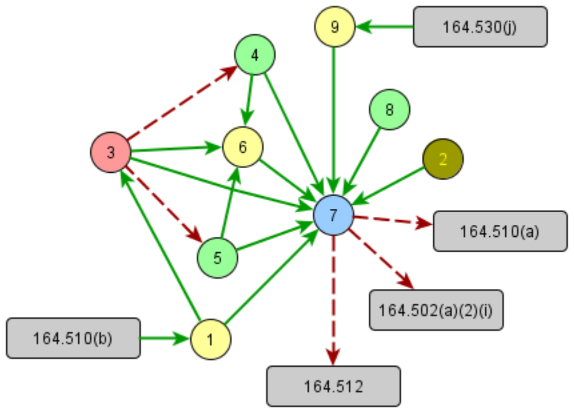

Figure 6: Dependencies from HIPAA §164.524(a)

As previously mentioned, these graphs provide a means to visualize the meaning of dependencies between normative goals. For example, Permissions 4 and 5 (to use or disclose restricted information for emergency treatment) are exceptions to Refrainment 3 (to not disclose such information). Obligation 6, visible as a refinement of Permissions 4 and 5, requires the covered entity to request that the recipient of such information not further disclose the information. Fact 7 coordinates several exceptions to the permissions and obligations in this graph. In this study, facts typically coordinate groups of exceptions and refinements.

6. Comparative Evaluation

There are several important differences between the manual method [4] and FBRAM [7]. The manual method [4] uses terms with definitions similar to FBRAM, including the terms *definition*, *right* (a kind of permission), *obligation*, *anti-right* (refrainment), and *anti-obligation* (exclusions). However, the manual method is viewed as less discriminating than FBRAM because, the manual method: 1) limits rights and obligations to “stakeholder actions” whereas FBRAM extends permissions and obligations to include system actions and states; 2) does not include the concept for “fact;” and 3) does not distinguish cross-references as typed dependencies. In addition, the FBRAM increases coverage over the manual method largely by classifying statements that were not previously classified using these new concepts.

Table 5 presents the total number of statements identified using FBRAM relative to the manual method [4]. FBRAM appears to be more effective in identifying obligations and references than the manual approach; however, we are currently designing an experiment to empirically validate this. We now discuss two important insights from this evaluation:

Increased specificity in the upper ontology and automation in the tool led to better coverage. Because the upper ontology provides analysts with concepts intended to classify every statement and every phrase in the regulatory document, a significant number of “content requirements” were identified using the FBRAM that were missed with the manual method. These requirements describe the required content of privacy notices (520), written denials of access (524) and amendments to electronic medical information (526). Moreover, the formalization of cross-references and use of regular expressions by the tool led to an increase in the number of identified references.

Table 5: Total Number of Statements Identified using FBRAM Relative to the Manual Method for HIPAA §164.520-164.526

Element	**164.520**	**164.522**	**164.524**	**164.526**
Permissions	–1	0	+3	0
Obligations	+33	–2	+10	+13
Exclusions	+2	0	0	0
Fact	+6	+1	0	0
References	+54	+7	+6	+3

The manual approach benefits from inferences by domain experts. The analysts who applied the manual method made inferences that enabled them to derive additional permissions and obligations from facts. These inferences are observable in this comparative evaluation due to steps that have not yet been formalized in FBRAM. The analysts inferred constraints from facts and applied them to previously extracted rules (via cross-references) or used them to create new rules. For example, consider the following fact from HIPAA §164.520(c)(3)(ii) annotated using FBRAM:

```
1   {#F [#s The provisions [!X(c)/(1) of
2   paragraph (c)(1) of this section]] [#a do
3   not apply] {#o to [*1 such group health
4   plan]}}
```

In this example, one analyst used the manual method, to infer a logical expression of constraints that described “such group health plan” from an earlier statement. They then negated this expression using DeMorgan’s Law and copied the new expression into the constraint sets for permissions and obligations in paragraph (c)(1). In contrast, using FBRAM an analyst annotates the logical expression that describes “such group health plan” and resolves the referential ambiguity on lines 3-4 by pasting the expression into the object slot; these tool-supported actions further maintain this important traceability. Because FBRAM maps rules to divisions in the document model, the tool can present analysts with the set of rules that correspond to paragraph (c)(1). However, the analyst must still select the *relevant* rules to which the logical expression should be copied.

7. Discussion and Summary

The FBRAM is designed to help analysts systematically acquire requirements from regulations while reducing ambiguity and maintaining traceability to support compliance by demonstrating due diligence. We applied the FBRAM to four sections §164.520–§164.526 in the HIPAA Privacy Rule that we had previously analyzed using an entirely manual variant of this methodology [4] to assess any improvement gained through automation. We are currently applying the methodology to extract accessibility requirements from the Telecommunications Act of 1996, Section 508. These requirements will be compared with another set of requirements that were acquired by an industry partner from the same regulation using a different approach. Extensions to the FBRAM that are under development include algorithms to generate domain-dependent, lower ontologies from definitions, expressed in the W3C Web Ontology Language (OWL) and identify missing slot values to improve requirements coverage. We are also exploring new requirements organization and presentation techniques to help engineers restrict their focus to only those regulatory requirements that affect their business practices.

The Frame-Based Requirements Analysis Method (FBRAM) makes several assumptions about the regulatory text and analysts’ skills. We assume the markup is distinguishable from the regulation text, using a separate character set, if necessary. The extent to which the markup can be used to identify and resolve ambiguity and to generate useful requirements patterns relies upon the consistent and correct use of English grammar. Although grammar checkers may assist regulatory document authors in satisfying this assumption, we do not expect this method to work on interview transcripts that use verbal cues and similar devices. In addition, we assume analysts can effectively:

identify divisions within the regulation text; consistently classify sentences and phrases using the upper ontology concept definitions; and identify and resolve the logical, attributive and referential ambiguities. We plan to validate these assumptions in a case study with multiple participants.

Acknowlegements

This work was partially supported by NSF ITR Grant #0325269 and an IBM PhD Fellowship.

Appendix A: Context-free Grammar

The context-free grammar is presented in extended Backus-Naur Form. The symbol TEXT is a sequence of characters excluding curly and square brackets.

```
⟨s⟩      := (block | TEXT)*
⟨block⟩  := [ ⟨body⟩ ] | { ⟨body⟩ }
⟨body⟩   := ⟨type⟩? ⟨op⟩? (block | TEXT)* ⟨alt⟩*
⟨type⟩   := HASH LETTER
⟨op⟩     := (cbop)? (crop)?
⟨cbop⟩   := (COPY | CUT | PASTE) NUMBER
⟨crop⟩   := REF LETTER (INDEX SLASH)* INDEX
⟨alt⟩    := (AND | OR) ⟨body⟩
```

References

[1] A.I. Antón, Goal Identification and Refinement in the Specification of Software-Based Information Systems, PhD Thesis, Georgia Tech, 1997.

[2] T.D. Breaux, A.I. Antón, "Analyzing goal semantics for rights, permissions and obligations," *IEEE 13th Int'l Conf. Req'ts. Engr.*, pp. 177-188, 2005.

[3] T.D. Breaux, A.I. Antón, "Mining rule semantics to understand legislative compliance," *ACM Workshop Privacy in the Elec. Society*, pp. 51-54, 2005.

[4] T.D. Breaux, M.W. Vail, A.I. Antón, "Towards regulatory compliance: extracting rights and obligations to align requirements with regulations," *IEEE 14th Int'l Conf. Req'ts. Engr.*, pp. 49-58, 2006.

[5] T.D. Breaux, J. Doyle, A.I. Antón, "Semantic parameterization: a conceptual modeling process for domain descriptions," To Appear: *ACM Trans. Soft. Engr. Methods,* NCSU #TR-2006-35, 2006.

[6] T.D. Breaux, A.I. Antón, "Analyzing regulatory rules for privacy and security requirements," To Appear: *IEEE Trans. Soft. Engr.*, NCSU #TR-2007-9, 2007.

[7] T.D. Breaux, A.I. Antón, "A systematic method for acquiring requirements from regulations: a frame-based approach," *6th Int'l Work. Req'ts for High Assurance Sys.*, 2007.

[8] D.M. Berry, E. Kamsties, "Syntactically dangerous all and plural specifications," *IEEE Software*, pp. 55-57, 2006.

[9] Bureau of Labor Statistics, U.S. Dept. of Labor, *Career Guide to Industries, 2006-07 Edition*, Health Care.

[10] N. Chomsky, *Syntactic Structures*, Janua Linguarum, no. 4, Mouton, p. 116, 1957.

[11] L.M. Cysneiros and J.C.S.P. Leite, "Nonfunctional requirements: from elicitation to conceptual models," *IEEE Trans. Knw. Data Engr.*, 30(5): 328-350, 2004.

[12] D. Dardenne, A. van Lamsweerde, S. Fickas, "Goal-directed requirements acquisition," *Sci. Comp. Programming*, 20:3-50, 1993.

[13] C. Denger, D.M. Berry, E. Kamsties, "Higher quality requirements specifications through natural language patterns," *IEEE Int'l Conf. Soft. – Sci., Tech. & Engr.*, pp. 80-90, 2003.

[14] C.J. Fillmore, "The case for case," In E. Bach and R. Harms (eds.), *Universals in Linguistic Theory*, Holt, Rhinehart, Winston, NY, 1967, pp. 1-90.

[15] A. Fuxman, L. Lin, J. Mylopoulos, M. Pistore, M. Roveri, P. Taverso, "Specifying and analyzing early requirements in Tropos," *Req'ts Engr.*, 9(2):132-150, 2004.

[16] S. Ghanavati, D. Amyot, L. Peyton, "Towards a framework for tracking legal compliance in healthcare," *Advanced Information Systems Engineering*, LNCS v. 4495, pp. 218-232, 2007.

[17] L. Goldin, D.M. Berry, "AbstFinder: A prototype natural language text abstraction finder for use in requirements elicitation," *Auto. Soft. Engr.*, 4(4): 375-412, 1997.

[18] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, "Modeling security requirements through ownership, permission and delegation," *IEEE 13th Int'l Conf. Req'ts Engr.*, pp. 167-176, 2005.

[19] J.F. Horty, Agency and Deontic Logic, Oxford University Press, 2001.

[20] E. Kamsties, "Understanding ambiguity in requirements engineering," *Engr'ing and Mng'ing Soft. Req'ts*, pp. 245-266, Springer, 2006.

[21] S. Konrad, B.H.C Cheng, "Real-time specification patterns," *IEEE 27th Int'l Conf. Soft. Engr.*, pp. 372-381, 2005.

[22] S-W. Lee, R. Gandhi, D. Muthurajan, D. Yavagal, G-J. Ahn, "Building problem domain ontology from security requirements in regulatory documents," *Int'l Workshop Soft. Engr. Secure Sys.*, pp. 43-50, 2006.

[23] S. Overmyer, B. Lavoie, O. Rambow, "Conceptual modeling trhough linguistic analysis using LIDA," *IEEE 23rd Int'l Conf. Soft. Engr.*, pp. 401-410, 2001.

[24] H.B. Reubenstein, R.C. Waters, "Requirements Apprentice: automated assistance for requirements acquisition," *IEEE Trans. Soft. Engr.*, 17(3), 1991, pp. 226-240.

[25] R.C. Schank, R.P. Abelson, *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Discovery*. Lawrence Erlbaum Assoc., Hillsdale, NJ, 1977.

[26] K. Wasson, "Case study in systematic improvement of language for requirements," *IEEE 14th Int'l Conf. Re'qts Engr.*, pp. 6-15, 2006.

[27] P. Zave, M. Jackson, "Four dark corners of requirements engineering," *ACM Trans. Soft. Engr. Methods.*, 6(1):1-30, 1997.