A Systematic Method for Acquiring Regulatory Requirements: A 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 with these laws and regulations, 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 intended and unintended ambiguity and because maintaining traceability across paragraphs and cross-references is essential to demonstrate due diligence in adhering to the law. To address these challenges, we introduce the Frame-Based Requirements Analysis Method (FBRAM) to systematically acquire semi-formal representations of requirements from regulations. The method provides a means to identify and document certain and possibly conflicting interpretations of regulatory requirements using an upper ontology, a context-free markup language and regulatory document model. This tool-supported method produces frame structures that are used to generate an HTML requirements document; this document is then visually inspected by analysts and domain experts who determine the correctness of the resulting requirements documents.*

1. Introduction

National and international standards, regulations and policies impose dependability requirements (e.g., accessibility, safety, security requirements, etc.) on industries and business practices that affect a software system's non-functional properties. Because these requirements are broadly written to govern entire 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.

In this paper, we introduce the Frame-Based Requirements Analysis Method (FBRAM) to extract software requirements from regulations. The FBRAM extends an earlier, entirely manual methodology [4] and 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. Throughout this paper, we introduce the FBRAM using an example from a HIPAA Privacy Rule case study. The 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 FBRAM; and Section 5 concludes with a brief discussion and summary.

2. Related Work

In requirements engineering, goals are used to model requirements, as in KAOS [10] and GBRAM [1]. *Goals* represent states that a system must achieve, maintain or avoid. Elaborated goals can include normative statements about rights, permissions and obligations [2]. *Rights* and *permissions* describe actions that stakeholders are permitted to perform, whereas *obligations* describe actions that stakeholders are required to perform. In this paper, we adapt a frame-based representation to elaborated goals in which a *frame* corresponds to a concept, comprising *slots* that represent stereotypical properties of that concept [13, 18]. We use frames to formalize a subset of the deep structure or semantics of regulatory language [8].

Ontologies are formal collections of knowledge consisting of concepts and properties; an *upper ontology* describes domain-independent concepts. Examples from different upper ontologies can be found in Cyc [17] and WordNet [12]. We align our frame concepts and properties with an upper ontology consisting of requirements knowledge in the form of a model of elaborated, regulatory goals.

Lexicons and natural language (NL) patterns are often used to analyze requirements. Wasson et al. [19] and Cysneiros and Leite [9] employ lexicons to improve natural language (NL) requirements analysis. NL patterns are used to identify critical real-time properties

[1] U.S. Public Law No. 104-191, 110 Stat. 1936 (1996)
[2] U.S. Public Law No. 104-104, 110 Stat. 56 (1996)

[15] and improve requirements quality for embedded systems [11]. FBRAM complements this work by providing a means to formalize NL patterns using a standard lexicon (ontology). Lee et al. describe an ontological approach to acquire requirements from regulations [16]. Herein, our contribution to their work is a method to help analysts systematically decompose verbose regulatory statements into requirements.

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, the term *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 become better 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 in this paper: logical, attributive, and referential. Because one of these ambiguities can affect the interpretations of multiple, related requirements in a requirements document, Kamsties classifies these ambiguities as *requirements document ambiguity* [14].

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 [7]. 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]. Herein, 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 of 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 [6]. 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.

4. Frame-Based Requirements Analysis

In the Frame-Based Requirements Analysis Method (FBRAM), analysts manually annotate a regulatory document and an accompanying tool parses the annotations to extract regulatory requirements (Figure 1). The manual annotation process uses the following three artifacts:

1. A reusable *upper ontology* of requirements concepts and properties, used to classify regulatory statements independently of any single regulatory domain.
2. A *context-free markup* language that describes the deep structure of natural language [8] 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.)

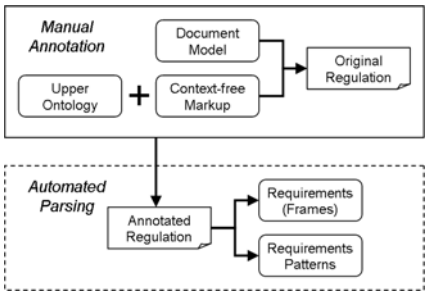

Figure 1: Overview of the FBRAM

During the manual annotation process (top of Figure 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. The 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.
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 to the analyst. The successfully parsed frame objects and patterns are serialized using the W3C eXtensible Markup Language (XML) and then transformed into the Hypertext Markup Language (HTML) using eXtensible Stylesheet Language Transformations (XSLT). The analyst uses the HTML representation to validate whether the frame-based semantics of the applied annotations match their intended interpretation of the regulations.

4.1. The Upper Ontology

The upper ontology describes knowledge about the semantic structure of regulatory requirements that is domain-independent. Figure 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.

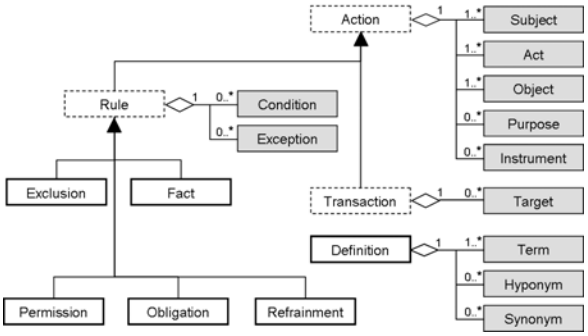

Figure 2: Regulatory Requirements Upper Ontology

The concepts in the upper ontology are connected by two types of arrows: (1) arrows that terminate with dark triangles that lead from sub-classes to super-classes; and (2) arrows that terminate with white diamonds that lead from properties to classes, containing those properties. The upper ontology consists of three types of concepts:

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

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.
- *Definition* means a statement that describes a term-of-art by equivocating the term with a phrase or other word (synonym) or by listing its kinds, specializations or varieties (hyponyms).
- *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*.
- *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.
- *Exception* is the condition(s) that must be false before an entity acts.
- *Term* is a term-of-art with a specialized meaning.
- *Hyponym* is a word or phrase that represents a kind, specialization or variety of a more general word or phrase.
- *Synonym* is a word or phrase that has an equivalent meaning to another word or phrase.

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 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 exclusion statements or exception phrases, which can be formalized as priorities between requirements [6]. Regulatory statements may begin in one paragraph and end in a sub-paragraph. Usually, this syntactic device 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. 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 formalized 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 above excerpt appears in Figure 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 the 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.

4.3. The Context-free Markup

Analysts use the context-free markup language to codify their interpretation of a regulation text. The interpretation requires analysts to align concepts from the upper ontology with regulation sentences and phrases, removing logical, attributive and referential ambiguities. The context-free grammar for the markup appears in Appendix A. Table 1 presents concept codes that are used in the markup 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	s	Subject
F	Fact	t	Target
O	Obligation		

The running example from HIPAA Privacy Rule §164.520(a)(2)(i)(B)(ii) appears below with the markup in **bold**; the document model has not been applied to this example for easier reading:

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

The markup is used to structure regulatory text into two types of nested blocks denoted by opening and closing brackets: (1) *pattern blocks*, denoted by curly "{}" brackets, indicate the start of a requirements pattern or sub-pattern; and (2) *value blocks*, denoted by square "[ ]" brackets, indicate spans of text that will be mapped to slot values by the parser. In addition, 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" are mapped to logical connectives using the operators "&" and "|" for logical-and and logical-or, respectively (see lines 4, 7, 15).

To resolve attributive and referential context-sensitive ambiguities, we introduced 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 16-17) and "such group health plan" (line 21) introduce this type of ambiguity. If the paste operator is applied to a block that contains text, as is the case in these two phrases, the text in the block will be replaced by the pasted text.

The parser detects syntax and semantic errors, such as missing brackets, cycles that occur in the copy/ cut/ paste operations, unknown concept codes, etc., and alerts the analyst who must then resolve these errors.

4.4. Requirements

Parsing the annotated regulation text yields requirements that are formalized as frame objects. These frames are serialized using XML and transformed into HTML using XSLT. In HTML, the requirements are presented in a table format. Parsing the example markup from Section 4.3 yields two requirements; the second requirement is presented in Figure 4 using the same table format that is used in practice.

Frame: 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 as defined in §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 Created after Parsing the Annotated Regulation

The example in Figure 4 begins with the statement frame type (Frame), the requirements pattern (Pattern) and the traceability information (Trace) with the requirement ID, the line number and line index and the corresponding paragraph number in the regulation text. Next in the table, 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 Figure 4), the values are presented as trees comprised of logical-and branches (solid line) and logical-or branches (dotted line).

5. Discussion and Summary

The FBRAM is designed to help analysts systematically acquire requirements from regulations while addressing two challenges: the need to reduce ambiguity and maintain traceability to support the need

to demonstrate due diligence. We applied the FBRAM to four sections §164.520-§164.526 in the HIPAA Privacy Rule that we previously analyzed using an earlier, entirely manual variant of this methodology [4] to assess any improvement gained through automation. In addition, we are currently applying the methodology to extract accessibility requirements from the Telecommunications Act of 1996, Section 508. These requirements will be compared with a different set of requirements that were acquired from the Telecommunications Act using a different approach by an industry partner. 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), prioritize requirements by using exceptions, and improve requirements coverage and consistency by identifying missing slot values and checking pattern correspondences to upper ontology concepts.

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 and annotate 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.

Appendix A: Context-free Grammar

The context-free grammar is presented in 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⟩    := (COPY | CUT | PASTE) NUMBER
⟨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] D.M. Berry, E. Kamsties, "Syntactically Dangerous All and Plural Specifications," *IEEE Software*, pp. 55-57, 2006.

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

[9] 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.

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

[11] 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.

[12] C. Fellbaum, "WordNet: an electronic lexical database," MIT Press, 1998.

[13] 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.

[14] E. Kamsties, "Understanding Ambiguity in Requirements Engineering," *Engr'ing and Mng'ing Soft. Req'ts*, pp. 245-266, Springer, 2006.

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

[16] S-W. Lee, D. Muthurajan, R. Gandhi, D. Yavagal, G-J. Ahn, "Building decision support problem domain ontology from sercurity requirements to engineer software-intensive systems," *Int'l J. Soft. Engr. & Kno. Engr.* 16(6):, 851-884, 2006.

[17] C. Matuszek, J. Cabral, M. Witbrock, J. DeOliveira, "An introduction to the syntax and content of Cyc," *AAAI Spring Symp. Formalizing and Compiling Bg. Knowledge and its Apps. to Knowledge. Rep. and Question Answering*, pp. 44-49, 2006.

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

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