Regulatory Requirements Traceability and Analysis Using Semi-formal Specifications

Travis D. Breaux[1] and David G. Gordon[2]

[1] Institute for Software Research, Carnegie Mellon University, Pittsburgh, PA, USA
breaux@cs.cmu.edu
[2] Engineering and Public Policy, Carnegie Mellon University, Pittsburgh, PA, USA
dggordon@andrew.cmu.edu

Abstract. Information systems are increasingly distributed and pervasive, enabling organizations to deliver remote services and share personal information, worldwide. However, developers face significant challenges in managing the many laws that govern their systems in this multi-jurisdictional environment. In this paper, we report on a computational requirements document expressible using a legal requirements specification language (LRSL). The purpose is to make legal requirements open and available to policy makers, business analysts and software developers, alike. We show how requirements engineers can codify policy and law using the LRSL and design, debug, analyze, trace, and visualize relationships among regulatory requirements. The LRSL provides new constructs for expressing distributed constraints, making regulatory specification patterns visually salient, and enabling metrics to quantitatively measure different styles for writing legal and policy documents. We discovered and validated the LRSL using thirteen U.S. state data breach notification laws.

Keywords: requirements specification, traceability, domain specific languages, legal requirements.

1 Introduction

Increasingly, new government laws and regulations are being introduced to address new challenges posed by emerging information systems (IS). For software developers, this emergence of IS-related laws places constraints on what systems must do (the matter of requirements) and whether system requirements documents include all the right requirements (the matter of validation). In the United States, a prominent example includes the recent surge in state data breach notification laws, which have been empirically observed to reduce identity theft [27]. Collectively, these laws combine the act of notification to various stakeholders with technical security controls (e.g., encryption, data destruction, etc.) targeted at different information types, business practices and consumers. These laws require the development of a new, interstate information system that most businesses in the U.S. must participate in by modifying their organizational practices and software systems to account for data breaches and to deliver notices under specifically governed situations. Many of the legally imposed security requirements follow conventional security design wisdom; however, the legally

© Springer-Verlag Berlin Heidelberg 2013

mandated parameters in these requirements vary across jurisdiction. For example, using encryption or disposing of unnecessary data is a security best practice; however, the required type of encryption and length of data retention does vary across state and national boundaries. The challenge for developers, especially in small businesses, is to distill these regulations into actionable requirements that are traceable across their business practices. Simply skimming a regulation for keywords or phrases exposes software developers and users to the risk of missing subtle constraints and relationships. Example relationships affect who is covered, under what circumstances, and to what extent. Finally, a systematic, traceable and comprehensive account of existing legal requirements can facilitate the integration with industry standards to further articulate how businesses comply with government laws [28].

We believe existing approaches to governance, which consists of independently published, paper-based laws and policies, can no longer scale with rate of technology innovation. Furthermore, if an honest expectation of compliance is to be preserved in this new environment, regulations must be made accessible to policy makers, business analysts and software developers, alike. We propose that regulators and industry can reach a coordinated solution wherein regulations become a computational software artifact that are dynamically linked across jurisdictions and that enable tool-based requirements analysis. These computational artifacts can integrate with industry standards to become more easily comparable and addressable in a manner that reflects the jurisdiction of the computer's memory state, users' location, and the rate of technological change. To this end, we report our efforts to develop a legal requirements specification language (LRSL), derived from grounded analysis of conflicting regulations from multiple jurisdictions. By translating requirements into the LRSL, document authors can design and debug their requirements documents using improved tracing, patterns and metrics that we discuss in this paper.

The remainder of the paper is organized as follows: in Section 2, we discuss related work; in Section 3, we introduce the LRSL by example; in Section 4, we present our research methodology to discover and validate the LRSL; in Section 5, we summarize our research findings, including techniques for navigating and cross-linking legal requirements; and in Section 6, we conclude with our discussion and summary.

2 Related Work

Related work includes research on requirements languages, extract requirements from laws, prioritize requirements, and model legal documents and their legal effects.

Requirements specification languages (RSLs), including requirements modeling languages (RMLs), have a rich history in requirements and software engineering [20]. RSLs include informal, natural language descriptions to provide readers with context and elaboration, and formal descriptions, such as mathematical logic, to test assumptions across requirements using logical implications [13]. Goal-oriented languages, such as i* [36] and KAOS [11], and object-oriented notations, such as ADORA [17], include graphical notations to view relationships between entities, such as actors, actions and objects. Because of computational intractability and undecidability of using highly expressive logics [16], RSLs often formalize only a select class of requirements phenomena, e.g., using description logic [5] and various temporal logics, such as interval [26], real-time [11] or linear [14] temporal logic. Consequently, RSLs and RMLs may struggle with the balance between expressivity and readability [13]. Unlike i*, KAOS and ADORA, the LRSL proposed herein is designed for the law and policy domain by integrating formal expressions of document structure using regular expressions with semi-formal expressions of rights, permissions and obligations using text-based predicates and annotations. Unlike frame-based approaches that seek to classify phrases by logical roles [7], our LRSL simulates how policies are written by formalizing the cross-links among requirements in ways originally specified by regulators, and preserving traceability to the original legal document references. The aforementioned notations do not account for this integration of requirements and original sources in policy and law.

Approaches to formalize laws in requirements engineering have focused on *prescriptions*, called rights, permissions and obligations [6], ownership and delegation [15], and production rule systems [22]. In addition, cross-references within and among laws have been shown to coordinate definitions, exceptions and refinement and must be addressed in a comprehensive legal requirements management strategy [8]. Recent analysis of external cross-references emanating from the Health Information Portability and Accountability Act (HIPAA) shows the potential for conflicts between HIPAA and other laws [23]. Recently, Siena et al. describe the *Nómos 2* framework to model norms, which they claim can be used to determine compliance with law [31]. We believe the LRSL could be combined with the inference layer provided by *Nómos 2* to reason about legal requirements coverage.

Research in artificial intelligence (AI) and law has long sought to encode regulations into formal models. Among many others, this includes work by Biagoli et al. [2] and Sergot et al. [29] to express statutes as logic programs. Allen and Saxon describe the A-Hohfeld language [1] based on Hohfeld's legal concepts [18]. The language is used to reason about legal powers, rights, and duties. More recently, Sergot describes a theory of normative positions based on the Kanger-Lindahl theory [30]. The aim of this work was to develop automated legal reasoning tools. Because regulatory documents were not intended to be formalized and often contain ambiguities, our approach has been to develop methods to express a normative semi-formal semantics [9] that yield "islands of formality" while preserving legal ambiguity for later analysis by an appropriate legal analyst. Stamper argues this approach provides an "economy of expression" in regulatory requirements analysis [32], which is a commonly held view of domain specific languages, in general [25]. Thus, our approach is concerned with repeatable, semi-formalization that strictly deals with issues of ambiguity and document structure. Approaches to formalize judicial legal arguments, such as LegalXML, concern a different problem. Judicial reasoning can be used to refine one's interpretation of regulations, which aim to explore in future work.

Within the limited scope of our paper, Bourcier and Mazzega propose a vision to represent legal documents using networks, wherein legal articles are nodes connected by edges that represent either "legal influences" or quotations, called "legal selection" [3]. They advocate for content-based measures that account for legal effects produced by normative statements [3]. Massey and Antón propose several metrics for measuring regulation dependency and complexity [21]. Our LRSL addresses these needs in three respective ways: 1) by codifying legal influences in typed, priority-based relations (including exemptions, pre-emptions and waivers) that cross-link between portions of regulatory documents; 2) by assigning types to cross-references between individual requirements (a much finer level of detail than Bourcier and Mazzega) that encodes certain legal effects, such as refinement, exception and pre- and post-conditions; and 3) by measuring these relations to quantify complexity exhibited in legal writing styles.

2.1 Writing Legal Requirements Specifications

The Legal Requirements Specification Language (LRSL) makes several assumptions about the domain of legal requirements. These assumptions were first observed in our case study and thus incorporated into the LRSL syntax and semantics described here. As we discuss later, they support what we believe are good requirements specification practices. In addition to these assumptions, the analyst who translates a law into the LRSL uses several techniques that we have previously identified [4, 6]: phrase heuristics to identify modal verbs corresponding to rights, obligations and prohibitions; re-topicalization shifts the subject of a requirement to a principal actor; case-splitting to separate one compound requirement into separate requirements; and balancing rights and obligations to identify inferred requirements.

In the discussion that follows, we use the following excerpt in Figure 1 that was acquired from Arkansas Title 4, §110.105 to present the LRSL.

4-110-105. Disclosure of security breaches.

(a)(1) Any person or business that acquires, owns, or licenses computerized data that includes personal information shall disclose any breach of the security of the system… to any resident of Arkansas…

(2) The disclosure shall be made in the most expedient time and manner possible and without unreasonable delay, consistent with the legitimate needs of law enforcement as provided in subsection (c) of this section

Fig. 1. Excerpt from the Arkansas (AR) Title 4, §110.105 of the Personal Information Protection Act

The analyst converts statements and phrases from the original text into expressions in the LRSL. Figure 2 shows the excerpt from Figure 1 expressed in the LRSL: reserved keywords, special operators, and line numbers along the left side appear in bold. The `DOCUMENT` keyword (on line 1) assigns a unique index to the specification. The `SCHEMA` keyword (on line 2) precedes an expression consisting of *components* in curly brackets. Each component corresponds to a different reference level within the document model, beginning with the topmost level, in this case the title and chapter. References within the specification are parsed by the automated parser using this schema. Line comments are denoted by the "`//`" operator. We use the ellipsis "…" to denote omissions from the specification to simplify presentation in this paper.

The document model consists of sections and nested paragraphs, expressed in the LRSL by the `SECTION` and `PAR` keywords, respectively. These keywords are followed by a reference and an optional title: line 5 shows the section reference

4-110-105 followed by the title from Figure 1; sub-paragraphs (a) and (1) follow on lines 6-7.

Requirements consist of roles, pre-conditions and prescriptive clauses, organized into first-order logic expressions using operators "|" for logical-or (see line 9, Figure 2), and "&" for logical-and. Roles are noun phrases that describe the actors or objects to whom the requirements apply. Next follows the *clause*, preceded by a ":" and starting with a verb. Modal verbs indicate requirements, such as "shall" to indicate an obligation (see lines 13 and 16); otherwise, the clause is a pre-condition that is often assumed to be an implied permission (see line 10). Finally, analysts can link categories to requirements using the keyword `ANNOTATE` (see lines 11 and 17).

```
1    DOCUMENT US-AR-4-110
2    SCHEMA {title:4}-{chapter:110}-{section:\d+}{par:\([a-z]\)}{par:\(\d+\)} //...
3    TITLE 4-110 Personal Information Protection Act
4
5    SECTION 4-110-105 Disclosure of security breaches
6    PAR (a)
7    PAR (1)
8    person
9      | business
10     : acquires, owns, or licenses computerized data that includes personal
         information
11     ANNOTATION implied-permission
12     PRECEDES (a) #2 // comment: a pre-condition
13     : shall disclose a breach of the security of the system to any resident
14   PAR (2)
15   disclosure
16     : shall be made in the most expedient time and manner possible and without
         unreasonable delay
17     ANNOTATE timing-requirements
18     REFINES (1) #2
19     EXCEPT (c)(1) #1
```

Fig. 2. Excerpt from Arkansas 4-110-105 expressed in the LRSL

Cross-references serve to coordinate requirements and constraints expressed in different regions of a regulatory text. In some regulations, cross-references are coarse-grained, meaning they refer to whole paragraphs; in which case, the analyst must determine which specific requirements in that paragraph are intended. The LRSL allows analysts to express coarse references with the added ability to distinguish which requirements they deem as applicable; preserving their interpretation for later review by other analysts and legal counsel.

We discovered three types of cross-references in our case study (see Section 5):

- `REFINES`, with the inverse relation `REFINED-BY`, indicates that this requirement is a sub-process or quality attribute that describes how another requirement is fulfilled.
- `EXCEPT`, with the inverse relation `EXCEPT-TO`, indicates that this requirement has an exception (another requirement). If the pre-conditions of the exception are satisfied, then this requirement does not apply (it becomes an exclusion, e.g., *is not required*).
- `FOLLOWS`, with the inverse `PRECEDES`, indicates that this requirement is a *post-condition* to another requirement, e.g., this requirement is permitted, required, or prohibited after the other requirement is fulfilled.

In Figure 2, the command keyword `REFINES` (line 18) establishes a refinement relation from the preceding requirement (line 16) to the second requirement (line 13) in paragraph (1). The refinement on line 16 is a quality attribute, because it elaborates when the “disclose” action must occur: “expediently, without delay.” Generally, quality attributes refine another requirement’s action or object in the LRSL.

Section and paragraph references are either absolute or relative: absolute references begin from the top-level component in the schema and walk each component to the paragraph that contains the target requirement; relative references are matched by the nearest ancestor in the hierarchical schema, beginning with the parent paragraph. References in the LRSL can be expressed as a single paragraph, such as “(1)” or a paragraph range, such as “(1)--(3)”. Other operators exist to refer to the last paragraph and all sub-paragraphs (i.e., the transitive closure). Rule selection is done in three ways: a) by default, references select all rules within the referenced paragraphs; b) singular paragraph references followed by the ordinality operator “#” and a number *n* will identify the *n*th rule in that paragraph (see lines 12, 18, or 19); and c) references followed by a comma-separated list of annotations will find rules that share those annotations (e.g., all “permissions” or all “timing-requirements”). Finally, multiple references can be joined in logical expressions using simple Boolean logic operators: “&” for logical-and, and “ | ” for logical-or, and parentheses for associativity.

Definitions describe the actors and objects in the system. In Figure 3, paragraph (a) on lines 4-8 contains a definition for *data storage device*, indicated by the “=” operator. Definitions are expressed using the Boolean logical operators for logical-and and logical-or, in addition to the inclusion operator “<”, which means “includes” and precedes examples or sub-classes (see line 7), and the exclusion operator “~”, which means “excludes” (see line 13). By default, definitions apply to the paragraph in which they occur, unless instructed otherwise using the `INCLUDE` keyword, followed by two references: the source paragraph containing the definitions, and the target section or paragraph to which the definitions will apply. The instruction in Figure 3, line 2 tells the parser to apply all the definitions from paragraph (5) and all sub-paragraphs (indicated by the “*”) to §215. In contrast, the `INCLUDE EXTERNAL` instruction on line 15 instructs the parser to lookup the definition “payment card” by finding a regulatory specification indexed by NV-205.602, and to apply this definition to §215. This second usage enables reuse of definitions from and across multiple regulations. In other words, the LRSL supports tracing dependencies from one or more definitions to other definitions and requirements across multiple specifications.

```
1   PAR 5.
2   INCLUDE 603A.215.5* 603A.215*
3   PAR (a)
4   data storage device
5     = device
6     & stores information or data from any electronic or optical medium
7     < computers
8       | cellular telephones
9   // ...
10  PAR (c)
11  facsimile
12    = electronic transmission between two dedicated fax machines using Group 3
      or Group 4 digital formats...
13    ~ onward  transmission  to  a  third  device  after  protocol  conversion,
      including, but not limited to, any data storage device
14  PAR (d)
15  INCLUDE EXTERNAL NV-205.602 603A.215* "payment card"
```

Fig. 3. Excerpt from Nevada 603A.215(5)(c) expressed in the LRSL

2.2 Tool Support and Generated Artifacts

The LRSL is complemented by an automated parsing tool, which checks the language for syntax errors, such as malformed or unassociated logical expressions, and semantic errors, such as incorrect references, empty relations that refer to no rules, unreferenced definitions, and cycles among relations of the same type, e.g., REFINES, EXCEPT, FOLLOWS. The parser applies Deontic annotations to requirements based on established phrase heuristics [6], and the model created by the parser can then be used to find requirements as needed, e.g., find all the obligatory timing requirements. The parser-constructed model is exportable to other formats, such as the HyperText Markup Language (HTML), the Graph Markup Language (GraphML), and the eXtensible Markup Language (XML). Each format offers a different perspective: the HTML allows users to browse the specification by clicking hyperlinks, viewing definitions and referenced rules *in context* of a single rule; the GraphML allows users to visualize relationships across multiple requirements and identify regulatory patterns, which we discuss in Section 4.2; and the XML enables data inter-operability, which may eventually include exporting the model to the Requirements Interchange Format (RIF) and the User Requirements Notation (URN). Figure 4 shows a graph generated from the LRSL example in Figure 2: text labels include a unique requirement identifier (e.g., AR-7), followed by the requirement clause (abbreviated in this figure). Nodes are colored by whether they are permissions (green), obligations (yellow), and prohibitions (red) based on annotations generated by the phrase heuristics. Directed edges represent relations and point to referenced rules as follows: solid edges are REFINES, dashed edges are EXCEPT, and dotted edges are FOLLOWS relations. This support addresses previously identified limitations in analysis tools, including the need to reference requirements at the statement-level [19, 24] and the need to add types to cross-references [34].

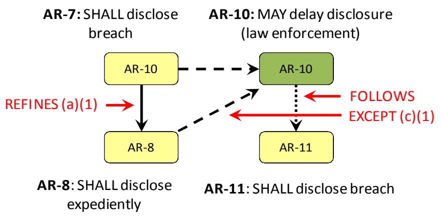

AR-7: SHALL disclose breach
AR-10: MAY delay disclosure (law enforcement)
AR-10
AR-10
REFINES (a)(1)
FOLLOWS
EXCEPT (c)(1)
AR-8
AR-11
AR-8: SHALL disclose expediently
AR-11: SHALL disclose breach

Fig. 4. Excerpt from Arkansas §110.105 expressed in GraphML

3 Research Methodology

Our study aims to describe variation in regulations across multiple jurisdictions. In preparation to achieve this goal, we focus on developing a method to extract and encode these regulations. We selected a single theme (data breach notification) to illustrate dependencies between functional system requirements and personnel responsibilities. In the United States, this theme represents the recent enactment of 46 state and territorial laws from 2002-2011, each governing personal information about state residents. For distributed and pervasive systems, variations in these laws require businesses to reconcile different legally required practices for customers of different states. The laws we selected are as follows:

- **AK**: *Personal Information Protection Act*, Alaska Chapter 45.48, enacted 2009.
- **AR**: *Personal Information Protection Act*, Arkansas Chapter 14.110, enacted 2005.
- **CT**: *Breach of Security Regarding Computerized Data Containing Personal Information*, Connecticut General Statute 36a-701b, enacted 2006.
- **MA**: *Security Breaches*, Massachusetts Chapter 93H, enacted 2007.
- **MA-S**: *Standards for the Protection of Personal Information of Residents of the Commonwealth*, Massachusetts Chapter 17, enacted Sep. 19, 2008.
- **MD**: *Personal Information Protection Act*, Maryland Subtitle 14-35, enacted 2008.
- **MS**: *(no title given)* Mississippi House Bill 583. Enacted 2011.
- **NV**: *Security of Personal Information*, Nevada Chapter 603A, enacted 2006.
- **NY**: *Notification of Unauthorized Acquisition of Personal Information*, New York General Business Law 899-aa, enacted 2005.
- **OR**: *Oregon Consumer Identity Theft Protection Act*, Oregon Chapter 646A, enacted 2008.
- **UT**: *Protection of Personal Information Act*, Chapter 44, enacted 2006.
- **VT**: *Protection of Personal Information*, Vermont Chapter 26, enacted 2007.
- **WI**: *Notice of Unauthorized Access to Personal Information*, Wisconsin Chapter 134.98, enacted 2006.

We down-selected from 46 to 13 laws as follows: first, we surveyed legal expert with seven years of privacy and security law expertise to highlight industrial challenges, resulting in AR, MA-S, MA, MD, and NV; and second, we selected three laws with the largest number of pages, resulting in AK, OR, and VT. The remaining laws had noteworthy, uncharacteristic features: unique (WI) or broad (NY) definitions, the most recent law to expose evolution (MS), interfaces to external agencies (CT), and severe penalties (UT). In addition, we constructed document schemas for 49 data breach laws to validate the construction of SCHEMA expressions across a larger dataset.

Two investigators (the authors) separately translated each statement in each law using the LRSL. The translation includes a general classification of each statement, as a definition, requirement, exemption, etc., and writing an expression in the language to characterize the statement. Definitions were identified by key phrases, such as "*x* means *y*", where a term *x* has the logical definition *y*. Requirements and exemptions were identified using phrase heuristics identified by Breaux et al. [6]. Comments were used in the translation to capture questions, issues and other discrepancies. We maintained a *caveats list* of translation strategies that reflect unusual cases and how the parser should treat such cases, and a *proposed changes list* of requirements with examples for new language constructs. For each new construct, we reviewed each law to update the translation to ensure consistency across the entire dataset. Corbin and Strauss state, "The essential element of theory is that categories are interrelated into a larger theoretical scheme" and a theory represents an "abstract rendition of that raw data" [10]. In this regard, the LRSL is an expression of a grounded theory in a context-free grammar that explains how legal requirements are expressed. The theory extends prior theoretical findings [8] and consists of concepts (rights, obligations, permissions, etc.) and cross-reference relationships (refinements, exceptions and pre- and post-conditions) that link these concepts together and explains how to trace legal definitions and requirements across a legal text. Our analysis checked for internal consistency, and if the language covers variations across all cases that we studied.

Grounded theories are limited to studied cases and new cases may invalidate the theory.

4 Research Findings

The translation of thirteen laws by two investigators (the authors) yielded 808 statements, required an average of 2.26 minutes per statement with the longest document consisting of 148 statements and requiring an average of 5.5 hours. Each investigator spent an average total of 30.5 hours to encode the thirteen laws. Figures 5 and 6 present summary statistics for the units of analysis encoded in the LRSL. Recall these laws cover the same theme (data breach notification). We observed the number of definitions did not vary greatly and that the number of exemptions was a matter of writing style; neither definitions nor exemptions are proportional to the number of requirements in this dataset.

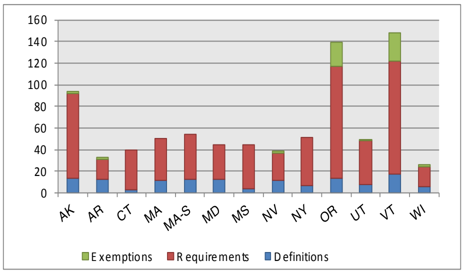


Fig. 5. Summary Units of Analysis– Statements

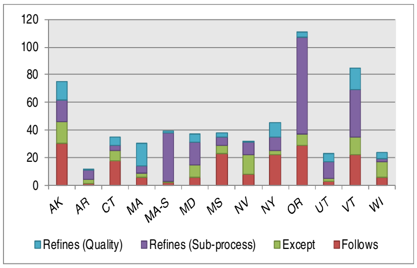


Fig. 6. Summary Units of Analysis – Reference

The references reported in Figure 6 originate from multiple origins, including: *anaphora*, which is indicated by determiners (e.g., such) and pronouns (e.g., this); *case-splitting*, which is indicated by English conjunctions (and, or) separating verb clauses that follow a modal phrase (e.g., must, may, shall); and direct references to sections and paragraph that may be anaphoric (*this* section, *this* paragraph) or indexed by paragraph number, such as "paragraph (a)." Table 1 presents summary statistics for each of these observed origins. For direct references, we present the number of corresponding rules identified by the original reference for each regulation, called *direct literal* (dL), and the number of corresponding rules indexed by the operationalized reference using the LRSL language construct, called *direct indexed* (dI). Because the operationalized references are more precise, we can calculate the ambiguity loss, which is the proportion of false positives referenced by an ambiguous cross-reference and which we express as (dL – dI) / dL. The operationalized references expressed in the LRSL, which allow analysts to link requirements to only true positives, reduce reference ambiguity by 50-93%.

Table 1. Cross-Reference Origins and Ambiguity

State Law	Anaphora	Case Split	Direct	Direct Literal	Direct Indexed	Ambiguity Reduction
AR	2	4	5	24	7	0.708
AK	16	19	35	143	36	0.748
CT	8	3	3	14	5	0.642
MA	20	1	3	45	3	0.933
MA-S	4	34	1	2	1	0.500
MD	4	12	21	62	23	0.629
MS	7	4	6	19	6	0.684
NV	7	3	13	83	14	0.831
NY	16	6	8	41	17	0.585
OR	29	15	24	190	24	0.874
UT	3	12	17	136	40	0.706
VT	36	10	25	269	32	0.881
WI	6	0	18	78	20	0.744

We developed metrics to measure stylistic properties that affect the extent to which an analyst must make inferences to resolve requirements ambiguity. Using the metrics, we observed the following styles: *cascading refinement* occurs when sections are organized around high-level goals in which goal-refinements and post-conditions are expressed in nested paragraphs; *reference uniqueness* occurs when cross-references refer to the fewest number of requirements, ideally one; and *block formatting* occurs when the paragraphs contains multiple requirements, but are rarely nested.

We now discuss other observations from this case study.

4.1 Shaping Conditionality and Coverage

Conditionality is the extent to which a legal requirement is conditioned by *who stakeholders are* and *what events have occurred*, which we call pre-conditions. Definitions and exemptions shape conditionality by relaxing or tightening the meaning of terms and thus scaling the number of possible situations those terms cover. We discuss two ways that these effects are observed through the LRSL: (1) cross-linking of terms-of-art to paragraphs and to pre-conditions, requirement clauses and other definitions; and (2), cross-linking of exemptions to modify pre-conditions and clauses.

The LRSL parser automatically cross-links definitions to requirements by matching terms-of-art in definitions with phrases in. Recall from Figure 3 the definitions for terms *data storage device* (line 4) and *facsimile* (line 11) and the imported term payment card (line 15) from another law, NV §205.602. The instructions `INCLUDE` (lines 2 and 15) orchestrate these definitions by applying them to all sub-paragraphs in §603A.215. This includes linking to other definitions, such as the phrase on line 13 that excludes "data storage device" from the onward transmission of a facsimile. Figure 7 illustrates this linking to requirements in paragraphs §603A.215(1) and (2): the underlined phrases match the terms-of-art from Figure 3 as determined by the parser. Both *when to apply* a prescription and *the extent of* the prescription can be computationally adjusted by relaxing or tightening definitions using the includes "<" and excludes "~" operators, respectively.

```
1    SECTION 603A.215
2    PAR 1.
3    data collector
4      : does business in this State
5      : accepts a payment card in connection with a sale of goods or services
6      : shall comply with the current version of the Payment Card Industry (PCI)
         Data Security Standard...
7      FOLLOWS #1 & #2
8    PAR 2.
9    data collector
10     : does business in this State
11     EXCEPT 1.
12   PAR (a)
13     : does not use encryption to ensure the security of electronic transmission
14     : shall not transfer any personal information through an electronic, non-
         voice transmission other than a facsimile to a person outside of the
         secure system of the data collector
15     FOLLOWS 2. #1 & 2.(a) #1
16   PAR (b)
17     : does not use encryption to ensure the security of the information
18     : shall not move any data storage device containing personal information
         beyond the logical or physical controls of the data collector or its data
         storage contractor
19     FOLLOWS 2. #1 & 2.(b) #1
```

Fig. 7. Excerpt from Nevada §603A.215(1) and (2)

For example, if we redefine *payment card* to exclude *gift card*, then the scope of when to apply the requirement to comply with the PCI DSS standard (on line 8, Figure 10) would be further restricted to omit the case of gift cards. Alternatively, if *data storage device* were redefined to include *USB drives*, then the extent of the prohibition on moving such devices (on line 18, Figure 7) would be extended to include this interpretation. The ability to shape *when to apply* and *the extent of prescriptions* using the LRSL can enable regulators and businesses to evolve conditionality as new technologies emerge over time.

Whereas definitions shape terms used in pre-conditions and requirements clauses, exemptions fine-tune what is excluded from pre-conditions and clauses. Figure 8 shows a description of the role "telecommunications provider" with a role constraint on line 4. The EXEMPT keyword instructs the parser to exclude this role and constraint from all rules in §215 and all sub-paragraphs therein. While such an exemption could be stated in a definition using the excludes operator "~", exemptions provide a mechanism to tighten meanings across a document cross-section, unbounded by a single term-of-art or definition.

```
1    PAR 4.
2    PAR (a)
3    telecommunications provider
4      : acts solely in the role of conveying the communications of other persons,
         regardless of the mode of conveyance used...
5      EXEMPT 603A.215 *
```

Fig. 8. Excerpt from Nevada §603A.215(4)(a) expressed in LRSL

Figure 9 illustrates how constraints, expressed as definitions and exemptions, are traced by the parser through parser instructions. The INCLUDE EXTERNAL instruction imports (in purple) the *payment card* definition from another regulation, NV 205.602, into NV 603A.215(5)(d). The INCLUDE instruction maps (in blue) the definitions from 603A.215(5), including any imported definitions, onto 603A.215; this mapping includes the inner link from *data storage device* to *facsimile*, and the outer links to requirements in 603A.215(1) and (2). Last, the exemption 603A.215(4)(a) is mapped (in red) onto requirements 603A.215 to exclude interpretations implied by definitions.

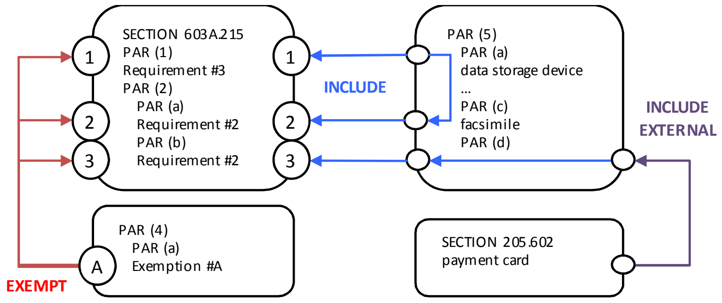


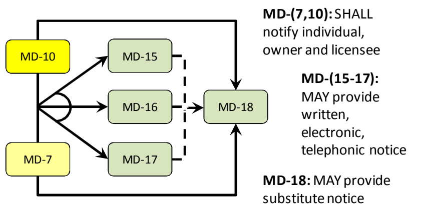
Fig. 9. Summarizing the Effects of Conditionality

4.2 Regulatory Specification Patterns

When visualized graphically, the LRSL-encoded regulations reveal several regulatory specification patterns. Visual specifications have been hypothesized to improve requirements comprehension [12]. These patterns describe legal mechanisms for prescribing the behavior of personnel and systems in the environment. In Figure 4, we presented the first pattern, called a *suspension*, in which a permission (AR-10) is an exception to an obligation (AR-7) and satisfying the pre-conditions of the permission causes the obligation to be suspended. We now discuss three other patterns: system design alternatives and scaling restrictions; standards and indemnification; and limited exceptions for legacy systems. We believe these patterns can be re-used in writing new regulations and standards or for identifying similar dependencies among requirements.

Figure 10 shows three system design options for sending written (MD-15), electronic (MD-16) and telephonic (MD-17) notices as means for notifying individuals, data owners and data licensees of a security breach under MD §14.3504(e); note the arc indicating the “or” relationship between these options means only one option is necessary to discharge the obligations MD-10 and MD-7. These alternatives are intended to allow businesses to leverage a diverse set of contact options based on the level of technological sophistication of the business. In addition, the exception MD-18 permits a substitute notice via statewide media and other broadcast mechanisms, when the cost of notification becomes too prohibitive. This type of scaling mechanism (a permitted exception conditioned on measurable limits of effect size, in this case a finite number of notices or monetary value) can be used to control regulatory system costs across an entire industry.

Figure 11 shows the combined uses of deference to external standards with indemnification from NV §603A.215. The Payment Card Industry Data Security Standard (PCI-DSS), cited in NV-5, prescribes several technical security requirements for businesses that handle payment cards. In Figure 11, a business is prohibited (in red) from transferring data (NV-6) or moving data storage devices (NV-7), excluding facsimiles. However, complying with the PCI-DSS standard (in yellow, NV-5) is an exception that permits transferring data and moving devices. Whether a business chooses to accept the more prohibitive restrictions or to comply with the exception, NV §603A.215 prohibits the business from being liable for data breach damages. This prohibition is an example of a *safe harbor*, which is a regulatory mechanism designed to encourage industry to act against uncertainty (the uncertain costs of data breach damages vs. the more certain and predictable costs of PCI-DSS compliance).


MD-10
MD-7
MD-15
MD-16
MD-17
MD-18
MD-(7,10): SHALL notify individual, owner and licensee
MD-(15-17): MAY provide written, electronic, telephonic notice
MD-18: MAY provide substitute notice

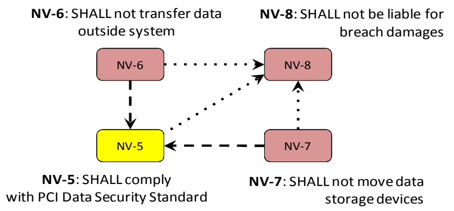

NV-6: SHALL not transfer data outside system
NV-8: SHALL not be liable for breach damages
NV-6
NV-8
NV-5
NV-7
NV-5: SHALL comply with PCI Data Security Standard
NV-7: SHALL not move data storage devices

Fig. 10. System Design Alternatives and Scaling **Fig. 11.** External Standards and Indemnification Restrictions

In Figure 12, the State of Vermont describes a set of prohibitions (in red, VT-40 through VT-43) on the use of Social Security Numbers (SSNs) of Vermont residents. In the United States, SSNs are issued by the government for tracking government-sponsored pensions, but have over time been used to track individuals for other purposes, such as health benefits and credit-based services, including cellular telephones, utilities, loans and credit cards. Because of the prevalent and historic use of SSNs to authenticate and identify individuals, VT §62.2440(c)(8)(A) includes an exception, which permits (in green) continuous use of SSNs to accommodate legacy systems. Continuous use includes the follow-on obligations (in yellow) to notify residents about such use (VT-48) and provide the option to halt such use (VT-49). Such exceptions provide businesses with the ability to scale their business practices to a new standard of care based on individual consumer preferences over time.

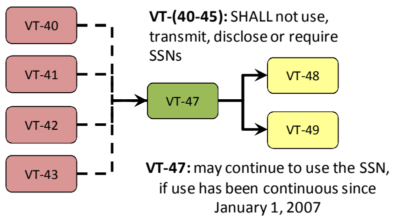

VT-40
VT-41
VT-42
VT-43
VT-(40-45): SHALL not use, transmit, disclose or require SSNs
VT-47
VT-48
VT-49
VT-47: may continue to use the SSN, if use has been continuous since January 1, 2007

Fig. 12. Limited Exceptions for Legacy Systems

As technology evolves, we foresee increasingly design-invasive regulations that can potentially limit the range of solutions available to a designer. Thus, we believe patterns such as these should be part of the requirements nomenclature, to aid businesses in understanding the scope and implication of regulations on system design.

5 Threats to Validity

In grounded analysis, multiple analysts derive theoretical constructs from a dataset to describe or explain the data; these insights only generalize to that dataset [10]. Because we selected a single theme (data breach notification), our theory may not be externally valid in other domains, such as medical devices or aviation. However, we did validate the schema notation and document model by systematically inspecting data breach notification laws in all 46 U.S. states and territories, two U.S. Federal regulations (HIPAA Privacy Rule and the Section 508 Access Standards), the European Union Directive 95/46/EC and a Canadian privacy law (PIPEDA). We found the schema sufficiently robust to model these documents and their cross-references.

Construct validity is the correctness of operational measures used to collect data, build theory and report findings [35]. To improve construct validity, we maintained a *caveats list* of translation strategies that reflect unusual cases and how the parser should treat such cases, and a *proposed changes list* of requirements with examples for new language constructs. As new constructs were introduced, we reviewed each previously encoded law to update the translation to reflect the new construct to ensure consistency across the translated datasets. In addition, we developed analytic tools using the parser and a research database to collect all the statistics reported, here.

Internal validity is the extent to which measured variables cause observable effects within the data [35]. Our results show that writing styles can positively or negatively impact reference ambiguity and ambiguity loss, as measured by our LRSL translation presented in Table 1. New research is needed to evaluate if these styles affect an analyst's ability to resolve cross-references and locate relevant requirements.

External validity describes the extent to which a theory generalizes. While two investigators have applied the LRSL in 13 cases, further evaluation is needed to know to what extent others can apply the language with the same effects and to what extent the language is complete.

Reliability describes the consistency of the theory to describe or explain environmental phenomena over repeated observations [35]. To improve reliability, both investigators (the authors) separately translated the datasets into the LRSL and compared their results to identify alternate modes of expression and language caveats.

6 Discussion and Summary

In this paper, we introduce a legal requirements specification language (LRSL) for codifying legal requirements with typed cross-references. In Section 4, we show how the LRSL can be used to shape conditionality of regulatory coverage, which is enabled by the tool-supported ability to trace definitions across a single regulation, or across multiple regulations as definitions are shared across laws. Reusing technical terminology improves requirements engineering practices, as it avoids misconceptions among stakeholders and competing viewpoints that introduce inconsistency into design specifications [33]. Zave and Jackson have noted the importance of grounding terminology in the reality of the environment to which a machine will be built [37]. Increasingly, this includes the *legal reality* as software systems contribute to social and environmental hazards and regulators attempt to shape the outcome of automation by defining legal boundaries that limit the behavior of information systems and software-supported practices. By systematically tracing and encoding legal terms, constraints and requirements, we believe the LRSL can aide engineers at design time to manage this changing reality, while also supporting users who are responsible for deployment and maintenance. The work to write better software requirements and realize how software satisfies a particular legal constraint, however, is still outside our findings.

In addition, we discovered several regulatory requirements patterns that become visually salient and enable measuring different styles of regulatory document construction. These patterns, described in Section 4.3, include strategies for pairing permitted refinements to the obligations that they refine to create design alternatives that allow organizations to scale their information practices over time. A similar pattern invokes prohibitions with limited exceptions to accommodate legacy systems: this pattern effectively expires the legacy system as the exceptions are discharged over the life of the new system. Finally, a third pattern uses indemnification to encourage design changes to accommodate increased security. We envision requirements analysts using these patterns in several ways. First, analysts may be trained to identify these and similar patterns from the LRSL-generated graph. Identifying these patterns can help analysts see higher-order constructs, such as temporary suspensions of duties and legal indemnification. Second, these patterns can be used to compare and contrast regulatory mechanisms across regulations: indemnification is an incentive to reduce legal liability, whereas design alternatives are a legal means to accommodate variation in practices. Because we only observed these patterns in a few cases, however, further evidence must be collected to understand the extent to which regulators reproduce these patterns. That said, the LRSL's ability to transform the encoded regulatory specifications into corresponding graphs enables visualizing this higher-order information and provides analysts with access to this regulatory information described in the regulation.

The LRSL only begins to address a small part of the larger problem, however. Laws include statutes that govern regulatory agencies, regulations created by those agencies to govern industry, and informal agency guidance intended to help companies interpret laws. In addition, court proceedings describe judicial interpretations of regulations. While the LRSL is not a legal document, it provides an intermediary artifact that legal and requirements analysts can use to engage in discussing compliance strategies. These discussions may link legal opinion and context to the LRSL-generated artifacts as a means to preserve rationale and enable traceability.

We further envision the LRSL capabilities as enabling document authors to design and debug specifications, to remove ambiguity and organize requirements around central themes. The LRSL's ability to reuse and extend definitions and link to regulatory rules across multiple regulations supports our vision of requirements as open, dynamically evolving systems, wherein the discovery of conflicts becomes increasingly critical to creating regulatory harmony. Finally, the LRSL parser supports several features that can be used to "debug" regulatory specifications, by identifying cycles in cross-references, definitions for terms not used in the regulation, and possible conflicts or contradictions through visual inspection of the generated graphs. We believe these techniques can benefit both regulators who write regulations as well as requirements engineers and software designers who seek to understand the regulation and seek guidance from their corporate legal compliance office. We found the time required to translate the regulations into the LRSL well worth the ability to debug and analyze the relations using the LRSL-generated model.

Acknowledgment. This research was supported by the U.S. Department of Homeland Security (Award #2006-CS-001-000001), Hewlett-Packard Labs (Award #CW267287) and ONR Award #N00244-12-1-0014.

References

[1] Allen, L.E., Saxon, C.S.: Better language, better thought, better communication: the a-hohfeld language for legal analysis. In: 5th Int'l Conf. AI & Law, pp. 219–228 (1995)

[2] Biagioli, C., Mariani, P., Tiscornia, D.: ESPLEX: A rule and conceptual model for representing statutes. In: Proc. 1st Int'l Conf. AI & Law, pp. 240–251 (1987)

[3] Bourcier, D., Mazzega, P.: Toward measures of complexity in legal systems. In: Int'l Conf. AI & Law, pp. 211–215 (2007)

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

[5] Breaux, T.D., Antón, A.I., Doyle, J.: Semantic parameterization: a process for modeling domain descriptions. ACM Trans. Soft. Engr. Method. 18(2), 5 (2008)

[6] Breaux, T.D., Vail, M.W., Antón, A.I.: Towards compliance: extracting rights and obligations to align requirements with regulations. In: 14th IEEE Int'l Req'ts Engr. Conf., pp. 49–58 (2006)

[7] Breaux, T.D.: Exercising due diligence in legal requirements acquisition: a tool-supported, frame-based approach. In: IEEE 17th Int'l Req'ts Engr. Conf., pp. 225–230 (2009)

[8] Breaux, T.D.: Legal requirements acquisition for the specification of legally compliance informaiton systems, North Carolina State Univ. Ph.D. thesis (2009)

[9] Bench-Capon, T.J.M.: Deep models, normative reasoning and legal expert systems. In: Proc. 2nd International Conference on Artificial Intelligence and Law, Vancouver, British Columbia, Canada, pp. 37–45 (1989)

[10] Corbin, J., Strauss, A.: Basics of Qualitative Research, 3rd edn. Sage Pubs (2008)

[11] Dardenne, A., Fickas, S., van Lamsweerde, A.: Goal–directed requirements acquisition. Sci. Comp. Prog. 20, 3–50 (1993)

[12] Dulac, N., Viguier, T., Leveson, N., Storey, M.-A.: On the use of visualization in formal requirements specification. In: IEEE Joint Int'l Conf. Req'ts Engr., pp. 71–80 (2002)

[13] Fraser, M.D., Kumar, K., Vaishnavi, V.K.: Informal and formal requirements specification languages: bridging the gap. IEEE Trans. Soft. Engr. 17(5), 454–466 (1991)

[14] Fuxman, A., Liu, L., Mylopoulos, J., Pistore, M., Roveri, M., Traverso, P.: Specifying and analyzing early requirements in Tropos. Req'ts Engr. Journal 9(2), 132–150 (2004)

[15] Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Modeling security requirements through ownership, permissions and delegation. In: IEEE 13th Int'l Req'ts Engr. Conf., pp. 167–176 (2005)

[16] Greenspan, S., Mylopoulos, J., Borgida, A.: On Formal Requirements Modeling Languages: RML Revisited. In: 6th IEEE Int'l Soft. Engr. Conf., pp. 1–13 (1994)

[17] Glinz, M., Berner, S., Joos, S.: Object-oriented modeling with ADORA. Info. Sys. 27, 425–444 (2002)

[18] Hohfeld, W.N.: Some fundamental legal conceptions as applied in judicial reasoning. The Yale Law Journal 23(1), 16–59 (1913)

[19] Lauritsen, M., Gordon, T.F.: Toward a general theory of document modeling. In: Int'l Conf. AI & Law, pp. 202–211 (2009)

[20] Levene, A.A., Mullery, G.P.: An investigation of requirement specification languages: theory and practice. IEEE Computer 15(5), 50–59 (1982)

[21] Massey, A.K., Anton, A.I.: Triage for legal requirements. NCSU Technical Report #TR-2010-22 (October 11, 2010)

[22] Maxwell, J., Anton, A.I.: Developing production rule models to aid in acquiring requirements from legal texts. In: IEEE 17th Int'l Req'ts Engr. Conf., pp. 101–110 (2009)

[23] Maxwell, J., Anton, A.I., Swire, P.: A legal cross-references taxonomy for identifying conflicting software requirements. In: IEEE 19th Int'l Req'ts Engr. Conf., pp. 197–206 (2011)

[24] Martinek, J., Cybulka, J.: Dynamics of legal provisions and its representation. In: Int'l Conf. AI & Law, pp. 20–24 (2005)

[25] Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Computing Surveys 37(4), 316–344 (2005)

[26] Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M.: Telos: representing knowledge about information systems. ACM Trans. on Info. Sys. 8(4), 325–362 (1990)

[27] Romanosky, S., Telang, R., Acquisti, A.: Do data breach disclosure laws reduce identity theft? In: W'shp Econ. of Info. Sec. (WEIS), June 25-28 (2008)

[28] Rubinstein, I.: Privacy and Regulatory Innovation: Moving Beyond Voluntary Codes. I/S: A Journal of Law and Policy for the Information Society (April 2011) (in press)

[29] Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T.: The British Nationality Act as a logic program. Communications of the ACM 29(5), 370–386 (1986)

[30] Sergot, M.: A computational theory of normative positions. ACM Transactions of Computational Logic 2(4), 581–622 (2001)

[31] Siena, A., Jureta, I., Ingolfo, S., Susi, A., Perini, A., Mylopoulos, J.: Capturing variability of law with Nomós 2. In: 31st Int'l Conf. Conc. Mod., pp. 383–396 (2012)

[32] Stamper, R.K.: LEGOL: Modelling legal rules by computer. In: Proc. Advanced Workshop on Computer Science and Law, pp. 45–71 (September 1979)

[33] Wasson, K.S.: A case study in systematic improvement of language for requirements. In: Proc. IEEE 14th Int'l Req'ts Engr. Conf., pp. 6–15 (2006)

[34] Winkels, R., Boer, A., de Maat, E., van Engers, T., Breebaart, M., Melger, H.: Constructing a semantic network for legal content. In: Int'l Conf. AI & Law, pp. 125–132 (2005)

[35] Yin, R.K.: Case study research, 4th edn. Applied Social Research Methods Series, vol. 5. Sage Publications (2008)

[36] Yu, E.: Modeling organizations for information systems requirements engineering. In: Int'l Symp. Req'ts Engr., pp. 34–41 (1993)

[37] Zave, P., Jackson, M.: Four dark corners of requirements engineering. ACM Trans. Soft. Engr. & Method. 6(1), 1–30 (1997)