Addressing Legal Requirements in Requirements Engineering

Paul N. Otto and Annie I. Antón
Department of Computer Science
North Carolina State University
{pnotto, aianton}@ncsu.edu

Abstract

Legal texts, such as regulations and legislation, are playing an increasingly important role in requirements engineering and system development. Monitoring systems for requirements and policy compliance has been recognized in the requirements engineering community as a key area for research. Similarly, regulatory compliance is critical in systems that are governed by regulations and law, especially given that non-compliance can result in both financial and criminal penalties. Working with legal texts can be very challenging, however, because they contain numerous ambiguities, cross-references, domain-specific definitions, and acronyms, and are frequently amended via new regulations and case law. Requirements engineers and compliance auditors must be able to identify relevant regulations, extract requirements and other key concepts, and monitor compliance throughout the software lifecycle. This paper surveys research efforts over the past 50 years in handling legal texts for systems development. These efforts include the use of symbolic logic, logic programming, first-order temporal logic, deontic logic, defeasible logic, goal modeling, and semi-structured representations. This survey can aid requirements engineers and auditors to better specify, monitor, and test software systems for compliance.

1. Introduction

The need for system developers to monitor systems for both requirements and policy compliance has been identified as a challenging and important problem in the requirements engineering community [41]. In fact, according to a survey of nearly 1,200 senior information security professionals, compliance has been the primary driver of information security policy for the past two years [17]. Requirements engineers, developers, and auditors currently face two major problems in assessing legal compliance: (a) determining the applicable regulations, and (b) creating the policies necessary to achieve compliance with those regulations [24]. Methodologies for monitoring compliance with requirements and policies are currently not available to developers [41]. And yet, stakeholders need to better understand the regulations that govern the systems for which they are responsible and they require precise answers to specific queries about what is allowed and what is not allowed [4, 35].

For requirements engineers, access to specific laws and regulations has become easier with the push towards online access to all government legislation and regulations. However, an organization must still identify the regulations relevant to its specific system before it can even begin to assess its compliance with the law. Once the relevant regulations are identified, extracting requirements from legal texts is still a difficult and error-prone process [46]. In addition, an organization must still engage in traditional software engineering activities (e.g. analysis, modeling, development) as well as traditional security activities (e.g. policy enforcement and auditing) in order to properly implement compliance processes [14].

This paper surveys research efforts over the past 50 years in modeling and using legal texts for system development. Our survey identifies the strengths and weaknesses of each approach, and based on our analysis of the literature to date as well as our own prior experiences in analyzing policy and regulations [13, 19, 37], we propose a broad set of requirements for tool support that would aid requirements engineers and compliance auditors. It is our hope that these requirements will prompt serious consideration by the requirements engineering community, as it is within this community that we believe significant progress can be made to address the challenges related to legal compliance in software systems.

The remainder of this paper is organized as follows. Section 2 discusses the nature of regulations, noting the various characteristics that make legal texts difficult to work with. Section 3 analyzes various efforts from the past 50 years in modeling regulations, extracting key concepts, and using legal texts in system development. Based on our extensive review of prior work, Section 4 proposes a set of broad requirements for a comprehensive system to assist requirements

1090-705X/07 $25.00 © 2007 IEEE
DOI 10.1109/RE.2007.65

engineers and auditors with regulatory compliance tasks. Finally, Section 5 discusses our analysis and outlines future work needed to realize such a system.

2. The Nature of Regulations

There are certain characteristics of regulations that make them both useful and difficult to apply to design methodologies. Regulations tend to be very structured and hierarchical documents. However, agencies at the federal, state, and local level can all specify new regulations, and these regulations may complement, overlap, or even contradict one another due to differing objectives and changes over time [25]. As a result, some areas of law undergo constant changes, whereas other areas are relatively stable [6]. In addition, amendments and revisions to the same piece of regulation can lead to internal contradictions [4].

Depending on the field of law under consideration, there may also be the complicating influence of case law. Prior research has noted the coexistence of two forms of law: statutory law, or the specific regulations in force; and case law, or the interpretation of those rules by the courts [43]. The amount and influence of case law on any given regulation varies widely. Some areas of law, such as tax law, are well-settled and have a large body of case law; as such it is possible to classify most cases as 'routine' [21]. Other areas, such as information security and data privacy law, are still emerging fields and are therefore subject to greater fluctuation in the law's requirements. Regulations in these fields are relatively new and, as a result, very little case law exists to guide requirements engineers in interpreting the law.

In addition to case law, regulations are often accompanied by other guiding documents on how to interpret and use the law. Such supplemental references may include previous administrative rulings, reference handbooks, or other published guides to interpreting the regulation [24]. The ambiguity associated with regulations has forced government agencies to provide these detailed reference materials and instructive handbooks to aid understanding and compliance efforts [28]. For example, the U.S. Department of Health and Human Services publishes a summary of the Health Insurance Portability and Accountability Act (HIPAA)[1] Privacy Rule and guidance documents for implementing the HIPAA Security Rule. Some of these supplemental guides are created by organizations separate from the government agencies that actually promulgated the regulations [29]. This large, diverse set of documentation can be crucial for software developers who are attempting to identify regulatory compliance requirements early in the design process. However, requirements engineers must be careful when using these supplemental documents, as they do not have the same legal standing and may even contain misinterpretations of the original regulatory text.

Another important characteristic of regulations is the frequent references to other sections within a given legal text and even to other pieces of law. Much of the prior work in computer science that examines regulations has noted the difficulty of handling these numerous cross-references within regulations (e.g. [7, 13, 24]). These cross-references force requirements engineers to spend additional time reading and understanding legal texts, before they can even begin to extract key concepts or apply the regulations to system design. May et al. employ a methodology to derive formal models from regulations that they applied to the HIPAA Privacy Rule [33]. In their study, discussed in Section 3.6, they assume that external and ambiguous references are satisfied by default [33]. This contradicts our own study of the HIPAA Privacy Rule [13], discussed in Section 3.3, in which we discovered that cross-references introduce important constraints from other sections that restrict which rules apply in different situations and contexts.

If references to other sections of a particular regulation or other external laws are unaccounted for, requirements engineers are prone to make interpretations and inferences that are inconsistent with the law. Such assumptions will inevitably lead to overlooking important exceptions or priorities and ultimately to non-compliance. Traceability within the context of regulatory systems takes on a far greater significance than we already afford it in the requirements engineering community because legal traceability is supercharged, so to speak, with priorities and exceptions that govern special cases (e.g. which information can be accessed, when such access is allowed). Thus, the ability to manage cross-references and maintain traceability from the originating law, regulation, or policy to the relevant requirements must be addressed in any system for supporting requirements engineers and/or compliance auditors.

Regulations typically specify a large number of relevant definitions and acronyms, further complicating the job of requirements engineers and system designers [24]. Along with cross-references, such extensive definitions necessitate a significant amount of domain knowledge before the regulations are comprehensible and usable. When spread across multiple regulations that may have overlapping, inconsistent, or contradictory terms, the domain-specific lexicon significantly raises the barrier to entry

[1] Health Insurance Portability and Accountability Act of 1996, 42 U.S.C.A. 1320d to d-8 (West Supp. 1998).

for developers hoping to build regulatory compliance into their software systems.

A more fundamental problem in dealing with regulations is the fact that regulations and law are laden, often by design, with ambiguities. For example, §164.306(a)(2) in HIPAA requires organizations to "protect against any reasonably anticipated threats or hazards to the security or integrity" of protected health information; the section does not define what constitutes reasonable anticipation. Researchers from our literature survey frequently note the difficulty in identifying and resolving such ambiguities in legal documents (e.g. [1, 13, 29, 43]).

Researchers have examined the problem of ambiguity in natural language as it affects requirements engineering. Kamsties characterizes ambiguity as more problematic than other forms of defects that appear in requirements [22]. Kamsties et al. discuss various types of linguistic and requirements engineering-specific, context-dependent ambiguities [23]. They also note that using formal methods to remove ambiguities from natural language representations simply results in an unambiguously wrong specification [23]. Moreover, such formal specifications are inaccessible to the majority of stakeholders and their formality makes it more difficult to discover ambiguities than in a natural language representation [23]. Berry et al. present a guide to disambiguating text for requirements engineers and lawyers alike [8].

There are several different classifications of ambiguities. A simple dichotomy of ambiguities is those that are intentional — to allow the law to be generalized — and those that are unintentional [1]; the example from §164.306(a)(2) likely represents an intentional ambiguity. Additional categorizations include: high-level classifications (e.g. implication-coimplication, disjunctive-conjunctive, ambiguity of reference) [1]; categorization within specific domains, such as software engineering, linguistics, and law [8]; a taxonomy of ambiguities occurring in requirements [22]; a distinction between nocuous and innocuous ambiguities [15]; and specific types of ambiguities uncovered during empirical analysis (e.g. conjunctions, under-specifications) [13]. Just as courts must struggle to interpret the law when ambiguities are present, so must users, be they requirements engineers or developers, make crucial interpretation decisions during requirements gathering and software design.

3. Survey of Work with Regulations

We now examine various approaches for modeling regulations, extracting key concepts from legal texts, and creating compliance checking systems. In our survey, we followed the principles of systematic review as detailed by Kitchenham [26]. Our research question for the survey is, "What efforts have been made to model legal texts for use in requirements engineering and system development?" Due to space limitations, our methodology is not fully disclosed in this paper. In short, we first identified potentially relevant research through use of the ACM and IEEE databases, then followed citations to uncover additional papers for review; this resulted in the discovery of over 150 relevant publications. After careful analysis and review, we included 38 papers in the following discussion, grouped into nine categories.

3.1. Symbolic Logic

One of the earliest attempts to model legislation involved the use of symbolic logic, also known as mathematical logic. The approach attempted to balance the benefits of natural language with the rigor of symbolic logic [1], serving as a precursor for later efforts to provide both human- and machine-readable interpretations. Allen's technique employed six key logical connectives: implication, conjunction, coimplication, exclusive disjunction, inclusive disjunction, and negation [1]. By identifying the logical connectives, one could largely eliminate the unintended ambiguities present in legislative texts by using a more mathematical representation. This effort, while noteworthy in its systematic legal representation, did not leverage the processing and data manipulation capabilities of computers; it sought to answer specific queries and make legalistic determinations, rather than shape requirements gathering or systems development.

3.2. Knowledge Representation

Numerous approaches to representing legal text as computer programs began in the late 1970s, largely based on logic programming techniques. These knowledge representation efforts were based on the premise that a model of legal texts should closely parallel the language of those texts [10]. As such, most of these approaches used Prolog — a logic programming language targeted for knowledge representation and expert systems — to represent the legal rules extracted from laws and regulations. Specific efforts included: TAXMAN, modeling the United States Internal Revenue Code [34]; representing the British Nationality Act as a logic program [43]; modeling the Income Tax Act of Canada [44]; representing the United Kingdom welfare law as a logic program [7]; ESPLEX, a logic system for representing legal rules [10]; capturing the Indian Central Civil Service Pension Rules in logic [42]; and modeling the Dutch Opium Act [39]. Each of these knowledge representation techniques would aid requirements engineers in understanding legal texts and answering specific queries during requirements elicitation.

Knowledge representations of legal texts afford certain advantages to system developers and policymakers alike. Logical representations enable users to identify unintended ambiguities in the text [43]. This allows requirements engineers to pinpoint specific ambiguities and resolve those issues before system development commences. It allows policymakers to address these ambiguities in future amendments to the law. Developers can use expert systems to make specific queries when issues arise regarding compliance or design decisions. Such targeted queries enable developers to resolve known compliance issues with the relevant regulation(s).

Several characteristics of these systems limit the generalizability or applicability of this research to current regulations. The knowledge representation approach has mainly focused on either well-settled areas of law or regulations with minimal accompanying case law. Most of the projects considered themselves case studies and, to the best of our knowledge, no final product or working system ever resulted from the research. The goal was often to answer specific queries or handle what-if scenarios; none of these early efforts used the model to influence system development or check for compliance. These knowledge representation approaches had no degree of automation: for each new piece of regulation, the user would be required to manually extract the legal rules and encode them in logical clauses. Finally, the research efforts referenced above, with the exception of [39], make no mention of providing traceability between the representation and the original legal text. As previously discussed, this lack of traceability creates compliance vulnerabilities as the law evolves via case law or new regulations. These drawbacks combine to make knowledge representation techniques problematic and very limiting for software developers who need to extract requirements and system design elements directly from regulations.

A more recent variation on logic programming efforts employs event calculus to track the changes in legal texts over time [31]. The approach uniquely captures the frequent changes associated with legal texts, enabling users to model and understand how the law changed across revisions [31]. Martinek and Cybulka create a knowledge base maintaining information for when changes are made to regulatory texts [31]; this provides a limited measure of traceability for developers evaluating changes over time. The approach provides a unique look at the dynamic nature of legal texts, but does not address the same aforementioned shortcomings facing other logic programming implementations.

3.3. Deontic Logic

Another logic-based approach to modeling regulations involves the use of deontic logic to capture the rights and obligations present in the law. The impetus for this approach is that "the law is like a programming language controlling a society … [where] observations must be made, calculations performed, records kept and messages transmitted" [45]. Extracting specific rights and obligations from legal rules permits the creation of a knowledge base, as was possible with the logic programming efforts, to model the key elements of regulations and answer directed user queries. The major deontic logic efforts include: LEGOL, a formal LEGally Orientated Language for capturing obligations [45]; ON-LINE, an ONtology-based Legal INformation Environment for capturing and analyzing legal texts as legal knowledge [47]; work establishing the legal importance of monitoring permissions as well as obligations [11]; and systems for automated extraction of normative references from legal texts [9, 38].

Deontic logic approaches have not yet met users' needs for working with regulations and ensuring compliance. By extracting the rights and obligations, deontic logic systems disambiguate regulations and make them more palatable for system designers. Early work established the utility of such an approach, but the user was still required to manually encode the law into the deontic operators for rights and obligations [45, 47]. The ON-LINE system was able to deal with only small sections of legislation at a time and the usability of the ontology-based approach proved problematic during usability testing [47]. More recent efforts include automated extraction of normative references, such as specific rights and obligations, detailed in a legal text, and addressed the problem of the law's evolution by tracking changes over time [9, 38]. This provides for some degree of traceability, as the system maintains information on each extracted section including its type, number, date, section and subpart headers, and the normative references [38]. However, these more recent projects were not completed, and there are few examples to illustrate the effectiveness of this approach. While these research efforts established deontic logic as a worthwhile approach to extract key information from regulations, they did not result in usable tools for developers to influence system design or monitor compliance.

A more recent deontic logic implementation involves the explicit extraction and balancing of rights and obligations from regulations [13]. The research focuses on providing requirements monitoring and compliance support for system developers and maintainers [12]. Semantic parameterization entails identifying the ambiguities within a legal text and balancing the extracted rights and obligations [12]. This decomposition of regulations enables the user to identify both explicit and implied rights and obligations [12]; capturing these implied rights and obligations is not addressed by the other deontic logic approaches. The process, however, requires manual extraction of the rights, obligations, delegations, and constraints. Unlike most other approaches, Breaux and Antón maintain traceability across all artifacts from the HIPAA section and paragraph number to the corresponding software requirements and access control rules. This approach has been tested on only a part of the HIPAA Privacy Rule; as such, its scalability and applicability to other domains is not yet validated.

3.4. Defeasible Logic

Defeasible logic provides an alternative logic-based approach to modeling regulations. Defeasible logic is a form of non-monotonic skeptical reasoning, wherein there are strict rules, defeasible rules, and defeaters. Strict rules always hold, while defeasible rules hold true unless an exception, or defeater, exists for the rule. Given the existence of overlapping and conflicting legal texts at different levels of government, defeasible logic appears to be a natural fit for modeling regulations [3]. The practical use of defeasible logic in routine legal practice is emphasized as a key advantage for system developers and users of regulations [21]; defeasible logic can aid in both decision support and legal reasoning [4].

Proponents of defeasible reasoning have also noted that deontic logic will not capture all eight fundamental legal conceptions [20]: right, no-right, privilege, duty, power, disability, immunity, and liability [18]. Hohfeld presented these fundamental legal conceptions as the basic elements needed to understand any legal relation, noting specifically that 'rights' and 'duties' (obligations) were insufficient to address the complexities in many areas of law [20].

Antoniou et al.'s approach has yielded an operational implementation of a defeasible logic system [3], but there remain several disadvantages to such an approach for modeling regulations and monitoring compliance. For example, numerous features need to be added to any 'pure' defeasible logic implementation (e.g. representing hierarchies, arithmetic and temporal operators, and capturing underlying legal knowledge) to model all the nuances of the law [3]. The computational complexity of a defeasible logic system is in dispute: early research touted low complexity as a major advantage [4], whereas more recent research indicated that approximating a model was necessary due to concerns about complexity [21]. Again, these efforts in defeasible logic make no mention of maintaining traceability and provide no examples of directly modeling regulations. Given the lack of follow-up on defeasible logic approaches, the viability of such systems remains uncertain. There is currently no system available to leverage defeasible reasoning in requirements engineering and compliance monitoring.

3.5. First-Order Temporal Logic

Barth et al. proposed using first-order temporal logic to extract key concepts — context, roles, type of information — rather than precisely modeling the regulation [5]. The approach, which is based on the conceptualization of privacy using the contextual integrity framework [36], captures only the privacy-related elements of regulations such as parts of HIPAA [5]. The use of formal logic is reminiscent of other logic-based approaches, but the narrower focus on privacy limits the applicability of this approach to other regulations. Preliminary results show that the contextual integrity framework captures most privacy elements from the regulations tested to date; however, Barth et al. do not disclose what percentage of privacy elements originally present in the legal text were extracted using their framework [5].

The research establishes the framework's viability in assessing compliance between privacy policies and the privacy provisions of regulations. However, a major limitation of this approach for the requirements engineer is that Barth et al. make no mention of maintaining traceability between the extracted concepts and the original regulatory text. Although this approach may be capable of aiding developers in evaluating system requirements and design against privacy regulations, its narrow framework does not appear to extend to other legal texts. Unlike many of the earlier research projects discussed in this section, this framework may soon be available to other researchers for validation and extension.

3.6. Access Control

Another approach to modeling regulations employs access control techniques to capture the privacy-related elements of legal texts. May et al. propose an "auditable privacy system" that includes conceptualizations for transfer, actions, creation, rights establishment, notification, and logging [33]. Leveraging the similarity between legal privacy texts and APIs in specifying rules on accessing protected information, they derive privacy-focused mandatory access control rules directly from regulations [33]. This translation into access control rules captures regulatory conditions and obligations as allow/deny operations. Those conditions and obligations that cannot be represented as access control rules are cast instead as external environmental flags [33].

The auditable privacy system implementation fulfills some key requirements engineering tasks, but its narrow focus keeps it from adequately supporting the complex needs of requirements engineers working with legal texts. May et al. use a modeling language to represent legal texts and privacy policies, thus enabling model checking and verification operations. Such formalism supports queries on the regulatory model, so that developers and policymakers alike can analyze a given text and evaluate compliance and design issues [33]. However, their regulatory model abstracts away many key aspects and characteristics of legal texts; for example, the assumption that external and ambiguous references are satisfied by default. In addition, the model omits many low-level system requirements (e.g. password procedures) specified by HIPAA [33]. The narrow privacy focus, coupled with the inadequate support for key elements of legal texts, makes this approach unsatisfactory for requirements engineers who need to extract requirements from legal texts and monitor compliance.

3.7. Markup-Based Representations

Given the hierarchical nature of legal texts, some researchers are attempting to capture regulations with semi-structured markup languages, such as Standard Generalized Markup Language (SGML) and Extensible Markup Language (XML). Such markup-based representations can mimic the structure of regulations and also maintain annotations and other metadata regarding each section, part, or even sentence of the original legal text [24]. A markup-based representation also enables the system to easily capture and display information on definitions, acronyms, and cross-references within the regulation(s), thereby addressing several of the key requirements for using legal texts during system development. A semi-structured representation can be combined with well-established information retrieval techniques and first-order predicate logic to aid users in both locating and analyzing relevant regulation sections [29]. In addition, some newer legal texts are already being represented in XML; augmenting these existing representations is a relatively easy task [35]. Research efforts in this area include: SGML modeling of decisions of the Supreme Court of Canada [40]; REGNET, an XML framework for representing regulations [24, 25, 28, 29, 30]; and an overview of several XML models for representing legal texts [35].

Markup-based representations hold promise for providing requirements engineers with the necessary framework for leveraging regulations in system development. The work in SGML was an isolated effort now superseded by research utilizing XML, a simplified derivative of SGML that is easier to process. The REGNET project, based on an XML framework, has generated over 25 published papers describing the system and its use in tasks such as: representing regulations [24], providing similarity analysis between different regulations [28], and helping policymakers in drafting new regulations [30]. The REGNET project includes a parser to automatically transform regulations into XML and uses other tools to semi-automatically generate conceptual tags for the markup [24]. REGNET provides the foundation for verifying compliance with a specific regulation, but has been tested only in limited domains and the prototype system is not yet available to other researchers. In addition, in its current form REGNET does not provide a precise model of the regulations [24].

Finally, the research evaluating several different markup-based approaches does not provide details on the underlying representations; instead it focuses on techniques for ranking the different XML models being reviewed [35]. Thus, while markup-based approaches benefit from mimicking the hierarchical, semi-structured nature of regulations, previous research approaches do not offer developers any available tools to shape requirements engineering and design efforts around regulatory compliance. The REGNET prototype system shows the most promise in assisting with compliance efforts, but comparing and drafting regulations, rather than extracting system requirements, has become the main focus of this work.

3.8. Goal Modeling

The SecureTropos approach, based on the i* framework, involves extracting and representing the goals, soft goals, tasks, resources, and social relationships for defining obligations [32]. It then uses these concepts to model the relationships for actors, dependencies, trust, delegation, and goal refinement [32]. SecureTropos has been used to assess a university's compliance with the Italian Data Protection Act [32]. Whereas the focus of the research is on applying requirements engineering principles to security requirements, the broader context examines how an organization can assess its compliance with standards from a particular regulation.

The SecureTropos approach requires a manual extraction of the concepts. As with previously discussed approaches, traceability is not addressed, and we have yet to find any examples of the mapping between the extracted concepts and their presence in the original regulation. SecureTropos may enable developers to better design systems to be compliant with the fundamental concepts of a specific security regulation, but its scalability and applicability to a broader range of legal texts is as yet unproven. Finally, SecureTropos does not currently provide users with the ability to answer specific legal queries or identify changes in the law over time.

3.9. Reusable Requirements Catalog

Toval et al. recently created a reusable catalog of legal requirements that were derived from specific legal texts regarding security and personal data protection [46]. The Personal Data Protection (PDP) Catalog enables requirements engineers to incorporate legal requirements into the development lifecycle and build compliance into new systems [46]. By providing reusable legal requirements, analysts can more easily uncover ambiguities and inconsistencies, and the quality of the catalog increases with each usage [46].

This initial foray into applying requirements engineering methodologies to legal requirements provides some interesting insight, but does not satisfy the comprehensive set of requirements engineering needs that we address in this paper. For example, Toval et al. highlight traceability as particularly important in requirements engineering, yet they provide no evidence of maintaining traceability between the derived requirements and the source in the legal text, much less the traceability required for all the cross-references to other texts. Although their process appears to be a manual effort, Toval et al. fail to mention the length of the regulations they processed or how much time they spent extracting requirements from the law. Thus, it is difficult to properly evaluate the efficiency and efficacy of their approach. In addition, a legal requirements catalog requires updates each time the law changes. Finally, the PDP Catalog does not address the problem of overlapping or conflicting legal texts; the ability to manage and resolve these conflicts is an essential part of the requirements engineering process for systems governed by laws and regulations.

4. Supporting RE in Legal Contexts

Given our experiences to date [2, 13, 19] and our thorough survey of efforts to support the analysis of legal texts discussed herein, we identify several key elements for any system to support the analysis of regulatory texts for requirements specification, system design, and compliance monitoring.

Identification of Relevant Regulations

Our discussion in Section 1 focused on the need to identify relevant regulations, extract the requirements for a given system, and answer specific legal queries to test for compliance. Identifying relevant regulations may not appear to be a problem facing requirements engineers, but our experience to date shows that it is a key consideration during requirements elicitation. Oftentimes, analysts discover additional relevant laws or regulations only when they are midway through a careful analysis of a particular legal text. Much as the reader of this paper may see a citation and check the list of references to locate and read that paper, requirements engineers similarly identify external regulations or laws that constrain the very law they are examining at any given point in time. This is not a trivial activity. The referenced regulation may have a completely different set of definitions and terminology, requiring further interpretation and careful analysis. Making use of the supplemental documents to identify similar and related regulations will also aid in addressing the regulation identification problem.

Classification of Regulations with Metadata

Some classification of regulations is necessary for developers and auditors to sort through the large corpus of legal texts and identify those with relevance to the project or system at hand. To this end, the idea of tagging regulations with metadata, as proposed by [24] and others, can lead to a categorization of regulations over time. For example, a regulatory section such as HIPAA §164.310 can be annotated as generally describing security, or specifically detailing physical safeguards; in another categorization, it could be tagged as containing low-level system requirements. With each new regulatory text tagged, the corpus becomes more accessible and easily navigated.

Prioritization of Regulations and Exceptions

A system for handling regulations should address the nature of legal texts in its underlying approach. One key requirement is to handle the hierarchical nature of regulations. Oftentimes exceptions take precedence over the normative regulatory requirement. To properly assist requirements engineering efforts within this context, a support system should understand and manage the relationships between overlapping or contradictory regulations. This will enable analysts and auditors to make determinations about which regulations override others, depending on jurisdiction. This becomes particularly important when considering the effects of globalization. For example, various nations' regulations on personal data protection may differ or contradict one another; thus, users need mechanisms for resolving those situations. In addition, it is important to accommodate case law as well as other guiding documents. This information can again be captured as metadata; sections further explained or disambiguated by supplemental texts can be annotated with the more detailed information.

Management of Evolving Regulations and Law

It is critical for requirements engineers and compliance auditors to be able to manage the evolution of regulations over time. Given the frequent revisions to legal texts as previously discussed, requirements engineers need to be able to capture these changes and maintain an up-to-date view of the relevant regulations requiring analysis at any given time. It may be necessary to compare changes, and understand the impact of their scope, at distinct time periods to understand how requirements have evolved and how compliance efforts are impacted by modifications in the law. Thus the system must not only maintain traceability between regulations and requirements, but must also track the point in time at which that link was established. For legal analysis and the future development of case law, such metadata may be critical for verifying compliance. Analysts may be forced to update requirements or concepts as regulations change, and therefore they will require methods for tracking the status of development efforts vis-à-vis the changes in legal texts over time.

Traceability Between References and Requirements

As previously discussed, traceability support for both external and internal references is critical to ensure requirements engineers are able to accurately capture the full meaning of any given regulatory text. In Section 2 we discussed the prevalence of cross-references within regulations; external references also occur frequently in legal texts. Thus, it is imperative to maintain traceability between any section with a reference and the legal text being referenced. This is especially important given that external references often establish legally binding priorities among requirements and allowable information accesses, uses, disclosures, and removals. Navigating across these references, as well as from specific regulatory statements to the derived requirements, will improve analysts' understanding of the legal text and is essential for gathering all requirements and concepts expressed by a particular piece of regulation.

Data Dictionary and Glossary to Ensure Consistency

The use of consistent definitions and terminology is important in the design of any software system, and of paramount importance in the context of regulatory compliance. A data dictionary for all domain-specific definitions and acronyms is needed to support requirements engineers, policymakers, and auditors in establishing a unified glossary for the system specification, design documents, and compliance audit artifacts. In dealing with regulations, requirements engineers will frequently encounter unfamiliar and complex terms, making a thorough glossary even more important [16]. Given that multiple regulations may share similar words with different interpretations, users must be able to view any word's definition given the context of a specific regulation. These definitions should then be referenced in the creation of a system-wide glossary; once again traceability between the original legal terms and the system glossary must be maintained.

Semi-automated Navigation and Searching

Analysts need to be able to access regulations in a machine- and human-readable state. Previous requirements engineering research emphasized the relevance of such access in highly-regulated domains such as health care [16]. Some tasks, such as extracting concepts and adding metadata, need to be supported by semi-automated processes; use of semi-automated annotation tools is an active research topic (e.g. Semio Tagger [24] and CERNO [27]). In addition, users must be able to view the original regulatory text at any time, and traceability needs to be maintained between any machine-readable or logic-based format and the original natural language representation. Analysts must be able to easily search and navigate regulatory texts with varying levels of granularity. Given the complexity of regulations, users may need to search for specific terms, for more general concepts, or even scan entire sections of legal texts to clarify their understanding or support requirements engineering efforts.

Annotation of Regulatory Statements

As discussed in Section 2, legal and regulatory texts are laden with ambiguities. Some ambiguities in the law may be intentional, but analysts still need to establish an interpretation of the law in these cases, as well as maintain traceability with the section being interpreted. Analysts must be able to attach auxiliary annotations to ambiguous sections to flag them for further analysis in collaboration with the proper stakeholders, such as the organization's legal counsel. Ideally, analysts should be able to track interpretations across legal texts such that users will be able to view all assumptions upfront and differentiate the interpretations according to the context and conditions associated with any given situation. The ability to link legislation and software requirements with supplemental documentation will aid analysts by providing them with additional support for disambiguating texts for requirements extraction.

Queries Comparing Legal Concepts and Compliance

As supported by a wide range of approaches [1, 7, 10, 33, 34, 42, 43, 44], it should be possible to perform directed queries on the regulatory model. These queries enable analysts to support disambiguation and auditing efforts. Specific legal queries can allow analysts and auditors to identify all applicable regulations, discover all uses of a particular term or concept, and compare different regulations. Auditors may also wish to query the system to determine whether a particular regulation has been addressed in a system's design, or whether any requirements correspond to a given section.

5. Discussion and Future Work

We now outline some limitations in this survey's analysis and discuss future work toward a system for managing regulations.

This survey has largely focused on work within the computer science and artificial intelligence domains. It is possible that there has been work with regulations in other engineering domains that can be applied to the tasks facing requirements engineers and auditors in devising a system for using regulations. It would also be useful to examine how system developers are currently handling legal texts. Empirical studies of specific organizations would likely reveal additional requirements in dealing with regulations. One such study could focus on a particular domain and examine how requirements engineers and system developers identify and handle relevant regulations. Another study could focus on a particular regulation to pinpoint what elements of a legal text are used and how the regulation is managed in terms of the project.

Other concepts studied in requirements engineering are likely to be relevant for systems managing regulations. Future work should consider how requirements engineering research on viewpoints and frameworks can be applied to regulatory compliance systems. Research into natural language processing may also provide insight into parsing legal texts.

We are currently examining how to mine legal texts to create hierarchies of stakeholders, data objects, and events. We are also conducting an empirical study of a requirements specification to check for compliance. Our study begins with the previously-derived requirements and is working back to establish traceability with regulatory texts. We expect to uncover additional issues in monitoring compliance by working backwards from requirements specifications to the regulatory text and anticipate discovering additional requirements for our regulatory system.

6. Conclusion

This paper discusses the role of law in requirements engineering and attempts to bring attention to this important domain within the requirements engineering community. The characteristics of regulations make them both necessary and challenging to use during system development. Our survey examines the past 50 years of work in modeling regulations, extracting key concepts from regulatory texts, and monitoring compliance. In addition, we discuss what is required to effectively support analysts that must deal with regulatory texts in specifying system requirements as well as auditors in determining legal compliance.

7. Acknowledgements

This work was funded by NSF ITR-Cybertrust grant #0430166. We thank Travis Breaux, Laurie Jones, and Aaron Massey for their comments and feedback.

8. References

[1] L.E. Allen. "Symbolic Logic: A Razor-Edged Tool for Drafting and Interpreting Legal Documents," *Yale Law Journal* 66(6), pp. 833-879, May 1957.

[2] A.I. Antón et al. "The Role of Policy and Stakeholder Privacy Values in Requirements Engineering," *Proc. of the 5th IEEE Int'l Symp. on Req'ts Eng.*, pp. 138-145, August 2001.

[3] G. Antoniou et al. "On the Modelling and Analysis of Regulations," *Proc. of the 10th Australasian Conf. on Info. Sys.*, pp. 20-29, December 1999.

[4] G. Antoniou, D. Billington, M.J. Maher. "On the Analysis of Regulations using Defeasible Rules," *Proc. of the 32nd Hawaii Int'l Conf. on Sys. Sci.*, pp. 1-7, January 1999.

[5] A. Barth et al. "Privacy and Contextual Integrity: Framework and Applications," *Proc. of the 2006 IEEE Symp. on Security and Privacy*, May 2006.

[6] T.J.M. Bench-Capon. "Support for Policy Makers: Formulating Legislation with the Aid of Logical Models," *Proc. of the 1st Int'l Conf. on AI and Law*, pp. 181-189, May 1987.

[7] T.J.M. Bench-Capon et al. "Logic Programming for Large Scale Applications in Law: A Formalisation of Supplementary Benefit Legislation," *Proc. of the 1st Int'l Conf. on AI and Law,* pp. 190-198, May 1987.

[8] D.M. Berry, E. Kamsties, M.M. Krieger. "From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity," Univ. of Waterloo Technical Report, November 2003.

[9] C. Biagioli et al. "Automatic Semantics Extraction in Law Documents," *Proc. of the 10th Int'l Conf. on AI and Law*, pp. 133-140, June 2005.

[10] C. Biagioli, P. Mariani, D. Tiscornia. "ESPLEX: A Rule and Conceptual Based Model for Representing Statutes," *Proc. of the 1st Int'l Conf. on AI and Law*, pp. 240-251, May 1987.

[11] G. Boella, L. van der Torre. "Permissions and Obligations in Hierarchical Normative Systems," *Proc. of the 9th Int'l Conf. on AI and Law*, pp. 109-118, May 2003.

[12] T.D. Breaux, A.I. Antón. "An Algorithm to Generate Compliance Monitors from Regulations," Technical Report TR-2006-9, March 2006.

[13] T.D. Breaux, M.W. Vail, A.I. Antón. "Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations," *Proc. of the 13th IEEE Int'l Conf. on Req'ts Eng.,* September 2006.

[14] M. Casassa Mont, R. Thyne, P. Bramhall. "Privacy Enforcement with HP Select Access for Regulatory Compliance," Technical Report HPL-2005-10, January 2005.

[15] F. Chantree et al. "Identifying Nocuous Ambiguities in Natural Language Requirements," *Proc. of the*

IEEE Int'l Req'ts Eng. Conf., pp. 56-65, September 2006.

[16] L.M. Cysneiros. "Requirements Engineering in the Health Care Domain," *Proc. Of the IEEE Joint Int'l Conf. on Req'ts Eng.*, pp. 350-356, September 2002.

[17] Ernst & Young. "2006 Global Information Security Survey," November 2006.

[18] G. Governatori, A. Rotolo, G. Sartor. "Temporalised Normative Positions in Defeasible Logic," *Proc. of the 10th Int'l Conf. on AI and Law*, pp. 25-34, June 2005.

[19] Q. He et al. "Ensuring Compliance Between Policies, Requirements and Software Design," *Proc. of the 4th IEEE Int'l Workshop on Info. Assurance*, April 2006.

[20] W.N. Hohfeld. "Some Fundamental Legal Conceptions as Applied in Judicial Reasoning," *Yale Law Journal* 23(1), pp. 16-59, November 1913.

[21] B. Johnston, G. Governatori. "Induction of Defeasible Logic Theories in the Legal Domain," *Proc. of the 9th Int'l Conf. on AI and Law*, pp. 204-213, June 2003.

[22] E. Kamsties. "Surfacing Ambiguity in Natural Language Requirements." Ph.D. Dissertation, University of Kaiserslautern, Germany, 2001.

[23] E. Kamsties, D. M. Berry, B. Paech. "Detecting Ambiguities in Requirements Documents Using Inspections," *Proc. of the 1st Workshop on Inspection in Software Eng.*, pp. 68-80, June 2001.

[24] S. Kerrigan, K.H. Law. "Logic-Based Regulation Compliance-Assistance," *Proc. of the 9th Int'l Conf. on AI and Law,* pp. 126-135, June 2003.

[25] S. Kerrigan et al. "Information Infrastructure for Regulation Management and Compliance Checking," *Proc. of the Nat'l Conf. on Digital Gov't Research*, pp. 167-170, February 2001.

[26] B. Kitchenham. "Procedures for Performing Systematic Reviews," NICTA Technical Report 0400011T.1, July 2004.

[27] N. Kiyavitskaya et al. "Annotating Accommodation Advertisements using CERNO," *Proc. of the 14th ENTER Conf.*, January 2007.

[28] G.T. Lau, K.H. Law, G. Wiederhold. "Similarity Analysis on Government Regulations," *Proc. of the 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pp. 111-117, August 2003.

[29] G.T. Lau et al. "An E-Government Information Architecture for Regulation Analysis and Compliance Assistance," *Proc. of the 6th Int'l Conf. on Elec. Comm.*, pp. 461-470, October 2004.

[30] G.T. Lau, K.H. Law, G. Wiederhold. "Legal Information Retrieval and Application to E-Rulemaking,*" Proc. of the 10th Int'l Conf. on AI and Law,* pp. 146-154, June 2005.

[31] J. Martinek, J. Cybulka. "Dynamics of Legal Provisions and its Representation," *Proc. of the 10th Int'l Conf. on AI and Law*, pp. 20-24, June 2005.

[32] F. Massacci, M. Prest, N. Zannone. "Using a Security Requirements Engineering Methodology in Practice: The compliance with the Italian Data Protection Legislation," Technical Report DIT-04-103, 2004.

[33] M.J. May, C.A. Gunter, I. Lee. "Privacy APIs: Access Control Techniques to Analyze and Verify Legal Privacy Policies," *Proc. of the 19th Computer Security Foundations Workshop*, July 2006.

[34] L.T. McCarty. "The TAXMAN Project: Towards a Cognitive Theory of Legal Argument," *Computer Science and Law*, B. Niblett Ed., Cambridge Press: New York, pp. 23-43, June 1980.

[35] M.-F. Moens. "Combining Structured and Unstructured Information in a Retrieval Model for Accessing Legislation," *Proc. of the 10th Int'l Conf. on AI and Law*, pp. 141-145, June 2005.

[36] H. Nissenbaum. "Privacy as Contextual Integrity," *Washington Law Review* 79(1), pp. 119-157, February 2004.

[37] P.N. Otto, A.I. Antón, D.L. Baumer. "The ChoicePoint Dilemma: How Data Brokers Should Handle the Privacy of Personal Information," Technical Report TR-2006-18, July 2006.

[38] M. Palmirani, R. Brighi, M. Massini. "Automated Extraction of Normative References in Legal Texts," *Proc. of the 9th Int'l Conf. on AI and Law*, pp. 105-106, June 2003.

[39] N. Peek. "Representing Law in Partial Information Structures," *Artificial Intelligence and Law* 5(4), pp. 263-290, December 1997.

[40] D. Poulin, G. Huard, A. Lavoie. "The Other Formalization of Law: SGML Modelling and Tagging," *Proc. of the 6th Int'l Conf. on AI and Law*, pp. 82-88, June 1997.

[41] W.N. Robinson. "Implementing Rule-Based Monitors within a Framework for Continuous Requirements Monitoring," *Proc. of the 38th Hawaii Int'l Conf. on Sys Sci.,* January 2005.

[42] M.J. Sergot, A.S. Kamble, K.K. Bajaj. "Indian Central Civil Service Pension Rules: A Case Study in Logic Programming Applied to Regulations," *Proc. of the 3rd Int'l Conf. on AI and Law,* pp. 118-127, June 1991.

[43] M.J. Sergot et al. "The British Nationality Act as a Logic Program," *Comm. of the ACM* 29(5), pp. 370-386, February 1986.

[44] D.M. Sherman. "A Prolog Model of the Income Tax Act of Canada," *Proc. of the 1st Int'l Conf. on AI and Law,* pp. 127-136, May 1987.

[45] R. Stamper. "LEGOL: Modelling Legal Rules by Computer," *Computer Science and Law*, B. Niblett Ed., Cambridge Press: New York, pp. 45-71, June 1980.

[46] A. Toval, A. Olmos, M. Piattini. "Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection*," Proc. Of the IEEE Joint Int'l Conf. on Req'ts Eng.*, pp. 95-103, September 2002.

[47] A. Valente, J. Breuker. "ON-LINE: An Architecture for Modelling Legal Information," *Proc. of the 5th Int'l Conf. on AI and Law*, pp. 307-315, May 1995.