Legal Accountability as Software Quality: A U.S. Data Processing Perspective

Travis D. Breaux
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania, United States
breaux@cs.cmu.edu

Thomas Norton
School of Law
Fordham University
New York, New York, United States
tnorton1@fordham.edu

***Abstract*—Software and hardware innovation has led to new consumer products and services with significant benefits to consumers and society. These advances, however, can come with great cost to society when they fail to comply with government laws and regulations. While compliance failures do result from technical missteps in design, there is also a wide gap between the technical expertise and culture of legal analysts and software engineers, as well as competing priorities between legal requirements and business objectives. In this perspective paper, we propose changing legal compliance from a corporate oversight activity to a principal design activity, wherein lawyers and software engineers employ enhanced methods and tools tailored to bridge the cultural and knowledge gap and assess legal and business trade-offs. To that end, we describe a new software quality, called Legal Accountability, which can be evaluated alongside other qualities, such as usability, modifiability, performance and testing. Legal Accountability has five properties that lawyers and designers must attend to, including legal traceability, completeness, validity, auditability and continuity. We illustrate the quality with examples from the U.S. data processing perspective, and prior work in requirements engineering, before concluding with future and ongoing research challenges.**

***Index Terms*—law, regulations, software quality, compliance**

I. Introduction

Within the last two decades, software engineering innovation has propelled new consumer products and services into nearly every aspect of daily life. An innovation culture has emerged that celebrates the old Facebook motto "Move Fast and Break Things," while encouraging entrepreneurship and rapid software deployment to quickly assume a position of market leadership. This culture has in some ways diminished the role of requirements in managing stakeholder expectations, particularly in the push toward agile development. Due to software's ubiquity and pervasiveness, untethered innovation risks harm to the public, such as reduced safety, privacy and security. When professions fail to self-regulate, governments enact laws that guide businesses about how to innovate without compromising societal goals [73]. In addition, investors have pushed back on the old motto, asking companies to increase stakeholder accountability and to design "virtuous" software that better achieves societal goals [113].

Over the past two decades, researchers have proposed new methods and tools to improve accountability to law, including techniques to model legal requirements [14], [39], [54] and trace legal requirements to other software artifacts [21], [44]. Such techniques have been criticised for being developed without substantial input from practitioners of law [9]. Moreover, while these advances improve our understanding of challenges and solutions, compliance with law remains elusive for many companies. In 2021, the U.S. Federal Trade Commission (FTC), despite limited resources, made an example of five companies that violated the Children's Online Privacy Protection Act (COPPA) Rule, levying fines totalling over $2 million, imposing strict multi-year compliance and reporting requirements, and halting at least one company's ability to process children's information. In the EU, over 440 companies were fined over $1 billion combined for failing to comply with the General Data Protection Regulation (GDPR). And yet, there is an even larger cost to society, because regulators are overwhelmed and most violations go unresolved. The FTC reports needing "millions of dollars to hire more experts" across product development, data privacy and analytics, algorithms and software development [61]. Similarly, 98% of cases referred to the Irish Data Protection Authority, the GDPR's de facto enforcer for all of Europe, remain unresolved, and only 9.7% of Irish DPA staff are technology specialists [95].

In many cases, enforcement actions like those referenced above were avoidable at design-time by integrating a few additional steps into data processing (e.g., requesting consent from users), or through enhanced reasoning about the legal implications of one design decision versus another (e.g., how to delete data across multiple services, or whether a lossy hash algorithm is a reasonable de-identification method). Even large companies, who are best equipped to afford the legal and engineering expertise needed to develop their own methods, are failing to comply with the law. In this respect, we believe there is a gap between requirements and design that continues to be unaddressed, despite advances in requirements engineering methods to comply with laws. Based on our experience in studying this area, we identified three unaddressed issues that contribute to this gap: (1) the differences between the culture of law and software engineering; (2) the gap in differing technical expertise between legal analysts and software engineers; and (3) the competing priorities between legal requirements and business objectives.

In this paper, we propose a new perspective to address this shortfall, which is to raise the activity of compliance to become a first-class software quality, which we call Legal Accountability. Similar to other qualities, such as performance and testability, Legal Accountability requires additional requirements and design activities to reduce the cost of showing how software is accountable to the law. This includes activities across the software development life-cycle (SDLC). Herein, we focus on requirements and architecture-related activities and note that quality assurance among other activities are not covered, but are also critical to assuring Legal Accountability. Unlike other qualities, and in addition to the SDLC, Legal Accountability introduces the challenge of aligning expectations between legal analysts and software engineers with regard to their interpretation of the law. In addition, Legal Accountability is a national and international quality concern, that must be tailored differently depending on the operative legal context in the jurisdiction in which the software is operating. For example, the extent to which court records are digitized in a jurisdiction can affect ease of access to a software's legal context; or, how courts adjudicate legal cases (e.g., civil versus common law systems) affects how law is interpreted, as does the extent to which regulators in a jurisdiction have authority to enforce the law. Although Legal Accountability is a cross-national concern, this paper adopts a US-centric perspective to demonstrate Legal Accountability's application in context.

The remainder of this paper is organized as follows: in Section II, we discuss the cultural and expertise differences in law and engineering; in Section III, we introduce background on software quality and show how it can be used to frame Legal Accountability and other quality trade-offs; in Section IV, we introduce Legal Accountability as a composite of five properties, which we contextualize using prior work in requirements engineering to identify advances and shortfalls in the state-of-the-art. This review of prior work is extensive, but not systematic, and thus important and relevant work is potentially missing. In Section V, we present new, and revive old, research challenges that arise from this perspective, with our conclusion presented in Section VII

II. Language, Cultural and Expertise Challenges

Legal Accountability faces cultural and expertise challenges from integrating law and technology. First, legal requirements are written in the language of law to govern entire industries [88] and not only one company or product, whereas software requirements are written in the language of engineers and users with a specific product or software feature in mind. This broad and encompassing audience for legal requirements leads to purposeful ambiguity which is viewed by lawyers as necessary for law to function properly. Because laws are written about the technology of the day, legal requirements can refer to outdated concepts, such as "workstations," while ignoring modern innovations, such as "tablets" and "mobile devices." Ambiguity and dated language require interpretation to understand the intent of the law. In addition, legal requirements can describe any type of requirement, from high-level, vague and aspirational goals to low-level, design constraints. Designers must assess how to respond to legal requirements by identifying overlaps with software requirements [15], by negotiating requirements conflicts [77], and by refining vague or ambiguous legal requirements into actionable software requirements [79], while aligning knowledge from the legal context with decisions in the design context. In this respect, legal requirements can disrupt business objectives by shifting priorities among software qualities. Finally, the culture of law, which is generally sensitive to legal risk and is often motivated to minimize it, is distinct from and sometimes in conflict with the culture of software development, which emphasizes fast releases and untested innovation, while allowing "policy" to emerge from what users and society accept as the new design status quo. Further, in this setting, a lawyer has a duty to their organization to minimize risk to it, which may conflict with the societal and ethical goals embodied in law and regulation.. Lawyers are trained to follow precedent, and the nature of law practice largely rests on applying guidance from previous case law or interpretation of legislators' intent. And the business of law is largely built around firms' and the ability to identify and mitigate risk efficiently. From the very beginnings of their legal education, those who enter the legal profession are trained to conduct nuanced analysis of situations that may be more ambiguous than they initially seem, and for which more than one approach may be correct. In contrast, laypersons may approach the same problems more simplistically [43]. In nearly all of law school's doctrinal courses such as torts, contracts, property, criminal law, and civil procedure (among others), law students are taught using a method that relies on thorough analysis of court decisions to derive legal principals and examine their functioning [4], [31]. But each decision, by its very nature, represents a situation in which a business, personal, or social relationship malfunctioned – so much so that court intervention was necessary to achieve a resolution. Many years focusing on such stories conditions one to a legal culture in which risk aversion is dominant. Software engineers, by comparison, use plan-driven software processes, such as variations on the V-model [32] or Spiral [10], or agile methods, such as Kanban [2] or Scrum [110]. Herein, we assume software developed using agile methods, motivated by a recent survey of software developers, where agile methods were used by 94% of respondents, whereas as few as 7% report using plan-driven methods [71]. Finally, plan-driven methods frequently integrate risk analysis activities, whereas agile methods emphasize fast delivery at the cost of limited documentation [11], which sits in contrast to the legal culture of risk aversion.

III. Software Quality, and Competing Priorities

Software quality attributes are cross-cutting design and validation concerns that include qualities that are critical to satisfying requirements, such as modifiability, performance, safety and security. Software quality can arise as non-functional or quality requirements [40], and softgoals [84]. These requirements necessarily affect how software designers choose among various technical means, sometimes called architectural patterns or tactics, to rationalize a quality increase [5]. Effective team coordination and communication, requirement specification quality, and responsiveness to changing requirements can affect how well verification and testing activities guarantee software quality [8]. Several qualities are guided by industrial standards, such ISO 9241-11:2018, which describes usability or "the extent to which a system can be used to achieve specific goals." Under ISO 9241-11:2018, software is thus designed to be more usable. To date, there are no standards on how to design software to be more legally accountable.

Software quality introduces quality trade-offs, where designers can be forced to decrease one quality in order to increase another. Trade-offs can be examined using goal refinement alternatives [85]. Software architects play a key role in engineering quality requirements [24]. Architects can use quality attribute scenarios to align requirements with architectural design decisions, as well as *architectural tactics*, which are reusable, techniques to achieve a specific quality attribute [5]. For example, deferred binding, which is an architectural tactic to increase modifiability, can be implemented using plugins. Deferred binding can also allow attackers to introduce malicious code at runtime, thus reducing security. Lost quality can sometimes be remediated, e.g., plugins can be checked for a trusted digital signature prior to binding to thwart malicious code injection. In other cases, decreases in one quality must be accepted to increase a higher priority quality, e.g., encrypting communications at the cost of performance needed to encrypt data in transit. Softgoal satisficing can be used "to determine the degree to which a set of nonfunctional requirements is supported by a particular design" [83].

Laws can be motivated by a single quality attribute, such as safety, and further defined by specific domains. In the U.S., the safety of software used in medical devices, automobiles and aircraft are governed by separate regulatory frameworks. Software used in medical devices is regulated in the U.S. by the level of risk to patient life, wherein the highest level of approval requires traceability among design, testing and risk management plans [51]. Similarly, Article 25 of the EU General Data Protection Regulation (GDPR) requires "data protection by design and default," in which developers are required to design for a quality without specific guidance on how to conduct design activities. Designing for a legally required quality may satisfy one but not all legal rules.

Software process focused, such as those described above, enumerate requirements and design processes that must be followed, whereas other regulations are focused on specifying design constraints on technology. The U.S. Information and Communication Technology Accessibility 508 Standards and 255 Guidelines, which governs information accessibility by individuals with disabilities, defines permissible display color and contrast levels, font sizes, and file formats, including ANSI/AIIM/ISO 14289-1:2016 (PDF/UA-1).

Finally, other laws increase quality by regulating business practices, such as the U.S. Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule, which regulates which healthcare practices must adopt specific privacy requirements. These laws often include legal ambiguity, which must be interpreted by designers [79], [94].

Unlike qualities that originate in design or development, Legal Accountability is a quality that arises from compliance activities, beginning in a review of laws, legal precedents and enforcement actions and leading into requirements and design. Legal Accountability is more than compliance, which has largely become an oversight activity where engineers apply canonical solutions to legal problems at the direction of legal analysts. In contrast, Legal Accountability benefits from co-design activities involving legal analysts and software engineers. In addition, where laws aim for organizations to achieve societal goals, and because technology frequently outpaces law, Legal Accountability can be achieved by engineers who move beyond any specific regulatory rule to achieve the legal goal, and by identify potential pitfalls and software failures not explicitly described by the law [12]. We envision Legal Accountability being applicable to any regulatory framework, and that its utility will be greater when regulation lacks specific technology-focused requirements, or when the regulated qualities are in competition with other business-driven qualities. Legal Accountability can be used to rationalize design decisions in ambiguous regulatory contexts, such as those that govern business practices, versus technologies and their engineering process (e.g., building codes).

IV. Legal Accountability

The manner in which requirements and software engineers approach software quality is ideal for addressing the current gap in software compliance. Quality is realized through a composition of supporting requirements that introduce alternatives and require trade-offs [85]. Quality fundamentally changes how software is designed, because it must support activities to demonstrate the quality (see ISO 9000:2015).

We define *Legal Accountability* as the extent to which software is accountable to law and regulation. Similar to other software qualities in design [5], Legal Accountability is increased by design decisions of software engineers as observed through measurable responses by the system. Unlike other qualities, Legal Accountability is predicated on bi-directional communication between software engineers, who must communicate to legal analysts how the software complies with law, and legal analysts, who must clarify the intent and context of the law. Therefore, we identified five minimum properties of Legal Accountability that are motivated by a software's need to demonstrate accountability to law and regulation, which embody societal and ethical goals recognized by lawmakers. Demonstrating accountability to law requires answering the following questions: in the event of a claim of legal non-compliance, one must demonstrate what was done to comply with a legal rule (Traceability), how does one know they have covered all legal concern (Completeness), and what design-time and runtime evidence exists to defend against a legal claim (Auditability)? The strength of the position as a defense depends on knowing that a decision is legally correct (Validity), and on being responsive to change: as societal or ethical goals change, the law changes, or the software changes, ensuring that Legal Accountability is preserved by revisiting such changes in the requirements and design (Continuity). A few of these properties have received significant attention in the requirements community, e.g., traceability, whereas others have received far less, e.g., validity.

In the remainder of this section, we refer to legal rules as *rights*, which describe what actors are permitted to do, *obligations*, which describe what actors are required to do, and *prohibitions*, which describe what actors are prohibited from doing. *Duties* are obligations imposed on one actor by another actor who exercises their rights. Formal taxonomies for legal rules have been based on Deontic Logic [56] and Hohfeld's legal concepts [49].

A. Traceability

Traceability holds, if every obligation, duty and prohibition that covers the software is traceable to a design decision. Companies may further wish to trace rights, which yield discretionary requirements, should the company choose to exercise those rights. Software requirements can be extracted from laws and regulations [16], and such artifacts need to be traceable back to legal citations to measure legal requirements coverage [42]. In addition to handling legal citations, requirements engineers must balance rights and obligations, reconcile exceptions to legal rules, and resolve ambiguity when extracting legal requirements [16].

Traceability depends on two critical techniques: identifying legal primitives, and resolving cross-references. Methods have been introduced to automatically annotate legal texts to identify rights, obligations and permissions, including techniques based on context-free grammars [86] and regular expressions [97] and constituency and dependency parsing [105]. Cross-references must be resolved to identify dependencies among legal requirement [14] to answer questions, such as where are terms-of-art defined, or does this requirement have any refinements, follow-on obligations, or exceptions? Techniques often employ schema to formalize the text's paragraph structure [14], [98], followed by regular expressions to detect cross-references in texts [98]. Annotations can be used to query legal texts for knowledge discovery [104].

Traceability supports the mapping or linking of legal requirements into software requirements artifacts that are more familiar to designers or more easily integrated into existing design processes. Target artifacts to which legal requirements can be traced include enhanced requirements templates [103], business models [99], and regulation scenario models with compliance links [37]. When deriving requirements or other software artifacts from law, tool-supported, document-based methods are needed to encode trace links from rules in legal texts to these artifacts [36]. Traceability information models can be configured to select the most relevant trace links between critical requirements and design decisions [108].

Techniques to automate traceability in compliance take advantage of machine learning [21], [44]. This includes applications of probablistic network models that were applied to trace among HIPAA security regulations and healthcare product requirements [21], and deep learning with word embeddings applied to positive train control requirements that combined regulatory rules with system requirements into a single dataset [44].

The above advances in traceability are necessary to address compliance in software design. Only tracing from legal text to software artifacts excludes the broader legal context, which we now discuss.

B. Completeness

Completeness holds, when for a law that covers a business practice supported by software, every obligation, duty and prohibition imposed by that law is accounted for; this includes obligations, duties and prohibitions that follow from the business or other parties exercising any rights granted to them by the law [16]. Completeness encompasses multiple considerations, including: (1) whether a practice is governed by a particular law; (2) if governed, what must be done to comply with the law; and (3) if governed, what are the risks of noncompliance (i.e., should the practice be designed to comply). To achieve completeness, it is necessary to comprehend to the entire the legal context, which includes laws and regulations, legal precedents, and enforcement actions. While it is impossible to guarantee completeness[1] and assert that no element of this context has been missed, designers should plan to periodically extend and update their comprehension of the legal context.

1) Laws and Regulations: In the U.S., laws consist of statutes, enacted by a legislative body, and regulations interpret statutes as directed by the legislature to yield rules that organizations must follow to comply with the law. Laws and regulations impose direct requirements in the form of duties and obligations, and can establish relevant rights. Guidance documents and other instruments, which are often non-binding, help to clarify existing obligations. These requirements are written to describe systems in one or more industries, thus they can closely align with system requirements in software engineering [88]. The subject of legal requirements can be organizations, a stakeholder or a technology. Legal requirements may be goal-oriented or they may describe low-level design constraints. Companies may need to comply with laws from multiple jurisdictions [38], [41]. Several distinctions are important for engineers to understand, including legal rights, obligations, and permissions, and relationships among requirements, such as exemptions and refinements [14], [16]. Legal requirements can be framed as rules versus standards: rules provide "clarity and forewarning," whereas standards provide "greater flexibility for interpretation" by engineers [68].

Yet obligations, duties, and exemptions prescribed in statutory law or regulation are not always clear. Indeed, law and policy often avoid clear specification in favor of purposefully ambiguous rules that require interpretation. Oftentimes, law

[1]Requirements engineers refer to "coverage" which accommodates incompleteness. Because ignorance of the law is not a legal defense, we chose completeness to describe this property.

and policy makers harbor uncertainty about when a particular law or policy will apply, and therefore build in flexibility to account for unknown or unpredictable circumstances. Or, drafters of law and policy may be unsure about how to solve a particular identified problem, and turn to ambiguity to support experimentation toward determining the most effective problem-solving methods.

2) Legal Precedents: Ambiguities in legal rules are often resolved through after-the-fact and retroactive oversight by courts and similar adjudicative bodies. Through cases, courts resolve disputes that arise between parties concerning the application of a law or regulation to a specific set of facts. Through the adjudicative process, ambiguities in law and regulation are interpreted and clarified over time by judicial rulings on specific situations. In a recent case involving the California Consumer Privacy Act (CCPA), a U.S. federal court was called to interpret the applicability of statutory provisions defining "businesses" and "service providers" to a cloud software company that suffered a cyberattack as the result of having an allegedly "deficient security program." The company argued that the CCPA provisions it was alleged to have violated only applied to "businesses," and the company was not a "business" under the law, but rather a "service provider." After reviewing the company's practices, the court disagreed, and determined that the company did qualify as a "business." Further, the court concluded that the statutory definition of "business" is a broader term that encompasses "service provider," and therefore, the company could not evade CCPA liability even if it were a "service provider."

Such cases form binding legal precedents that become an additional source of legal authority alongside statutes and regulations. The gradual development and extension of law and regulation through cases with specific fact patterns allows for careful consideration of meaning and effects at a level of granularity that is usually impossible to reach during the drafting process [30]. As a result, the decisions of adjudicative bodies become essential to comprehending and implementing law and regulation, and thus crucial to establishing the legal context and completeness. In the example above, the clarification on the interpretation of business affects how every company covered by the CCPA should interpret this definition. Thus, binding legal precedents represent reusable legal knowledge that designers can integrate into their compliance practices.

3) Enforcement Actions: Enforcement actions include legal actions arising in response to commercial failures. Enforcement actions by regulatory or oversight bodies constitute an additional source of legal requirements that must be accounted for in accountable systems. This is especially true in the U.S., where government agencies at the federal, state, and local levels are responsible for enforcing laws passed by legislative bodies. [See, e.g., Chevron U.S.A. Inc. v. Nat. Res. Def. Council, Inc., 476 U.S. 837 (1984)] For example, the Federal Trade Commission (FTC) is the de facto privacy and data protection authority in the U.S. The FTC polices unfair or deceptive acts in commerce to address privacy and data protection wrongs. Settlement agreements between the FTC and alleged violators make up a robust body of jurisprudence that organizations must examine to ascertain the current boundaries of prohibited and permissible conduct related to privacy and data protection [106]. Thus, enforcement actions are a valuable tool to gauge the regulatory risk associated with particular courses of conduct.

Legal analysts play an essential role in achieving completeness. A legal analyst specializing in a particular subject area, such as data protection law, will be fluent in not only existing statutory or regulatory law in that area, but in proposed law as well. In addition, the legal analyst will track the development legal interpretations of courts and adjudicative bodies by reviewing judicial opinions, the outcomes of enforcement actions, and documents supplying guidance on best practices for complying with relevant laws. Through the practice of law, as well as teaching, publishing, and presenting, legal professionals continuously hone their skills and expertise to be current and competent. In many jurisdictions across the globe, legal professionals are subject to continuing legal education and training requirements even after entering practice. Armed with legal fluency, legal analysts can help define, interpret, and convey the legal context to software engineers to better account for relevant obligations, duties and rights.

C. Validity

Validity holds, if every traceable software requirement and design decision is a valid interpretation of a legal requirement and its legal context. Because validity is a joint decision by the legal analyst and software engineer, validity benefits from designs that are simple and explainable to non-engineers with legal training, and that have guarantees in spite of dynamic, distributed and parallel processes that increase complexity. Interpretations of law and policy can be open and varying due to textual vagueness, incompleteness and syntactic, semantic, and referential ambiguity [79]. Such ambiguity and vagueness can yield *increasing flexibility* in how designers determine what is legally valid [101], while also increasing perceived risk by users [7]. What is legally valid is also subject to *decreasing flexibility* in light of increased enforcement risk [13].

In requirements engineering, important but little work has been done to understand the interpretability of law. Legal experts were shown to demonstrate higher rates of statistical agreement than technical professionals or lay people, when deciding if an organization is covered by a law and whether two legal requirements are similar or different [43]. Software engineering graduate students exhibit low agreement when deciding if a requirement satisfies an organization's legal obligations, including when using the Wideband Delphi method to organize engineers into a ritualized agreement exercise [78]. The above research shows that deciding coverage is difficult for engineers alone; whereas, crowdsourcing was shown to resolve lexical, syntactic and semantic ambiguity in law when choosing the correct definition of a term with 87% accuracy, as compared to expert decisions [29]. Techniques in requirements engineering have been criticized for having a "textualist view" of the law, often examining only a single requirement or single law, while ignoring the larger legal context, such as case law, doctrinal work, and unofficial judicial practice, and notably criticized for excluding the role and input of legal experts [9].

To address this limitation and properly advance the state-of-the-art, we envision one way to establish legal validity using a bidirectional design review. In Figure 1, we envision three distinct phases in the review: (1) *legal comprehension*, wherein the legal expert and software engineer work together to comprehend the legal context as it applies to the existing or planned software; phase (2) *design expression*, wherein the engineers propose a new or modified design to satisfy legal requirements; and phrase (3) *design recording*, wherein the justified design proposal is summarized as issues recorded in a modern issue tracker, including features for any new design proposals, and bugs when elements in an existing design do not satisfy new requirements.

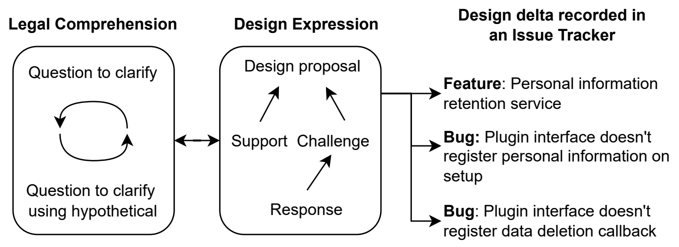
Legal Comprehension
Question to clarify
Question to clarify using hypothetical
Design Expression
Design proposal
Support
Challenge
Response
Design delta recorded in an Issue Tracker
Feature: Personal information retention service
Bug: Plugin interface doesn't register personal information on setup
Bug: Plugin interface doesn't register data deletion callback

Fig. 1. Establishing Legal Validity through Design Reviews

During comprehension and expression, the parties engage in an inquiry cycle-like discussion [90] that is supported by argumentation structures to clarify legal boundaries, and justify and challenge design proposals and counter proposals. Such structures have been proposed in RE in prior work: including design arguments [55], [59], [92] based on the Issue-Based Information System (IBIS) [69], security arguments [46] using the Toulmin model [65], [115], and legal compliance arguments integrated into goal models [53] based on the acceptability evaluation framework [57]. Tools to identify legal rules, cross-references and ambiguities from legal texts [29], [97], [98] can provide inputs to these discussions, however, establishing legal validity must integrate the broader legal context to include non-binding guidance documents, legal precedents from court cases that add judicial clarity, and histories of regulator enforcement actions. Moreover, these discussions can result in design alternatives that can be documented using goal-modelling tools that support argumentation [53].

The three phases shown in Figure 1 build on theory in cross-functional teaming to promote effective knowledge transfer across legal analysts and engineers using knowledge transcendence [74], and front- and back-loading knowledge transfers SRJ16, while recognizing the transfer impact of language differences, subject-matter novelty and value systems [28].

Legal validity is established by tracking discussions during comprehension and expression, and auditability (see Section IV-D) is supported in the expression phase as static evidence (e.g., library and service specifications, and their pre- and post-conditions) and dynamic evidence (e.g., log data) described at design-time and collected at runtime. While we present comprehension and expression as distinct phases, they are more likely intertwined.

Unlike traditional design reviews, the envisioned approach must integrate with agile methods by reusing existing infrastructure, such as user stories and issue tracking systems. This requirement is critical as some developers prefer defined compliance processes that prioritize the software release process [60], and agile methods have been widely adopted to support fast and responsive release cycles [71].

To further illustrate Figure 1, we present Table I that shows a hypothetical design discussion modeled on prior research [58] to comply with §1798.105(a) of the California Consumer Privacy Protection Act (CCPA), which reads: "A consumer shall have the right to request that a business delete any personal information about the consumer which the business has collected from the consumer." Indentation indicates a threaded response to an earlier comment. The discussion illustrates exchanges where legal comprehension and design expression lead to design decisions. Each statement is labeled by the participant role (engineer, legal analyst) and a unique statement number (/S1, /S2, etc.) In the left column, engineer #1 aims to interpret a legal definition by positioning a technical example (S1), which is justified by the legal analyst (S2) and challenged by engineer #2 (S3). To support auditability, the structure to comprehend the definition of "personal information" on the left illustrates the legal and design rationale for excluding web logs from functions created to satisfy legal rules, such as the purge function to support 105(a) described on the right. This is an example of a static design element that originates in the comprehension phase. In the right column, engineer #1 proposes a design concept for having developers register personal information types by category at runtime (S6), which is a dynamic design element that can be audited using logs of data type registrations by components. By registering data types, engineer #1 aims to reduce compliance costs if the law changes by introducing a general design solution (S8), which is a principal goal of Legal Accountability.

In the right column, engineer #1 also proposes a design concept for purge (S6), which undergoes a series of reciprocal challenges by engineers #2 and #1 (S7-S9), and a justification by the legal analyst (S10). To support legal validity, the decision to register data by category, and decentralize purge, is motivated by considering design alternatives (S7 versus S8) and design constraints (S9 is a constraint that justifies S7), as well as additional legal rules (S10). The outcome of the hypothetical discussion would be a new feature (the retention service) plus changes to existing components, as noted on the right side of Figure 1. Legal validity rests on the totality of these questions and responses in support of the law. This could be realized as Kaur suggests by identifying the kinds of statements, their relations to other statement types, and so on, as an evolving, coordinated, collaboratively constructed decision by the legal analyst and software engineer.

Legal Comprehension	**Design Expression**
Engineer 1 /S1: How do we define "personal information?" If we collect a user's IP address in a web server log file, is that personal information, and do we need to purge data from the log? I doubt the information security team would approve of that. #security **Legal Analyst /S2**: Personal information is defined in 140(o)(1) and 140(o)(1)(A) specifically lists IP addresses. **Engineer 2 /S3**: True, but 105(d)(2) makes an exception for when information is used to "detect security incidents, protect against malicious, deceptive, fraudulent, or illegal activity" which should cover log files. **Engineer 1 /S4:** Okay, but 105(d)(7) makes an exception for "internal uses that are reasonably aligned with the expectations of the consumer." If that's the original purpose for the collection, then why do we need to delete anything? **Legal Analyst /S5**: That's true, only if the uses are "internal" and "expected," noting that 105(d)(7) doesn't apply to third parties, and 105(c) requires us to "direct any service providers to delete the consumer's personal information from their records." #thirdparty	**Engineer 1 /S6**: I understand, so to support a 105(a)-compliant purge function, we need a way to categorize and track data as security-related and internal use-only. We may want a service where teams register personal information under these categories, and receive a future callback when a customer requests deletion. **Engineer 2 /S7**: Why do we need to track the categories? Can't teams receive the request and delete personal information on their own? **Engineer 1 /S8**: How would we verify that teams covered all the information types listed in 140(o)(1), and what if interpretations of the types change? I feel we should centrally manage retention. **Engineer 2 /S9**: I know of one situation where the team stores data locally on a mobile device to optimize for performance during poor network connectivity. In that case, they would need to receive notification to perform the purge locally. **Legal Analyst /S10**: A reason to track the categories is to ensure consumers are properly informed about the categories pursuant to 110(a)(1) and (c)(1), and 115(a)(1)-(3), and other similar sections. #categories

TABLE I
HYPOTHETICAL, THREADED DESIGN DISCUSSION BETWEEN LEGAL ANALYSTS AND SOFTWARE ENGINEERS

D. Auditability

Auditability holds, if the design and runtime artifacts provide evidence of when every obligation is discharged and, of when duties are discharged in response to stakeholders who exercise their rights and the business is a counterparty to those rights. Auditability serves internal business auditors, contracted third-party auditors, or those who have the legal right to challenge or request evidence of an accounting, including subpoena powers. Traceability can be used to aid auditability by linking legal rules to target software artifacts that implement those requirements. However, auditability concerns the type and quality of the target artifacts as they serve to demonstrate that obligations and duties have been properly discharged. We distinguish between *direct evidence* of legal requirements to document design decisions, and *indirect evidence* that covers the detailed implementation of the decision, and evidence of *static* representations of software at design-time, such as documented requirements, architecture, test cases and formal and informal models, versus *dynamic* representations of software at runtime, such as log data, to show the context of the design choice.

Legal texts and guidelines can be explicit about what is auditable, such as the HIPAA Privacy Rule, which includes an obligation in §164.514(b)(1)(ii) to document the decision of an expert when applying statistical methods to de-identify protected health information. If a statistics-trained software engineer were to propose using differential privacy [27] to de-identify health data, then this decision must be documented under the Rule. What is less clear from the Rule, however, is the extent to which the legal analyst and software engineer need to document the *implementation* of the decision. The guidance raises questions that companies should answer, such as [52]: who qualifies as an expert, how long is an expert decision valid for, and how does an expert assess the risk of a decision? The guidance does not cover how to document the software-based evidence created by companies who discharge the obligation. Does the software need to record the code-level implementation, or each event when data is de-identified, or which version of a method was applied, should the expert propose multiple methods, or revise a method later?

In addition to explicit documentation requirements found in legal texts, companies may choose to document decisions because of exigencies not covered by the text. For example, the California Consumer Privacy Act (CCPA) does not expressly require that a company obtain consent from a website user before placing cookies on their browser. Under the CCPA, certain obligations, duties, and rights are triggered when a business engages in a "sale" of personal information, which the CCPA defines as the disclosure of or otherwise making available personal information "for monetary or other valuable consideration." This definition applies to the exchange for value of all consumer information, including sharing personal data captured by cookies and other tracking technologies with third parties. However, a business does not "sell" personal information when the consumer directs the business to "(i) intentionally disclose personal information or (ii) intentionally interact with one or more third parties." Thus in order to minimize risk when permitting other parties to deploy cookies, a website may avoid the triggering of the attendant obligations, duties, and rights by obtaining and documenting a user's consent as evidence that the consumer directed the business to "intentionally disclose personal information."

In addition, other parties may be granted the right to request an audit. Organizations may implement their own policies for auditing, or contract with third parties, and laws may grant outside parties such rights. The GDPR, Article 15, for example, grants people described by data, called data subjects, the right to obtain a copy of their personal data that is being processed by software. While a data subject request could trigger a flurry of activity by system administrators to search databases for relevant data, a system designed to be Legally Accountable would reduce the effort and time to generate this copy by design. In all circumstances, auditability should be driven by *who has the right* to request an audit, and *what should narrowly be reported* within the scope of a legally required audit.

Design techniques for "forensic-ready" software, which explicitly preserves evidence to aid in digital forensic investigations, could support auditability [1], [89]. This includes methods that use harm-based, risk metrics to prioritize which requirements to audit as potential sources of evidence [26], and techniques to guide developers on deciding where to log and what to log [96]. While these approaches were initially developed in the context of security incidents, they may generalize to violations of law.

E. Continuity

Continuity holds, if the design ensures to the greatest extent possible that no gaps arise in compliance, and when gaps do arise due to environmental or technical issues, then designers can show that any issues are commercially reasonable. Continuity is challenged when technology innovation outpaces the law, introducing harms unforeseen by regulators that would be cause for updating the law. In this respect, designers who anticipate and achieve the legal goal, despite the absence of a specific legal rule, can increase the robustness of continuity against future changes in law. Continuity is further challenged when the legal context changes, such as the introduction of new laws and amendments to old laws, new judicial opinions that change how existing law should be interpreted, or when regulators update their guidelines. Under these circumstances, the legal analyst and software engineer must review an existing design and, using traceability, identify the requirements, design and implementation that must be re-assessed.

While laws and guidelines do change, many prominent laws do so infrequently. For example, the HIPAA was enacted in 1996, and one regulatory interpretation of the HIPAA, called the Privacy Rule, was finalized in 1999. The Privacy Rule was then modified in 2002, 2013, 2014 and 2016. In the U.S., the public is invited to comment on proposed changes, which occurred prior to each of these modifications. While the HIPAA legislative act was not amended between 1996-2021, the supporting regulation was amended. Similar examples can be found of changing laws and guidelines, such as DO-178B, which is the de facto industry standard covering flight control software to comply with 14 CFR 21.601-621 and which was introduced in 1992 and updated in 2011.

Continuity is also challenged when software is modified statically or dynamically, and when software is distributed across development teams. Designs that are resilient to unspecified changes reduce the risk of breaking continuity and of maintaining Legal Accountability. Technology innovation can outpace law, in which case revealing and achieving the regulatory goal can increase robustness against non-compliance when software changes [12]. To illustrate, consider a business objective to deliver flexible online learning environments to children in K-8 schools. The U.S. Children's Online Privacy Protection Act (COPPA) Rule requires this business to obtain verifiable parental consent before collecting information from children under 13. Figure 2 presents a hypothetical software architecture that consists of a consent module for standardizing consent functions across services, an educational platform that coordinates services, and one or more learning module plugins that allow the platform to be modified post-deployment with minimal cost. The platform was designed using the Broker architectural design pattern, which supports modifiability by dispatching service requests to responsible modules that can be changed at runtime. Notably, learning module #2 is a third-party plugin, thus platform designers must account for how third-parties will comply with COPPA.

Figure 2 presents four legal obligations about obtaining consent under COPPA, each numbered by the legal paragraph containing those obligations. To integrate compliance into an agile process, software engineers can write a user story, "As a parent, I will first consent to the collection of my child's personal information," and attach acceptance tests to the story, such as ensuring that the consent page includes an unchecked option to consent to third-party disclosures. The COPPA Rule describes requirements in §312.5(b) concerning permitted methods to obtain consent that may be included with this story as legally required story sub-tasks or acceptance tests. When consent is revoked, children's data must be deleted from any devices where that data is stored.

The "flexible learning environment" business objective is interpreted by engineers as a plugin-based architecture that supports dynamically loaded learning modules. This objective drives modifiability as a high-priority software quality attribute. Legal Accountability under COPPA, however, imposes two specific constraints on the architecture: (i) prior to offering parents access to third-party plugins, parents must first specifically consent to these plugins under §312.5(a)(2); and (ii) whenever a plugin is re-loaded due to a material change in plugin design, parental consent must be re-obtained for this change under §312.5(a)(1). In addition to these constraints, Legal Accountability requires legal analysts and software engineers to be able to communicate how this architecture implements these constraints, and if challenged by a regulator, they must show evidence that non-compliance events have not occurred at runtime.

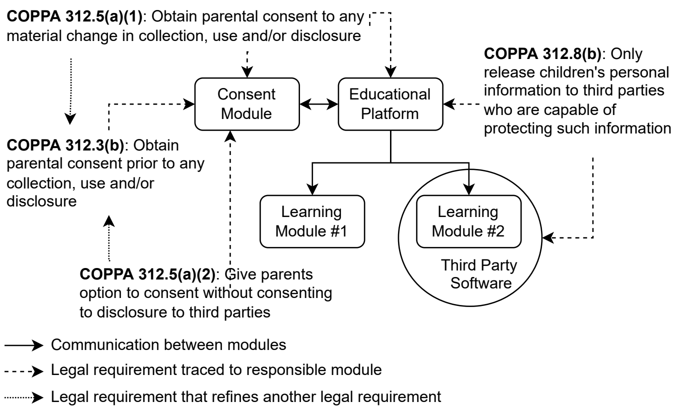
COPPA 312.5(a)(1): Obtain parental consent to any material change in collection, use and/or disclosure
COPPA 312.8(b): Only release children's personal information to third parties who are capable of protecting such information
Consent Module
Educational Platform
COPPA 312.3(b): Obtain parental consent prior to any collection, use and/or disclosure
Learning Module #1
Learning Module #2
Third Party Software
COPPA 312.5(a)(2): Give parents option to consent without consenting to disclosure to third parties
Communication between modules
Legal requirement traced to responsible module
Legal requirement that refines another legal requirement

Fig. 2. Legal Continuity and Platform Modifiability

In addition to requiring software to be easily auditable, Legal Accountability requires software to remain accountable, even when other design pressures, such as ease of modification, are prioritized. By introducing Legal Accountability as a software quality, these pressures can be assessed in the context of their impact on compliance at design-time.

V. Research Challenges

In this section, we identify a cross-section of research areas that require investigation to realize Legal Accountability.

A. Cross-functional Teaming

- **Challenge #1: How to integrate legal and engineering experts in agile, cross-functional teams?**

Design reviews can be used to establish legal validity by drawing upon legal and software engineering expertise in a combined, cross-functional team. Cross-functional teams are known to face considerable knowledge transfer challenges [74], which can be addressed by improving the codification of knowledge to support easy transfer [63], and by recognizing the boundaries of transfer due to expertise dissimilarity [64] and transactive memory [63]. Creativity in cross-functional teams increases when teams are encouraged to take risks [100], which can create tension with the risk aversive culture of law. A leading cause of failure in cross-functional teams is siloed team members and lack of adherence to shared artifacts, such as design specifications [112]. To support legal accountability, new methods and tools must improve knowledge transfer and enable bi-directional co-design of specifications early in the development process.

Agile methods are among the most commonly used software processes [71], and developers have reported a preference for defined compliance processes that prioritize software release [60]. While cross-functional teams exist in software development, they may be uncommon, even in safety-critical systems [76]. The emphasis on communication over documentation in Agile does not always lead to better functioning, interdisciplinary teams, especially when design rationale is undocumented or changing [48]. However, lightweight methods that support knowledge transfer between legal- and engineering-trained team members could be a promising area of requirements research. Moreover, integrating these methods into existing tools for continuous integration, continuous delivery (CI/CD) would further meet the challenge of aligning Legal Accountability with modern software release and maintenance activities.

B. Dialogical and Dialectical Reasoning

- **Challenge #2: How to achieve design acceptance with cross-disciplinary reasoning?**

Aligning the legal and design context to support Legal Accountability requires an extended exchange of ideas, which is dialogical, or comprised of multiple logics - the logic of law, and the logic of software design [22]. Speakers in a dialogue can presuppose certain ideas are known to the other party as background information, called *common ground* [20], [107]. Divergences from what is mutually known, including: *assumptions*, or what a speaker assumes is part of common ground and thus knowledge that is accessible to the other speaker; *presumptions*, or what a speaker believes is mutually believed by both parties; and *pretenses*, or beliefs that are false or ambitious in light of facts [107]. When the logic of law and software engineering conflict with one another, the exchange shifts to dialectical reasoning, in which arguments, points and counterpoints are constructed [22]. One cannot simply take the union of common grounds in law and software engineering to support design discussions and establish legal validity. Legal analysts and software engineers must instead construct dialogical “bridges” with intentional scaffolding so that each counterparty can surface and communicate assumptions, evaluate presumptions and avoid pretenses. Technical design details, which are taken for granted, can limit a legal analyst’s access to potential public harms and risk mitigations. Without such bridges, legal risk will influence how legal analysts sustain and amplify their own pretenses in ways that engineers fail to comprehend, wherein one party – the legal analyst who seeks to minimize risk – is left to assume monologic control of prohibitions over design choices.

In requirements engineering, past research to understand tacit knowledge [35] and common ground [109] provide direction, while Legal Accountability introduces new challenges: (1) how to support bi-directional communication, wherein legal analysts co-design with software engineers; and (2) how to interrelate the specialized forms of reasoning in which legal analysts and software engineers are separately trained. Notations and models have been proposed for documenting requirements during early design discourse [25], [90], and goal models in particular have been adapted to model legal requirements [39], yet to our knowledge no complementary models for documenting legal reasoning during discourse have been proposed in software engineering.

C. Education and Expertise

- **Challenge #3: How to train engineers and lawyers in cross-functional areas?**

Successful design reviews depend on participating software engineers and lawyers having a minimal, baseline understanding of each other's domain. For example, it is important that software engineers have a basic understanding of the legal context, the life cycle of law, and its potential scope from jurisdiction to jurisdiction. Software engineers must understand that law and regulation is frequently crafted to be purposefully vague or ambiguous about—or silent on—key operative features, such that interpretation in light of the full legal context is required to distill the law's full effect. Legal analysts must have a basic understanding of computer organization, distributed systems, and the software development life cycle, in particular, how requirements relate to architecture and quality assurance, and how these relationships are used perform validation and verification. These prerequisites raise questions about the breadth of knowledge, expertise, and education sufficient for stakeholders to effectuate Legal Accountability in practice.

D. Legal Ambiguity and Design Uncertainty

- **Challenge #4: How to synthesize requirements from purposeful ambiguity and overall legal intent?**
- **Challenge #5: How to assure system behavior in unspecified components?**

Requirements engineering research in ambiguity is extensive. The general approach is to acquire syntactic and semantic mappings from potentially ambiguous text to possible denotations [34]. While the challenges of identifying ambiguous terms in legal texts is important and has received critical attention and tool support [29], [79], [94], Legal Accountability draws attention to other research questions about when ambiguity is purposeful to provide design flexibility, and in what form a design is best to comply. For example, when determining which software engineering decisions are legally reasonable. This is an area of frustratingly high uncertainty for software engineers.

U.S. law frequently relies on "reasonableness" as an appropriate standard of conduct. For example, the U.S. FTC has brought enforcement action against companies who fail to provide reasonable data security. While the FTC has recently clarified "reasonable security" by enumerating specific safeguards in enforcement orders [13], it has yet to provide a firm definition. Companies typically formulate their own definitions based on industry norms and standards and what the FTC finds to be unreasonable. New research to define reasonable standards should consider: how a solution achieves broader legal goals, the level of expertise of the designer, available marketplace solutions and their technical limitations, the cost of a solution, and how reasonable standards change over time.

Laws that increase transparency shed new light on design uncertainty, which concerns situations where the law is clear, but whether and how the design complies with the law is unclear. This has generally been observed as a challenge to technical professionals [43] and software engineers [78] who analyze one or two requirements. In more advanced cases, distributed systems involving third-party components, which are typically protected by intellectual property law or as trade secrets, and the need for explainability in machine learning [18], [19], [66] introduce design ambiguity. Computational techniques to validate whether third-party components, designs and operations conform to first-party requirements are needed.

E. Risk Analysis and Harm

- **Challenge #6: How to model and estimate risk for abstract and prospective harms?**

Legal and engineering systems both assess and manage risk while reducing harms, although, sources of risk and what constitutes harm are not always in alignment. In law, *regulatory risk* concerns the "financial loss exposure arising from the [likelihood] that regulatory agencies will make changes in the current rules, or impose new rules," and *regulatory compliance risk* concerns exposure to losses "when business rules are not followed" [33]. In software engineering, *risk* is defined as the "combination of the probability of an abnormal event or failure and the consequence(s) of that event or failure to a system's components, operators, users, or environment" (ISO/IEC 24765:2010). Abnormal events may include the "inability of a product to perform a required function or its inability to perform within previously specified limits," called *failures*, or "a situation with a potential for human injury, damage to health, property, or the environment," called *harms* (ISO/IEC 24765:2010).

Requirements engineering research examines failures and *hazards*, or the "source of potential harm" (ISO/IEC 24765:2010). This includes goal-modeling methods to identify failures and hazards, called obstacles, and their countermeasures to reduce harm [3], [70]. Enhanced goal models can account for external sources of uncertainty, or risk likelihood, and be used to identify the most problematic, low-level goals [17]. Security requirements methods, e.g., SQUARE [80], include steps to identify misuse [102] and abuse cases [81] that document security-related hazards, before performing risk assessments to prioritize hazards based on likelihood and severity. Security risk covers confidentiality, integrity and availability [47], whereas privacy risk concerns privacy harms as they are experienced by data subjects [6].

Techniques have been proposed in safety-critical systems to address safety, risk and hazard analysis [116]. Method improvements to reduce harms by safety-critical systems include reducing communication gaps in cross-functional teams [75]. Certification processes to demonstrate compliance with standards often require evidence collection, however, some processes have been criticized for yielding "statements on paper," while actual requirements are embedded as artifact implementations and known only to safety engineers [76].

While prior research focuses mainly on failures and hazards, more work is needed to understand harms. Because law holds companies accountable to definitions of harm, methods that link requirements to harm are likely to inform lawmakers on better ways to legislate, and inform regulators on better ways to interpret law. This is increasingly true when software affects abstract harms. In law, harm typically encompasses physical harm to body or property, harm to one's reputation, emotional harm and harm due to violations of conferred rights. Recently, software has been designed to manipulate emotions [67], which shows how requirements can creep into more abstract harms, like emotions [93]. In the data processing, harm is often defined in terms of future uses of data, including potential for discrimination, manipulation, inadequate notice, lost control, and chilling effects. Such uses often produce relatively small effects such as frustration, aggravation, anxiety, inconvenience that are nevertheless experienced by a large number of people [23]. Courts struggle to recognize certain data processing harms[2], yet these harms inform data processing regulation. Therefore, new research on personal [91] amd human values [45], [82], and their relation to socio-political beliefs [114] can inform how regulators define and regulate harm in law.

VI. Limitations

This perspective is limited to data processing law. Detailed and mature methods and tools for legal compliance exist in other domains, such as aviation, automotive and medicine. We believe this is due in part in the U.S. to several reasons: there is a longer history of regulating safety in these domains; there are established and autonomous U.S. regulatory agencies with almost exclusive regulatory authority in these domains; and there is a clearer definition of harm in these domains, e.g., bodily harm, mental harm and financial harm. While much can be learned from these domains to support legal compliance in data processing, these differences also limit the effectiveness of knowledge transfer. The diversity and speed of innovation in data processing limits what regulators can accomplish through rulemaking, and thus explains why data processing law includes purposeful ambiguity. This difference requires software designers to bear more of the burden of specifying their own processes in the context of their software. Thus, we believe data processing law presents a starting point where new methods and tools can arise with a better fit to less prescriptive design contexts, while borrowing best practices from domains with more extensive regulation.

Second, the perspective is limited to companies operating in the United States. While companies who operate within the U.S. and outside the U.S. also need to comply with laws in non-U.S. jurisdictions, such as the GDPR, the authors' expertise is primarily with U.S. law and the U.S. legal system. That said, we believe there can be similarity across legal systems and that this perspective could be a starting point for researchers in non-U.S. jurisdictions to critique, extend and propose alternatives to this perspective to accommodate a broader vision of Legal Accountability. Concerns that should be considered across jurisdictions that primarily affect the construction and interpretation of the legal context include: (1) the extent of digitization of laws and court records, e.g., made available through e-justice systems [72], [117], which allows legal analysts and software engineers to more easily itemize and trace legal requirements, legal precedents, etc. into a Legal Accountability framework. (2) Legal systems, including courts, adjudicate legal claims using different procedures and background. Common law systems that rely on precedent, or a hierarchical court system, to reason about standing or violations of law suggest a different interpretation of the legal context by analysts and engineers than a civil law system, where each case is reviewed independently against the statute or regulation. (3) Finally, who is charged with enforcing the law, what authority that agency has been granted, and what resources the have available to perform enforcement can affect how legal analysts prioritize legal risk.

[2]TransUnion LLC v. Ramirez, 594 U.S. (2021)

VII. Conclusion

As software increasingly pervades nearly every aspect of daily life, software engineers need new tools to reduce the compliance burden and ensure that laws and policies are integrated into design, early. Specific challenges to this integration include the cultural and expertise misalignment between law and engineering. In this paper, we propose a new software quality, called Legal Accountability, which aims to change compliance from an *oversight activity* into a *principal design activity*. Legal Accountability is comprised of at least five properties, including legal traceability, completeness, validity, auditability and continuity. We motivate new and old cross-cutting research challenges that are introduced by this perspective by reviewing prior work in requirements and software engineering. Three implications of this proposal include that legal analysts and software engineers must learn new ways to collaborate in their decision making using shared artifacts to build shared understanding, which requires advances in cross-functional teaming; that new argumentation structures are needed to align and document legal and engineering reasoning; and that advances in curriculum development are needed to prepare law and software engineering students to design accountable systems.

References

[1] D. Alrajeh, L. Pasquale, B. Nuseibeh, "On evidence preservation requirements for forensic-ready systems," *11th Jnt. Mtg. Fnd. Soft. Engr.*, 2017 pp. 559–569.

[2] D. J. Anderson, *Kanban: successful evolutionary change for your technology business*, Blue Hole Press, 2010.

[3] Y. Asnar, P. Giorgini, J. Mylopoulos, "Goal-driven risk assessment in requirements engineering." *Req'ts Engr.* 16: 101–116, 2011.

[4] A.D. Austin, "Is the casebook method obsolete?" 6 Wm. & Mary L. Rev. 157, 157, 160 (1965)

[5] L. Bass, P. Clements, R. Kazman. *Software Architecture in Practice*, 3rd ed. Addison-Wesley, 2015.

[6] J. Bhatia, T. D. Breaux. "Empirical measurement of perceived privacy risk." *ACM Trans. Comp.-Hum. Inter.* 25(6): Article 34 (2018)

[7] J. Bhatia, T. D. Breaux, J. R. Reidenberg and T. B. Norton. "A theory of vagueness and privacy risk perception," *IEEE 24th Int'l Req'ts Engr. Conf.*, 2016, pp. 26-35.

[8] E. Bjarnason, P. Runeson, M. Borg, M. Unterkalmsteiner, E. Engström, B. Regnell, G. Sabaliauskaite, A. Loconsole, T. Gorschek, R. Feldt "Challenges and practices in aligning requirements with verification and validation: a case study of six companies." *Emp. Soft. Engr.* 19: 1809–1855 (2014).

[9] G. Boella, L. Humphreys, R. Muthuri, P. Rossi and L. van der Torre. "A critical analysis of legal requirements engineering from the perspective of legal practice," *IEEE 7th Int'l W'shp Req'ts Engr. & Law*, 2014, pp. 14-21.

[10] B. W. Boehm, "A spiral model of software development and enhancement," *ACM Soft. Engr. Notes*, 11(4): 14-24 (1988).

[11] B. Boehm, R. Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison-Wesley, 2003.

[12] T.D. Breaux, A.I. Antón, K.Boucher, M. Dorfman. "Legal requirements, compliance and practice: an industry case study in accessibility." *16th IEEE Int'l Req'ts Engr. Conf.,* 2008, pp. 43-52.

[13] T. D. Breaux, D. L. Baumer, "Legally 'reasonable' security requirements: A 10-year FTC retrospective," *Comp. & Sec.*, 30(4): 178-193, 2011.

[14] T.D. Breaux, D.G. Gordon. "Regulatory requirements traceability and analysis using semi-formal specifications." *Req'ts Engr.: Fnd. Soft. Qual.* LNCS 7830 (2013).

[15] T. D. Breaux, D. G. Gordon, N. Papanikolaou and S. Pearson, "Mapping legal requirements to IT controls," *6th Int'l W'shp Req'rs Engr. & Law*, 2013, pp. 11-20.

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

[17] A. Cailliau, A. van Lamsweerde, "Handling knowledge uncertainty in risk-based requirements engineering," *IEEE 23rd Int'l Req'ts Engr. Conf.*, 2015, pp. 106-115.

[18] L. Chazette, W. Brunotte, T. Speith, "Exploring explainability: a definition, a model, and a knowledge catalogue," *IEEE 29th Int'l Req'ts Engr. Conf.*, 2021, pp. 197-208.

[19] L. Chazette, K. Schneider, "Explainability as a non-functional requirement: challenges and recommendations." *Req'ts Engr.* 25: 493–514 (2020)

[20] H.H. Clark. *Using Language.* Cambridge: Cambridge University Press.

[21] J. Cleland-Huang, A. Czauderna, M. Gibiec, J. Emenecker. "A machine learning approach for tracing regulatory codes to product specific requirements." *32nd ACM/IEEE Int'l Conf. Soft. Engr.,* 2010, pp. 155–164.

[22] R.T. Craig, "Communication theory as a field," *Comm. Theory*, 9(2): 119–161 (1999).

[23] D.K. Citron, D.J. Solove, "Privacy harms," 102 B.U. L. Rev. (2022) (forthcoming)

[24] M. Daneva, A. Herrmann, L. Buglione, "Coping with quality requirements in large, contract-based projects." *IEEE Soft.*, 32(6): 84-91, 2015.

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

[26] L. Daubner, R. Matulevičius "Risk-oriented design approach for forensic-ready software systems," *16th Int'l Conf. on Avail., Rel. & Sec.*, 48, (2021)

[27] C. Dwork. "Differential privacy: a survey of results." *Int'l Conf. Theory & Appl. Models Comp.*, 2008.

[28] A.C. Edmondson, J-F. Harvey, J-F. "Cross-boundary teaming for innovation: Integrating research on teams and knowledge in organizations," *Human Res. Mgmt. Rev.* 28: 347-360 (2018).

[29] S. Ezzini, S. Abualhaija, C. Arora, M. Sabetzadeh and L. C. Briand, "Using domain-specific corpora for improved handling of ambiguity in requirements," *IEEE/ACM 43rd Int'l Conf. Soft. Engr.*, 2021, pp. 1485-1497.

[30] E.A. Farnsworth, *An Introduction to the Legal System of the United States*, Steve Sheppard ed., 4th ed. 2019.

[31] R.M. Fischl, J.R. Paul. *Getting to maybe: how to excel on law school exams.* Carolina Acad. Press, 1999.

[32] K. Forsberg, H. Mooz. "The relationship of system engineering to the project cycle," *The 12th INTERNET World Cong. Proj. Mgmt.*, 1991.

[33] B.A. Garner (ed), *Black's Law Dictionary*, 11th ed., 2019.

[34] V. Gervasi, A. Ferrari, D. Zowghi, P. Spoletini. "Ambiguity in requirements engineering: towards a unifying framework," *Soft. Engr. to Form. Mthd. & Tools, & Back*, 2019, pp 191-210.

[35] V. Gervasi, R. Gacitua, M. Rouncefield, P. Sawyer, L. Kof, L. Ma, P. Piwek, A. de Roeck, A. Willis, H. Yang, B. Nuseibeh. "Unpacking tacit knowledge for requirements engineering," *Mng'ing Req'rs Knowl.*, 2013, pp 23-47.

[36] S. Ghanavati, D. Amyot and L. Peyton, "Comparative analysis between document-based and model-based compliance management approaches," *IEEE W'shp Req'ts Engr. & Law*, 2008, pp. 35-39.

[37] S. Ghanavati, D. Amyot and L. Peyton, "Compliance analysis based on a goal-oriented requirement language evaluation methodology," *17th IEEE Int'l Req'ts Engr. Conf.*, 2009, pp. 133-142.

[38] S. Ghanavati, A. Rifaut, E. Dubois and D. Amyot, "Goal-oriented compliance with multiple regulations," *IEEE 22nd Int'l Req'ts Engr. Conf.*, 2014, pp. 73-82.

[39] S. Ghanavati, D. Amyot, A. Rifaut, "Legal goal-oriented requirement language (legal GRL) for modeling regulations." *6th Int'l W'shp Mod'ing Soft. Engr.*, 2014, pp. 1-6.

[40] M. Glinz, "On non-functional requirements," *15th IEEE Int'l Req'ts Engr. Conf.*, 2007, pp. 21-26.

[41] D. G. Gordon and T. D. Breaux, "Reconciling multi-jurisdictional legal requirements: A case study in requirements water marking," *20th IEEE Int'l Req'ts Engr. Conf.*, 2012, pp. 91-100.

[42] D. G. Gordon and T. D. Breaux, "Assessing regulatory change through legal requirements coverage modeling," *21st IEEE Int'l Req'ts Engr. Conf.*, 2013, pp. 145-154.

[43] D. G. Gordon and T. D. Breaux, "The role of legal expertise in interpretation of legal requirements and definitions," *IEEE 22nd Int'l Req'ts Engr. Conf.*, 2014, pp. 273-282.

[44] J. Guo, J. Cheng and J. Cleland-Huang, "Semantically enhanced software traceability using deep learning techniques," *IEEE/ACM 39th Int'l Conf. Soft. Engr.*, 2017, pp. 3-14.

[45] M. Harbers, C. Detweiler, M.A. Neerincx. "Embedding stakeholder values in the requirements engineering," *Req'ts Engr.: Fnd. Soft. Qual.* LNCS 9013 (2015).

[46] C.B. Haley, J.D. Moffett, R. Laney, B. Nuseibeh. "Arguing security: validating security requirements using structured argumentation," *3rd Symp. Req'ts Engr. Info. Sec.*, 2005.

[47] C. Haley, R. Laney, J. Moffett and B. Nuseibeh, "Security requirements engineering: a framework for representation and analysis," *IEEE Trans. Soft. Engr.*, 34(1): 133-153 (2008).

[48] A. Hess, J. Tamanini and S. Storck, "From screenplays to podcasts - new perspectives on improving requirements communication in interdisciplinary teams," *IEEE 29th Int'l Req'ts Engr. Conf.*, 2021, pp. 162-172.

[49] W.N. Hohfeld, "Some fundamental legal conceptions as applied in judicial reasoning." *Yale L. J.* 23(1):16–59 (1913).

[50] X. Huang, J. J. Po-An Hsieh, W. He. "Expertise dissimilarity and creativity: the contingent roles of tacit and explicit knowledge sharing," *J. Appl. Psych.* 99(5): 816-30 (2014).

[51] Health and Human Services, U.S. Department of. "Guidance for the content of premarket submissions for software contained in medical devices," May 11, 2005.

[52] Health and Human Services, U.S. Department of. "Guidance regarding methods for de-identification of protected health information in accordance with the health insurance portability and accountability act privacy rule," Nov. 26, 2012.

[53] S. Ingolfo, A. Siena, J. Mylopoulos, A. Susi, A. Perini, "Arguing regulatory compliance of software requirements," *Data & Know. Engr.* 87: 279-296 (2013).

[54] S. Ingolfo, I. Jureta, A. Siena, A. Perini, A. Susi A. "Nòmos 3: legal compliance of roles and requirements." In: Yu E., Dobbie G., Jarke M., Purao S. (eds) *Conceptual Modeling.* LNCS, vol 8824 (2014).

[55] M. Jarke, K. Pohl, S. Jacobs, J. Bubenko, P. Assenova, P. Holm, G. Spanoudakis. "Requirements engineering: an integrated view of representation, process, and domain." *Euro. Soft. Engr. Conf.*, 1993, pp. 100-114.

[56] A. J. I. Jones, M. Sergot. "Deontic logic in the representation of law: towards a methodology." *Art. Intel. & Law* 1: 45-64 (1992).

[57] I. Jureta, J. Mylopoulos and S. Faulkner. "Analysis of multi-party agreement in requirements validation," *IEEE Int'l Conf. Req'ts Engr.*, 2009, pp. 57-66.

[58] G. Kaur. "Analyzing email archives to better understand legal requirements," *2nd IEEE Int'l W'shp Req'ts Engr. & Law*, 2009, pp. 21-26.

[59] G.M. Kanchev, P.K. Murukannaiah, A.K. Chopra, P. Sawyer, P. "Canary: extracting requirements-related information from online discussions," *IEEE 25th Int'l Req'ts Engr. Conf.*, 2017, pp. 31-40.

[60] E. Kempe and A. Massey, "Perspectives on regulatory compliance in software engineering," *IEEE 29th Int'l Req'ts Engr. Conf.*, 2021, pp. 46-57.

[61] L.M. Khan, N.J. Phillips, R. Chopra, R.K. Slaughter, C.S. Wilson, C. S. "FTC report to congress on privacy and security," Sep. 13, 2021.

[62] B. Kitchenham, S. Charters. "Guidelines for performing systematic literature reviews in software engineering," EBSE Tech. Rep. EBSE-2007-01, 2007.

[63] J. Kotlarsky, B. van den Hooff, L. Houtman. "Are we on the same page? Knowledge boundaries and transactive memory system development in cross-functional teams," *Comm. Res.*, 42(3): 319-344 (2012).

[64] J. Kotlarsky, H. Scarbrough, I. Oshri. "Coordinating expertise across knowledge boundaries in offshore-outsourcing projects: the role of codification." *MIS Quarterly* 38(2), 607-A5 (2014).

[65] C.W. Kneupper, "Teaching argument: an introduction to the Toulmin model." *College Comp. & Comm.*, 29(3): 237–241 (1978).

[66] M. A. Köhl, K. Baum, M. Langer, D. Oster, T. Speith and D. Bohlender, "Explainability as a non-functional requirement," *IEEE 27th Int'l Req'ts Engr. Conf.*, 2019, pp. 363-368.

[67] A.D.I. Kramer, Jamie E. Guillory, and Jeffrey T. Hancock, "Experimental evidence of massive-scale emotional contagion through social networks." *Proc. Nat'l Aca. Sci.*, 111 (24): 8788-8790 (2014).

[68] J.A Kroll, J. Huery, S. Barocas, E.W. Felton, J.R. Reidenberg, D.G. Robinson, H. Yu. "Accountability algorithms." 165 U. Pa. L. Rev. 633 (2017).

[69] W. Kunz, H. Rittel. "Issues as elements of information systems." Wk'ing Paper No. 131, Inst. Urban & Reg. Dev., Univ. Cal., Berkeley, 1970.

[70] A. van Lamsweerde, E. Letier, "Handling obstacles in goal-oriented requirements engineering," *IEEE Trans. Soft. Engr.*, 26(10): 978-1005 (2000).

[71] G. Lucassen, F. Dalpiaz, J. Martijin, E.M. van der Werf. "The use and effectiveness of user stories in practice." *Int'l Wk'ing Conf. Req'ts Engr.: Fnd. Soft. Qual.*, 2016.

[72] G. Lupo, J. Bailey. "Designing and implementing e-justice systems: some lessons learned from EU and Canadian examples." *Laws* 3: 353-387 (2014).

[73] C. Ma. "Self-regulation versus government regulation: an externality view," *J. Req. Econ.* 58:166-183 (2020).

[74] A. Majchrzak, P.H.B. More, S. Faraj. "Transcending knowledge differences in cross-functional teams." *Org. Sci.* 23(4): 951-970 (2012).

[75] L. E. G. Martins, T. Gorschek, "Requirements engineering for safety-critical systems: overview and challenges." *IEEE Soft.*, 34(4): 49-57 (2017).

[76] L. E. G. Martins, T. Gorschek, "Requirements engineering for safety-critical systems: an interview study with industry practitioners," *IEEE Trans. Soft. Engr.*, 46(4): 346-361 (2020)

[77] A. K. Massey, P. N. Otto and A. I. Antón, "Prioritizing legal requirements," *2nd Int'l W'shp Req'ts Engr. & Law*, 2009, pp. 27-32.

[78] A. K. Massey, P. N. Otto and A. I. Antón, "Evaluating legal implementation readiness decision-making," *IEEE Trans. Soft. Engr.*, 41(6): 545-564 (2015)

[79] A. K. Massey, R. L. Rutledge, A. I. Antón and P. P. Swire, "Identifying and classifying ambiguity for regulatory requirements," *IEEE 22nd Int'l Req'ts Engr. Conf.*, 2014, pp. 83-92.

[80] N. R. Mead, T. Stehney. "Security quality requirements engineering (SQUARE) methodology," *ACM Soft. Engr. Notes* 30(4): 1–7 (2005)

[81] J. McDermott, C. Fox, "Using abuse case models for security requirements analysis." *15th Annual Comp. Sec. Appl. Conf.*, 1999.

[82] D. Mougouei, H. Perera, W. Hussain, R. Shams, J. Whittle, "Operationalizing human values in software: a research roadmap," *26th ACM Symp. Fnd. Soft. Engr.*, 2018, pp. 780–784.

[83] J. Mylopoulos, L. Chung, B. Nixon. "Representing and using nonfunctional requirements: a process-oriented approach," *IEEE Trans. Soft. Engr.* 18(6): 483-497 (1992).

[84] J. Mylopoulos, L. Chung, E. Yu. "From object-oriented to goal-oriented requirements analysis," *Comm. ACM* 42(1): 31-37 (1999).

[85] J. Mylopoulos, L. Chung, S. Liao, H. Wang and E. Yu, "Exploring alternatives during requirements analysis," *IEEE Soft.* 18(1): 92-96 (2001).

[86] N. Zeni, N. Kiyavitskaya, L. Mich, J. R. Cordy, and J. Mylopoulos, "GaiusT: supporting the extraction of rights and obligations for regulatory compliance," *Req'ts Engr.* 20(1): 1–22 (2015).

[87] C. Ncube and S. L. Lim, "On systems of systems engineering: a requirements engineering perspective and research agenda," *IEEE 26th Int'l Req'ts Engr. Conf.*, 2018, pp. 112-123.

[88] P. N. Otto and A. I. Anton, "Addressing legal requirements in requirements engineering," *15th IEEE Int'l Req'ts Engr. Conf.*, 2007, pp. 5-14.

[89] L. Pasquale, D. Alrajeh, C. Peersman, T. Tun, B. Nuseibeh and A. Rashid, "Towards forensic-ready software systems," *IEEE/ACM 40th Int'l Conf. Soft. Engr.: New Ideas & Emer. Tech. Res.*, 2018, pp. 9-12.

[90] C. Potts, K. Takahashi, A.I. Antón. "Inquiry–based requirements analysis," *IEEE Soft.* 11(2): 21-32 (1994)

[91] R. Proynova, B. Paech, A. Wicht, T. Wetter. "Use of personal values in requirements engineering: a research preview." *W'ing Conf. Req'ts Engr.: Fnd. Soft. Qual.*, LNCS 6182 (2010).

[92] B. Ramesh, V. Dhar. "Supporting systems development by capturing deliberations during requirements engineering," *IEEE Trans. Soft. Engr.*, 18(6): 498-510 (1992).

[93] I. Ramos, D.M. Berry, "Is emotion relevant to requirements engineering?" *Req'ts Engr.* 10: 238–242 (2005).

[94] J.R. Reidenberg, J. Bhatia, T.D. Breaux, T.B. Norton, "Ambiguity in privacy policies and the impact of regulation," *J. L. Studs.* 45(S2) (2016).

[95] J. Ryan, A. Toner. "Europe's enforcement paralysis: ICCL's 2021 report on the enforcement capacity of data protection authorities." Irish Council for Civil Liberties, 2021.

[96] F. Rivera-Ortiz, L. Pasquale "Automated modelling of security incidents to represent logging requirements in software systems," *15th Int'l Conf. Avail., Rel. & Sec.*, 2020, Art. No.: 35.

[97] N. Sannier et al., "Legal markup generation in the large: an experience report," *IEEE 25th Int'l Req'ts Engr. Conf.*, 2017, pp. 302-311.

[98] N. Sannier, M. Adedjouma, M. Sabetzadeh, et al. "An automated framework for detection and resolution of cross references in legal texts." *Req'ts Engr.* 22, 215–237 (2017).

[99] L. da Silva Barboza, G. A. de A. Cysneiros Filho and R. A. C. de Souza, "Towards a legal compliance verification approach on the procurement process of IT solutions for the Brazilian Federal Public Administration," *IEEE 7th Int'l W'shp Req'ts Engr. & Law*, 2014, pp. 39-40.

[100] R. Sethi, D.C. Smith, W. Park. "Cross-functional product development teams, creativity, and the innovativeness of new consumer products," *J. Mark. Res.* 38(1): 73-85 (2001).

[101] A. Siena, I. Jureta, S. Ingolfo, A. Susi, A. Perini, J. Mylopoulos, "Capturing variability of law with Nómos 2." In: Atzeni P., Cheung D., Ram S. (eds) *Conceptual Modeling*, 2012, LNCS 7532 (2012).

[102] G. Sindre, A.L. Opdahl. "Eliciting security requirements with misuse cases." *Req'ts Engr.* 10: 34–44 (2005).

[103] A. Sleimi, M. Ceci, M. Sabetzadeh, L. C. Briand and J. Dann, "Automated recommendation of templates for legal requirements," *IEEE 28th Int'l Req'ts Engr. Conf.*, 2020, pp. 158-168.

[104] A. Sleimi, M. Ceci, N. Sannier, M. Sabetzadeh, L. Briand and J. Dann, "A query system for extracting requirements-related information from legal texts," *IEEE 27th Int'l Req'ts Engr. Conf.*, 2019, pp. 319-329.

[105] A. Sleimi, N. Sannier, M. Sabetzadeh, L. Briand and J. Dann, "Automated extraction of semantic legal metadata using natural language processing," *IEEE 26th Int'l Req'ts Engr. Conf.*, 2018, pp. 124-135.

[106] D.J. Solove, W. Hartzog. "The FTC and the new common law of privacy," *Columbia L. Rev.* 114(583) (2014).

[107] R. Stalnaker. "Common ground," *Ling. & Phil.* 25(5/6): 701-721 (2002)

[108] J. -P. Steghöfer, B. Koopmann, J. S. Becker, M. Törnlund, Y. Ibrahim and M. Mohamad, "Design decisions in the construction of traceability information models for safe automotive systems," *IEEE 29th Int'l Req'ts Engr. Conf.*, 2021, pp. 185-196.

[109] A. Sutcliffe, "Collaborative requirements engineering: bridging the gulfs between worlds," *Intent'l Persp. Info. Sys. Engr.*, 2010, pp 355-376.

[110] K. Schwaber, J. Sutherland. *The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game,* 2014. http://www.scrum.org

[111] G. Szulanski, D. Ringov, R.J. Jensen. "Overcoming stickiness: how the timing of knowledge transfer methods affects transfer difficulty," *Org. Sci.* 27(2): 304-322 (2016)

[112] B. Tabrizi. "75% of cross-functional teams are dysfunctional," *Harvard Bus. Rev.*, June 23, 2015.

[113] H. Taneja. "The era of 'move fast and break things' is over," *Harvard Bus. Rev.*, January 22, 2019.

[114] S. Thew, A. Sutcliffe. "Value-based requirements engineering: method and experience." *Req'ts Engr.* 23: 443–464 (2018).

[115] S. Toulmin. *The Uses of Argument.* Camb. Univ. Press, 1969. .

[116] J. Vilela, J. Castro, L. E. G. Martins, T. Gorschek, "Integration between requirements engineering and safety analysis: A systematic literature review," *J. Sys. & Soft.* 25: 68-92 (2017).

[117] N. Yavuz, N. Karkin, M. Yildiz. "E-Justice: a review and agenda for future research." *Sci. Fnd. Dgt'l Gov. & Trans. Pub. Admin. & Info. Tech.*, 38: 385-414, 2022.