Modeling, Analyzing and Communicating Regulatory Ambiguity: An Empirical study

Evelyn Kempe, Samin Semsar, Aaron Massey, Sreedevi Sampath, Carolyn Seaman*
Department of Information Systems, University of Maryland, Baltimore County
Baltimore, MD, USA
{ekempe1,samin.semsar,akmassey,sampath,cseaman}@umbc.edu

ABSTRACT

Regulations outline high-level guidance or expectations for a profession or industry. Analyzing laws or regulations is one way a software developer would derive and document regulatory compliance requirements within their software design. However, ambiguities within regulations can make it challenging to define technical software design specifications for regulatory requirements. Further, due to the subjective nature of ambiguous phrasing within a law or regulation, the interpretation of the legal text can differ based on the interpreter's perspective. Our study examines whether software developers can analyze regulatory ambiguities as a group using our modeling process and our online Ambiguity Heuristics Analysis Builder (AHAB) tool.

Eleven participants formed three groups and modeled ambiguities within a regulation using our process and tool. Modeling regulatory ambiguity, while difficult for our participants, allowed them to communicate potential issues, ask meaningful questions, and deepen their knowledge of the regulation. Ambiguity modeling allows developers to articulate interpretation and compliance issues with the laws to other parties (i.e., lawyers) and document this requirement analysis step for future use. Documenting these intermediate steps is rarely highlighted in requirement analysis. However, it is useful to negotiate with regulators, avoid negligence, and show due diligence toward regulatory compliance. It can also lead to clarifying guidance software developers need to make better, more compliant choices during software design.

CCS CONCEPTS

• **Software and its engineering**; • **Computing methodologies** → **Model development and analysis**;

KEYWORDS

regulatory ambiguity analysis, modeling, software development

*All authors contributed equally to this research.

MO2RE 2024, April 16, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0569-4/24/04
https://doi.org/10.1145/3643666.3648576

ACM Reference Format:
Evelyn Kempe, Samin Semsar, Aaron Massey, Sreedevi Sampath, Carolyn Seaman. 2024. Modeling, Analyzing and Communicating Regulatory Ambiguity: An Empirical study. In *Workshop on Multi-disciplinary, Open, and RElevant Requirements Engineering (MO2RE 2024), April 16, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3643666.3648576

1 INTRODUCTION

Laws and regulations, including those that govern how software must perform, contain ambiguity. Software developers struggle with this ambiguity during regulatory requirements analysis. A lack of regulatory understanding makes it difficult or impossible for software developers to explain, interpret, or demonstrate the elements of their software design that ensure compliance. However, software developers can ask meaningful questions and explain confusions regarding regulatory ambiguities if they have a way to analyze, model, and describe those ambiguities. This type of analysis can lead to more productive discussions with software development stakeholders. It can also lead to actionable design solutions that demonstrate meaningful efforts for regulatory compliance and due diligence within the development process.

Our study examines how software developers analyze and model ambiguities within a regulation, addressing the following questions:

- RQ1: Can software developers analyze and model regulatory ambiguities?
- RQ2: What are the difficulties a software developer encounters when analyzing and modeling regulatory ambiguities individually and as a group?
- RQ3: Is there value in analyzing and modeling ambiguities during requirements analysis from a software developer's stance?

Our modeling approach allows developers to reason about regulatory ambiguity separately from their system under development and then trace decisions made to resolve regulatory ambiguities to affected requirements specifications. To evaluate our approach, we recruited eleven participants with backgrounds in software design to form groups and model ambiguities in regulation using our process and tool. We wanted to see if they could accomplish the modeling task individually and as a group (i.e., RQ1). We also wanted to identify the difficulties they generally encountered and their effect on the analysis process (i.e., RQ2). Lastly, we wanted to see if our participants saw value in modeling ambiguities (i.e., RQ3). Our results show that software developers can analyze and model regulatory ambiguities. In addition, developers can discuss their rationale with peers and agree on what they find ambiguous within a legal text. The group can present their analysis and models to other parties for further guidance and resolution. This process offers a

way to document the mitigation of ambiguity in compliance-related regulatory requirements for software design.

Our work offers several contributions to regulatory compliance and ambiguity analysis research. First, we expand upon Massey et al.'s previous work on ambiguity identification and classification [7–9] by operationalizing it within a modeling methodology as a strategy for analysis. Second, we offer a methodology that involves both individual and group regulatory ambiguity analysis, drawing on the strengths of both modes. Lastly, we offer insight into developers' reasoning and heuristics when performing this kind of analysis, which is necessary to effectively offer further support for the process. Overall, our analysis advances the development of our ambiguity analysis methodology and will facilitate further tool and artifact development. The rest of the paper discusses related work, the ambiguity modeling process we have refined, the design of the case study, results from our analysis, discussion of those results, threats to validity of our results, and finally conclusions.

2 RELATED WORK

Software developers face an inherent challenge with regulatory analysis because of ambiguities in laws and regulations [11]. Understanding and interpreting regulatory ambiguities correctly in law can be the difference between regulatory compliance and non-compliance. Breaux and Norton's work on legal accountability [2] focuses on how much a software organization is accountable to law and regulations. They discuss how aligning legal interpretation and expectations amongst different stake-holder groups is challenging. Therefore, laws and their applications to software design must be assessed in context and applied when applicable. Developers start to achieve legal accountability by understanding legal goals, identifying implementation concerns, and documenting them. Our work takes steps to address the need expressed by Breaux and Norton, by providing and assessing a methodology to help developers do this.

Prior work has proposed methodologies for analyzing regulations to include ambiguities [1, 3, 5, 8, 9]. Amaral et al. [1] and Ezzini et al. [5] examine solutions to automate compliance and ambiguity checking in regulatory requirements, adaptable for enforcement. A prelude to our work, Massey et al. [8, 9] recruited participants to examine their rationale in identifying and classifying ambiguities from regulatory texts using an ambiguity taxonomy. Our work extends that study by using the same taxonomy to analyze regulatory ambiguities, embedded in a larger analysis and modeling process. Further, we have included group analysis of regulatory ambiguity as a focus of our study versus Massey et al., who focused on individual ambiguity analysis. This expansion of the process to include group work is a step towards practical application, as compliance and regulatory requirement development is rarely done by a single person. Typically stakeholders from the design team, legal team and security team weigh in on the interpretation and examine how the software implementation meets the regulatory requirements. To facilitate online group collaboration, our participants used an online ambiguity modeling tool, Ambiguity Heuristic Analysis Builder (AHAB), described in Section 4.2. AHAB facilitates Massey et al.'s regulatory ambiguity analysis methodology and allows users to model and document ambiguities alongside other artifacts in software design [7]. To our knowledge, documenting these intermediate analysis steps is rarely highlighted in requirement analysis, but is useful to negotiate with regulators, avoid negligence, and show due diligence toward regulatory compliance.

Modeling as a tool to focus a discussion (in our case, of regulatory ambiguity) and document requirements analysis is not new. For example, Goal-oriented Requirement Language (GRL) [6] is useful for addressing non-functional requirements like regulatory requirements. Ghanavati et al. introduced a systematic method to extract legal requirements from regulations using GRL (i.e., Legal GRL) [6]. Our study builds on this previous work by using an online tool, AHAB, as an extension of Legal GRL [7]. Yet, what we try to convey through modeling is unique. To our knowledge, no one has applied any graphical modeling method to regulatory ambiguity analysis as a way to document this intermediate step.

3 AMBIGUITY MODELING PROCESS

The modeling process aims to analyze and document ambiguities within regulatory text. Through ambiguity modeling, one can examine, brainstorm, storyboard, and organize potentially confusing regulatory and compliance issues within software requirements analysis. Furthermore, modeling is a visualization of rationale. It captures the modeler's perspective on regulatory text and compliance issues. The modeler then can explain their interpretation to a third party using the model as a guide. A version of this process was first presented and analyzed by Massey et al. in 2017 [7] . We simplified the ambiguity modeling process outlined by Massey et al. [7] by removing recursive layering of ambiguity[1]. This section outlines the ambiguity modeling process as executed by our participants. Section 4 describes how this process was embedded in the larger group modeling activity in our study.

The Process: We define a regulatory ambiguity as a word or phrase within a regulation having no or multiple meanings. This definition is derived from the IEEE definitions for unambiguous[2]. The ambiguity modeling process applies our ambiguity definition, executed through five high-level steps (See Figure 1):

Step One: The first step is reading the regulatory text. The modeler can read the text in its entirety before identifying any ambiguities or they can identify ambiguities as they progress through the text on the first reading.

Step Two: When the modeler comes across a word, phrase, or paragraph that they view as ambiguous, they capture the text and begin the process to mark it as ambiguous.

Step Three: To create the ambiguity node, the modeler must expand on their reasoning as to why they view the text as ambiguous by documenting specific prescribed attributes of the ambiguity:

(1) Capture Text: Identify the ambiguous word or phrase.
(2) Ambiguity Type: Classify the captured text to an ambiguity type. Classification helps clarify the logic as to why the text is ambiguous. We provide an ambiguity taxonomy to assist the modeler in classifying the ambiguity (See Table 1 [8, 9]).
(3) Notes: The modeler further explains the logic behind identifying a regulatory text as ambiguous, beyond the classification within this attribute.

[1] Describing the recursive layering of ambiguity is out of the scope of this paper.
[2] "unambiguous: 1) Not having two or more possible meanings. 2) Not susceptible to different interpretations. 3) Not obscure, not vague. 4) Clear, definite, certain."

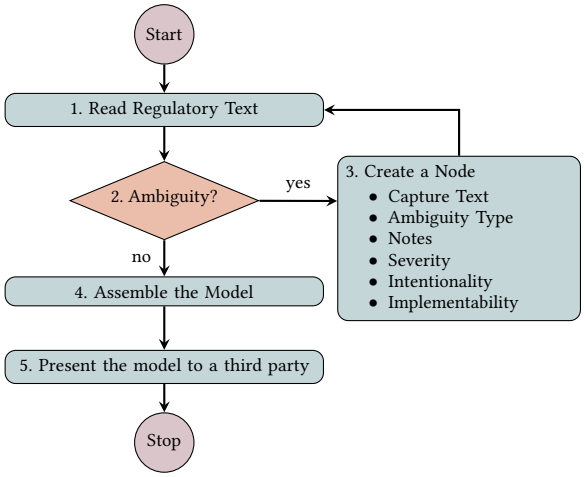

Figure 1: Ambiguity Modeling process Flowchart

(4) Severity: On a scale of 1 to 5, the severity rating indicates the degree to which the resolution of the ambiguity impacts the software design. The severity level increases if the ambiguity challenges the software design process.
(5) Intentionality: Regulators intentionally place ambiguity within regulations or laws, so as the law and its interpretation evolve, so can the regulations, including applicable technology supporting compliance with the law. Therefore, by marking Intentionality as a "Yes," the modeler recognizes that the ambiguity was intentional when written.
(6) Implementability: A "Yes" means that the ambiguity exists, but the developer can derive a software requirement specification without further resolution or clarification.

Step Four: Assembling the model involves logically organizing the created ambiguity nodes for presentation to a third party (i.e., Step Five). This organization of the model is the storyboard aspect of modeling, where the modeler highlights potential dependencies, relationships, similarities, and flow of the identified ambiguities.

Step Five: The point of the modeling process is to facilitate communication between stakeholder groups, including people not involved in building the model. This communication solicits further guidance to clarify meaning or intent within a regulation and document further action, interpretation, or decisions made to meet regulatory compliance requirements.

As seen in Figure 1, the first three steps (i.e., Reading the Regulatory Text, Identifying the Ambiguity, and Creating the Ambiguity Node) are performed iteratively, until the modeler identifies all ambiguities, if any, in the regulatory text and creates the associated ambiguity nodes. In step four, the modeler organizes the nodes, thus assembling the model. Then, the modeler proceeds to step five by presenting the model to a third party for further discussion and guidance.

4 METHODS

We conducted a pilot study in November 2021 with two people to test the study design. The primary study with eleven participants was conducted from March 21, 2022, to December 16, 2022 [3]. All participants were graduate students in Cybersecurity, Data Science, Software Engineering, or Information Systems at UMBC. They formed three case groups of 3-4 participants each. All participants were 23-30 years old and had Software Developer (11) or Analyst (2) backgrounds[4]. They reported an average of about three years of work experience in their roles (range of 1-9 years). This section discusses the Case Study design including outcomes from the Pilot Study and Case Group One, a description of the AHAB tool, and Data Collection and Analysis.

Table 1: Case Study Ambiguity Taxonomy [9]

Ambiguity Type	Definition
Lexical	A word or phrase with multiple valid meanings.
Syntactic	A sequence of words with multiple valid grammatical interpretations regardless of context.
Semantic	A sentence with more than one interpretation in its provided context.
Vagueness	A statement that admits borderline cases or relative interpretation.
Incompleteness	A grammatically correct sentence that provides too little detail to convey a specific or needed meaning.
Referential	A grammatically correct sentence with a reference that confuses the reader based on the context.

4.1 Case Study Design

Each case group in our primary study met in three online sessions with two periods of "homework" between the sessions. All participants analyzed the European Union's General Data Protection Regulation (GDPR) Article 17, the "Right to erasure" [5]. The sessions were structured as follows:

Session One was a one-on-one training session on ambiguity modeling with each participant. We provided participants access to training material and the Ambiguity Heuristics Analysis Builder (AHAB) tool(See Section 4.2 for details on the tool).The session included an overview of the case study and Ambiguity Taxonomy (See Table 1) and a "Hands-On" AHAB demo [6]. During the demo, the facilitator provided the participant with a list of ambiguity modeling tasks using the AHAB tool. The participants discussed their actions as they accomplished the tasks as the facilitator observed. This demo in Session 1 gave each participant some practice with the AHAB tool and the ambiguity modeling technique before building an ambiguity model on their own for Session Two. In addition, this session also allowed the facilitator the ability to provide technical assistance to the participant if necessary. At the end of Session 1, the participant's homework was to examine the assigned section of legal text from the GDPR, mentioned above, and create an ambiguity model before the next session.

[3]This study was reviewed and approved by the UMBC's Institutional Review Board under Protocol #984 and was partly supported by NSF SaTC Award #1938121.
[4]Software Developer: A person that builds and maintains software or IT systems. Analyst: A person who gathers and interprets data for requirements.
[5]Also known as 'Right to be forgotten' at https://gdpr-info.eu/art-17-gdpr/.
[6]Before Session 1, we gave the participant access to AHAB and tutorial material.

Observation sessions were scheduled "homework" time for participants to build ambiguity models with the facilitator observing the participant's progress. They were optional and allowed the facilitator to note any technical difficulties a participant might have with the online AHAB modeling tool.

Session Two was an online group session where the participants presented their ambiguity models to their group. Presenting the models allowed everyone to see how their peers approached the ambiguity modeling task. At the end of the session, we gave the participants a JSON file with all the group's ambiguity models. This file allowed the participants to compare ambiguity models using the AHAB tool and conduct further analysis before Session 3.

Session Three began with any updates to the models that the participants might have made since Session 2. Then, the session progressed into the group analysis with the participants attempting to achieve consensus to construct a final joint ambiguity model. At the Session's end, if the group reached a consensus, they submitted their joint ambiguity model. If not, they submitted all models in whatever state they ended up in, and the participant's role in the study concluded.

End of Case Survey [7] was an anonymous and optional 10-minute survey hosted through Qualtrics to gain participants' honest feedback on the ambiguity model process and AHAB tool.

The final structure of the Sessions described above was finalized after analyzing data from our pilot study and Case Group One. The four primary changes to our Case Study protocol design were:

(1) Expanding the Sessions from two to three online Sessions (From Pilot Study).
(2) Incorporating Ambiguity Taxonomy Table (See Table 1) into an AHAB information icon (From Pilot Study).
(3) Allowing participants the option to participate in more online "Observation" Sessions (From Case One).
(4) Adding the "end of case" survey (From Case One)

We made no additional changes to the Case Study's Protocol after Case 1. A more detailed study protocol to include the Session presentation slides are at: https://doi.org/10.6084/m9.figshare.23297717.

4.2 Ambiguity Heuristics Analysis Builder

To execute the ambiguity modeling process and facilitate an online group analysis and collaboration of regulatory ambiguity, we developed a tool, which we call The Ambiguity Heuristics Analysis Builder (AHAB). AHAB is designed to allow users to build Regulatory Ambiguity Models [7] online. The tool is written primarily in JavaScript and uses Canvas as the drawing framework. A user can access the tool through a web browser and build a model by following the process outlined in Section 3.

Figure 2 is a screenshot of AHAB with an example model. The regulatory text is on the left of the picture (i.e., Art. 17 GDPR). An example model is in the middle of Figure 2 with two linked ambiguity nodes and a start and stop node. On the top right of Figure 2 is the ambiguity node attribute box, outlining all the prescribed ambiguity attributes described in Step three of Section 3 and Figure 1. The bottom right is the modeling shapes panel, from which a shape (node) can be dragged into the canvas to expand the model further.

[7]The complete survey is found at https://doi.org/10.6084/m9.figshare.23297717

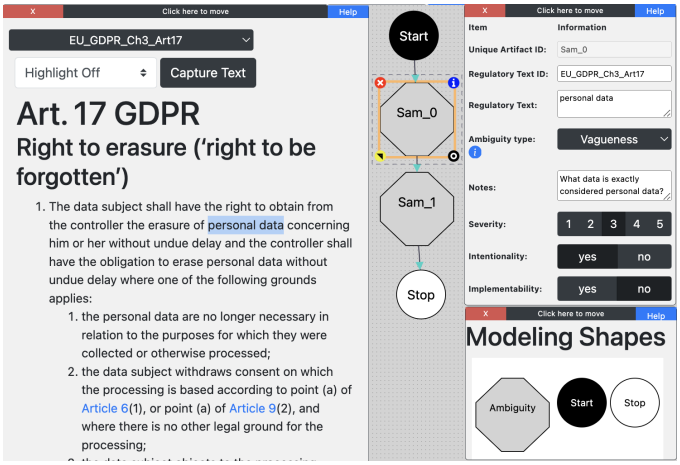

Figure 2: Example AHAB version 1 screenshot

AHAB allows for the ambiguity model to be output as a JSON file. The JSON file contains information about every Ambiguity node, including links between nodes. Two other output formats are supported: a tabular format and textual analysis. AHAB also creates a log as a text file that captures every step of making the model, including deleted ambiguity nodes or links.

AHAB has several features that assist with the group analysis of ambiguity models. One feature is the ability to import several models of the same regulatory text onto the same canvas or screen. This feature allows groups to analyze, compare, and combine different ambiguity models built against the same regulatory text without maintaining several instances of AHAB.

Another feature is the heat mapping of ambiguity nodes. The heat map feature uses coloring to let users see different aspects of the ambiguity nodes within the graphical view of the model. For example, AHAB uses a gradient color ranging from yellow (Severity 1) to orange to a darker red for higher levels of the Severity level attribute. Every attribute of the ambiguity node described in Step 3 in Section3 has a heat map instance for selection and viewing.

Overall, AHAB supports the Ambiguity modeling process for individual and group analyses of regulatory text by providing an online, accessible platform and multiple views for documentation and artifact development. To review the AHAB tool, use the following link: https://www.sixlines.org/ahab/tutorial/AHAB.html.

4.3 Data Collection and Analysis

We used data collected from three sources: 1) Online sessions with the case participants recorded through Google Meet and transcribed using Otter.ai[8]; 2) The participants' ambiguity models using AHAB; 3) The online close-out survey results[9]. All data was collected and analyzed using NVivo version 12[10].

We analyzed our data using grounded theory [4] and with-in and cross-case analysis [10]. We generated our initial coding scheme[11]

[8] https://otter.ai
[9]We added the survey after Case 1 and hosted it through Qualtrics. The complete survey is found at https://doi.org/10.6084/m9.figshare.23297717
[10] https://lumivero.com/products/nvivo/
[11]The initial coding scheme is at: https://doi.org/10.6084/m9.figshare.23297717

based on the timeline of events, delineating the three sessions in each case, and three parts of each session (i.e., the participant's preparation before the session, what happened during the session, and what happened at the end of the session). We then built content-based sub-codes under those initial sequence-based codes. We saw five themes emerge from the initial application of codes. These emerging themes became our final coding scheme (see list below) and helped generate our findings. The next section discusses our findings based on our analysis.

(1) Common reasoning for identifying an ambiguous legal text
(2) Common reasoning for classifying an ambiguous legal text
(3) Modeling difficulties
  (a) Understanding the Regulatory Text
  (b) Classifying ambiguities
  (c) Consolidating models
(4) Ambiguity analysis and discussion
(5) Importance of ambiguity modeling

5 RESULTS

This section highlights our three major findings answering our research questions.

5.1 Completing the Ambiguity Models - RQ1

Finding 1: With a tool and guidance, software developers can perform regulatory ambiguity analysis and modeling individually and as a group.

Our study participants were assigned two tasks. First, they each needed to build a regulatory ambiguity model. Second, they had to discuss and consolidate their models into one group model.

Everyone accomplished the first task. Two of our three groups accomplished the second task by consolidating their models into a group model by the end of Session 3. Group 1 was not able to accomplish this in the time allotted[12]. Some participants even reported that ambiguity modeling was easy because of the guidance and the tool, contrary to our pilot results. For example:

> **ID6** "Yes, same for me, it was really easy. [AHAB] has given various options like...the heat map selection... the text [capture], the ambiguity type, and the severity. [It] was really easy to visualize my whole model."

Given time, tools, and guidance, software developers can model and communicate concerns about an ambiguous regulatory text. Furthermore, by documenting and sharing these concerns, they can look to third parties (i.e., lawyers) for further guidance to clarify or resolve the ambiguities.

5.2 Difficulties with Ambiguity Modeling - RQ2

Finding 2: Regulatory ambiguity analysis is difficult, but the difficulties directly lead to identifying ambiguities. Discussion of these difficulties is evidence that the analysis is being done through this intermediate documentation.

Interpreting regulatory ambiguities can be difficult for software developers with no legal training. This section highlights three modeling process difficulties common among our participants.

Understanding the Regulatory Text: One difficulty in modeling ambiguities was understanding the regulatory text. Most participants found the wording or intent in the legal text difficult to understand. Consider the following comments:

> **ID 8:** "When I read this for the first time, I got confused. I did not know whether they are talking about data subject or the controller"
> **ID10:** "I specifically found understanding the document in the first try [difficult]. I would have to read it a number of times to understand what they're trying to convey. That was one difficulty"

Seven of the 11 participants used phrasing such as "confused," "unclear," or "complex" when presenting their analysis. We noted this trend as the top reason amongst our participants when identifying ambiguous text. Not all participants expressed difficulty understanding the legal text. Similarly, the survey showed mixed responses to the question about understanding the regulation difficulty[13]. Yet, when such difficulties arose, they led to progress in the analysis.

Classifying Ambiguities: Another difficulty pointed out by our participants was classifying ambiguities, i.e. assigning values to the various ambiguity attributes in AHAB, such as severity, type, etc. Take, for example, ID6's comment:

> **ID6**: "If I read a sentence initially, I [would] think it was one type of ambiguity. If I revisit the model or that text, I [would think] *"No, this is something else"*, interpreting it as [another] ambiguity type."

The survey results told a slightly different story. Four respondents to the survey disagreed with the statement: *"I found it difficult to identify and classify the ambiguities within the regulation."*. One of the five respondents agreed, however.

The participants' confidence about the modeling process at the study's end could explain the differences in the data collected. Evidence shows that the taxonomy and the AHAB tool evolved the participants' understanding of regulatory ambiguities. For example:

> **ID2:** "The tool helped me understand what exactly ambiguity is. I didn't know what the word ambiguity meant before this [study]."

Some participants used the ambiguity taxonomy definitions (See Table 1) to explain their ambiguity analysis. Consider the following comments:

> **ID11:** "...the ambiguity type is vagueness, because it is a borderline case"
> **ID1:** "I felt this was an ambiguous statement & [is] vagueness [since] it covers only a borderline cases."
> **ID4:** "This phrase, "legitimate grounds", it may have different interpretations from person to person. So I think that is a semantic ambiguity."

As the participants' understanding evolved, so did their confidence regarding the modeling process as shown in ID12's comment.

> **ID12:** "I feel like if we had a fourth session, we can [build] another model in one meeting...[it took] three sessions, [for] our [model] because we were new to

[12]Screenshots of models are available at: https://doi.org/10.6084/m9.figshare.23297717

[13]Out of the five survey responses, two agreed, one was neutral, and two disagreed.

> AHAB...I feel that will be more rapid if we [did] another article."

Some participants expressed difficulty using the ambiguity classification taxonomy during the sessions; however, some participants felt much more confident in their analysis and the modeling process by the end of the exercise. The ambiguity taxonomy and the AHAB tool were aids to that progression.

Consolidating models: The third difficulty with modeling was consolidation. Some participants highlighted that agreeing on ambiguities (identifying or classifying) for model consolidation was challenging. Take, for example, these quotes:

> **ID2:**"I think it's the agreeing with others. Trying to get their perspective [versus] your perspective is one thing [that was] difficult"
> **ID8:** "Choosing an ambiguity type is also a bit difficult, especially in this case study, where we had different opinions."

Despite the difficulty of consolidation, two groups consolidated and created a group model. Both of these groups quickly developed a systematic approach, involving analyzing the ambiguity nodes one by one, interactively discussing their representations of that node, and coming to a consensus. Group 1, which did not complete consolidation, came up with their review approach a little later than the other groups and encountered technical difficulties with Google Meet, which is why they ran out of time in Session 3. This process of interactively discussing each node was effective in consolidating the models and, as discussed in the next section, also helped change our participants' perspective on the importance of this process.

5.3 Valuing Ambiguity Analysis - RQ3

Finding 3: Engaging software developers in ambiguity analysis can create buy-in and lead developers to value regulatory activities.

Some participants started the process with doubts about the value of modeling regulatory ambiguities. However, by the end, opinions changed:

> **ID7:** "When I started working [on the model], I thought *"it won't be that important"*. But then I started to realize that this is an important step in the [requirement analysis] process.
> **ID4:** "I first thought that it is a simple task, we don't need a model like this. [It] could be done as we progress. I realized that there is a lot more ambiguities than I realized... it will get complex. So, it will make the process easier if we use this model."

Some participants expressed their thoughts about the modeling process by providing real-world feedback. Others commented on how they might want to use this process to consider other stakeholder perspectives. For example:

> **ID10:** "This is very important, because there are many times in which [I read] our terms and conditions [contracts and] have a different meaning than what the customer means."
>
> **ID3:** "As a developer with the stakeholder, I would love to have a conversation about this legal text, because I wonder how it can be interpreted."

Not all of our participants shared this view. We asked participants the below questions in the survey:

(1) "Did you feel there is value in reviewing regulations and building ambiguity models as part of a Software Development Process?"
(2) "Would you suggest this modeling process as part of the requirement phase to your software development team?"

Four out of five participants responded with a "Definitely Yes," or "Probably Yes" to the two questions. One responded with a "Might or might not" to the first question and a "Probably Not" to the second question.

These survey answers indicate that some participants did not see the value in the ambiguity modeling process. Nevertheless, others did. Some participants realized that ambiguity modeling is about the models produced and more. It is also about perspective and understanding the regulatory requirements during the requirements analysis and documentation. The analysis and documentation are evidence of compliance due diligence within software design.

6 DISCUSSION

In this section, we distill the findings reported in the previous section into two takeaways that have implications for future research in this area, as well as practitioners who are concerned with effective regulatory compliance in their software projects.

6.1 Certain difficulties aid regulatory analysis

Takeaway 1: Lawyers seeking to aid software development teams can benefit from hearing developers articulate their difficulties when reviewing a regulation.

Good regulatory analysis requires that everyone be on the same page and that requires communication and engagement between stakeholders. Knowing some of the difficulties a stakeholder might have in understanding a regulatory text should be part of the conversation. Our modeling process is a tool to aid in that conversation. Most of our participants said they needed help understanding the regulatory text (i.e., Finding 2). Other studies with legal text have made similar points [3, 8, 9, 11]. A lack of regulatory understanding means developers cannot explain or account for regulatory compliance actions in their software design. Lawyers wanting to advise their software teams on applicable regulations should note and discuss these struggles. Our modeling process facilitates and documents the discussion by getting developers to communicate their confusion (i.e., the ambiguities), provide a rationale, and ask meaningful questions about their requirements. Lawyers can respond with clarifying guidance to assist developers with their understanding and make better software design choices.

6.2 Valuing tools and guidance that support regulatory compliance

Takeaway 2: Well-designed processes and tools are vital to aid and document effective regulatory analysis and a culture of compliance for software developers.

Some software developers will view ambiguity modeling as an unnecessary hassle. This type of analysis, though, is necessary for regulatory compliance requirements development and documentation. Having proper tools not only reduces the hassle of regulatory analysis and documentation but also, over time, helps build regulatory analysis into the software design process, thus promoting an organizational culture of compliance. Once a software development team has done an initial assessment, they can communicate and discuss their work with other stakeholders, like lawyers, for more guidance. More importantly, software developers will internalize and see value in implementing such a process within their requirements development (i.e., Finding 3). Lastly, ambiguity models serve as documentation of due diligence, highlighting how developers addressed risks and trade-offs related to complex compliance concerns like privacy and security and complimenting other software engineering artifacts.

7 THREATS TO VALIDITY

All research has validity threats and our study is no exception. This section discusses the threats to our findings' validity.

Internal validity: All our participants built an ambiguity model and explained what they identified as ambiguous. However, within our pilot study, one participant did not complete the model for process and external reasons. Even though we provide resources for any software developer to complete the regulatory ambiguity modeling task, other factors can waylay the process like technical difficulties, competing priorities, and scheduling conflicts. These factors and others might hinder regulatory analysis. Furthermore, we provided incentives for recruitment purposes to conduct the study. Therefore, our participants' motivations to complete the study are different than in the real world.

External Validity: We are limited in generalizing our findings from this study because the participant sample is small. Our study had 11 participants whose ages ranged from 23 to 30 years, had similar cultural backgrounds, and most of the participants' work experience was less than three years[14]. In addition, all the participants identified as software developers, but two had analyst work experience. A repeat of this study would benefit by using a more extensive and diverse selection of participants. Exploring participants and regulations from different domains and jurisdictions may enable other results for comparison.

Reliability: This type of regulatory ambiguity study is novel and does not have a comparison point in the literature. Therefore, we have made available details of our methods and evaluation techniques for others interested in replicating this study[15].

Construct validity: This process allows software developers to communicate issues during requirement analysis to other parties to get answers. We did not test the next step by having our groups present their model to an outside expert. Our ongoing work includes investigations that will explore this next step.

Another construct validity threat is that we conducted the study in a lab environment. Our study used UMBC graduate students, and they worked on tasks unrelated to their jobs or school work. Therefore, our results may have differed if we had used an established software development team operating in the industry. We tried to mitigate this threat by recruiting participants with real-world experience in the software development industry.

[14]Four participants had three or more years of experience

[15]Details are available at: https://doi.org/10.6084/m9.figshare.23297717

8 CONCLUSIONS

We observed software developers interpreting and modeling ambiguities within a regulation. We found that software developers can analyze regulatory ambiguity with a tool and guidance. The analysis will be challenging; however, software developers experiencing and discussing their analysis difficulties is essential to the process. In some ways, the difficulties prove that developers are meaningfully engaging with the regulatory text, and thus actually performing regulatory analysis, and the models are a way to document these due diligence efforts to understand and comply with the law. Overall, engaging developers in these types of activities is vital. It allows them to communicate and document potential issues regarding the understanding of regulatory and compliance requirements. Furthermore, engagement in the regulatory analysis process can change their perspective and create buy-in in the software design analysis and compliance process. Our work has limitations, but our ongoing research addresses some of these by recruiting lawyers, managers, and software developers operating in the software industry to provide feedback on the ambiguity modeling process. The input will assess the utility of our modeling process and improve it to support regulatory compliance in software design.

REFERENCES

[1] Orlando Amaral, Sallam Abualhaija, Mehrdad Sabetzadeh, and Lionel Briand. 2021. A Model-based Conceptualization of Requirements for Compliance Checking of Data Processing against GDPR. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. IEEE, 16–20.

[2] Travis D Breaux and Thomas Norton. 2022. Legal Accountability as Software Quality: A US Data Processing Perspective. In *2022 IEEE 30th International Requirements Engineering Conference (RE)*. IEEE, 101–113.

[3] Travis D Breaux, Matthew W Vail, and Annie I Antón. 2006. Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In *14th IEEE International Requirements Engineering Conference (RE'06)*. IEEE, 49–58.

[4] Juliet Corbin and Anselm Strauss. 2014. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (fourth edition ed.). SAGE Publications, Inc, Los Angeles.

[5] Saad Ezzini, Sallam Abualhaija, Chetan Arora, Mehrdad Sabetzadeh, and Lionel C Briand. 2021. Using domain-specific corpora for improved handling of ambiguity in requirements. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 1485–1497.

[6] Sepideh Ghanavati, Daniel Amyot, and André Rifaut. 2014. Legal goal-oriented requirement language (legal GRL) for modeling regulations. In *Proceedings of the 6th international workshop on modeling in software engineering*. 1–6.

[7] Aaron K Massey, Eric Holtgrefe, and Sepideh Ghanavati. 2017. Modeling regulatory ambiguities for requirements analysis. In *Conceptual Modeling: 36th International Conference, ER 2017, Valencia, Spain, November 6–9, 2017, Proceedings 36*. Springer, 231–238.

[8] Aaron K Massey, Richard L Rutledge, Annie I Antón, Justin D Hemmings, and Peter P Swire. 2015. *A strategy for addressing ambiguity in regulatory requirements*. Technical Report. Georgia Institute of Technology.

[9] Aaron K. Massey, Richard L. Rutledge, Annie I. Antón, and Peter P. Swire. 2014. Identifying and Classifying Ambiguity for Regulatory Requirements. *22nd IEEE International Requirements Engineering Conference (RE),* Karlskrona, Sweden (2014), 83–92.

[10] Sharan B. Merriam and Elizabeth J. Tisdell. 2015. *Qualitative Research: A Guide to Design and Implementation* (4th edition ed.). John Wiley & Sons, San Francisco, CA.

[11] Paul N. Otto and Annie I. Antón. 2007. Addressing Legal Requirements in Requirements Engineering. In *15th IEEE International Requirements Engineering Conference*. 5–14. https://doi.org/10.1109/RE.2007.6