A Pattern-Based Method for Identifying and Analyzing Laws*

[×]Kristian Beckers, [×]Stephan Faßbender,
[*]Jan-Christoph Küster, and [×]Holger Schmidt

[×]paluno - The Ruhr Institute for Software Technology – University of Duisburg-Essen
{firstname.lastname}@paluno.uni-due.de
[*]Australian National University, Canberra, Australia
Jan-Christoph.Kuester@anu.edu.au

Abstract. This paper presents a novel *method for identifying and analyzing laws*. The method makes use of different kinds of *law analysis patterns* that allow legal experts and software and system developers to understand and elicit relevant laws for the given development problem. Our approach also helps to detect dependent laws. We illustrate our method using an online-banking cloud scenario.

1 Introduction

Identifying relevant *compliance* regulations for a software system and aligning it to be *compliant* is a challenging task.The construction of software systems that meet compliance regulations, such as laws, is considered to be difficult, because it is a cross-disciplinary task in laws and software and systems engineering [1]. Otto and Antón [2] conclude in their survey about research on laws in requirements engineering that there is a need for techniques to identify and analyze laws, and to derive requirements from laws.

We present a *pattern-based method for identifying and analyzing laws*. We introduce *law analysis patterns* that allow legal experts and software developers to understand and elicit laws that are relevant for a given development problem.

In this paper, we consider compliance in the field of *cloud computing systems* (or short *clouds*) as an example domain, because using clouds to store and manage critical data and to support sensitive IT processes harbors several problems with respect to compliance. We illustrate our approach using the example of a bank offering an online-banking service for their customers. Customer data such as account number, balance, and transaction history are stored in the cloud, and transactions like credit transfer are processed in the cloud. The bank authorizes the software department to design and build the cloud-specific software according to the interface and platform specification of the cloud provider.For simplicity's sake, we focus in our running example on relevant compliance regulations for privacy. In 1995, the European Union (EU) adopted the *Directive 95/46/EC* on the processing of personal data that represents the minimum

* This research was partially supported by the EU project Network of Excellence on Engineering Secure Future Internet Software Services and Systems (NESSoS, ICT-2009.1.4 Trustworthy ICT, Grant No. 256980).

privacy standards that have to be included in every national law. Germany implements the European Privacy Directive in the *Federal Data Protection Act (BDSG)*.

The rest of the paper is organized as follows: We present patterns to deal with laws in requirements engineering in Sect. 2. Then, in Sect. 3, we discuss related work. In Sect. 4, we give a summary and directions for future research.

2 Pattern-Based Law Analysis

Commonly, laws are not adequately considered during requirements engineering. Therefore, they are not covered in the subsequent system development phases. One fundamental reason for this is that the involved engineers are typically not cross-disciplinary experts in law and software and systems engineering. Hence, we present in this section a pattern-based approach to systematically consider laws in the requirements engineering process. For our method we chose the German law as the binding law. However, we believe that our law identification and analysis method is also valid for laws of other nations.

2.1 Structure of Laws, Sections and Dictates of Justice

The German law is a *statute law* in the tradition of the Roman jurisdiction. Statute laws are specified by the legislator and written down in legal documents. Hence, every judgment of a court is based exclusively on the analysis of the legal documents relevant for the judged case [3, p. 41]. We analyzed, how judges and lawyers are supposed to analyze a law, based upon legal literature research. These insights lead to a basic structure of laws which we used to create law patterns. We describe the results of this analysis in the following.

First of all a law is a textual document. This law document is structured into *sections*. Each section defines a legal aspect of the law and contains several statements. These statements are *dictates of justice*, so-called *legal rules* [4, p. 240]. There are different types of dictates of justice. Complete and self-containing dictates of justice are one type. This type is the fundamental building block of every law [4, p. 241]. We derived the following structure of complete and self-containing dictates of justice. A * next to an element of the structure means the element is optional.

Addressee(s) describe(s) actions that an addressee has to follow or avoid to be compliant

Facts of the case

- **Activity(ies)** describe(s) actions that an addressee has to follow or avoid to be compliant.
- **Target subject(s)*** describes impersonal subjects that are objectives of the activity(ies).Subjects can be material, such as a product, or immaterial, such as information.
- **Target person(s)*** are directly influenced by the activity(ies) of an addressee, or have a relation to the target subject(s).

Legal consequence defines the consequence for an addressee, e.g. the punishment when violating the section.

A dictate of justice is divided into the *facts of the case*, the setting which is regulated, and the *legal consequence*, the resulting implications of the setting [5, p. 7]. Furthermore, a dictate of justice has also an *addressee(s)*. The reason is that every complete dictate of justice is an *imperative*, or can be transformed into an imperative [4, p. 243-44], and an imperative has to be directed towards an addressee(s) [3, p. 3-4].

The facts of the case need to be further refined to be useful for a pattern. The legal method called *subsumption* contains a further refinement of the facts of the case [4, p. 260-64]. This refinement results in the basic elements *activities*, *target subjects*, and *target persons* [5, p. 23-31]. Lawyers use the subsumption to analyze if a dictate of justice is applicable to a specific case. The case is described in terms and notions. Lawyers map these to the notions and terms describing the basic elements [3, p. 52-53]. If not all terms and notions of the case can be mapped to basic elements, the dictate of justice is not relevant for the case.

However, a mapping between all terms and notions of the case and the basic elements is not sufficient to prove the relevance of a dictate of justice for a case. The reason is that the facts of the case of the dictate of justice can contain an element that has no mapping to a term or notion of the specific case. The subsumption solely considers a mapping from the term or notion of the specific case to the dictate of justice. The other direction is not considered. Moreover, such an element has the potential to prove that the law is not relevant for the specific case. The subsumption provides this gap intentionally, because the mapping of specific cases to laws is based upon human interpretation.

Besides the complete, self-containing dictates there are [4, p. 247-251] *definition dictates* that describe and refine terms and other basic elements, *restricting dictates*, which add exceptions to a complete dictate, *directing dictates*, which reference one or more other dictates, and *fiction dictates*, which equate different facts of the case.

These dictates cannot be analyzed in isolation. All of them have relations to other dictates (or even laws). The types of relation between these dictates are *refinement*, *addition*, and *constraint*. This implies that all of resulting dictates and laws, and the relations between them, have to be considered when analyzing laws. A *regulation* is the set of rules applicable to a specific case [4, p. 254].

Thus, relations between laws, sections and dictates of justice are of fundamental importance. They are arranged in a hierarchy, which is not always free of conflicts [4, p. 255]. A special part of these relations is the terminology used within a jurisdiction. This terminology is organized as hierarchical tree where the terms and notions of the more general dictates of justices are refined by subsequent dictates of justice.

2.2 A Process for Identifying relevant Laws

Our general process for identifying relevant laws consists of five steps. For this process law experts and software engineers have to work together for the necessary knowledge transfer. Step one can be done alone by legal experts and for step two only software engineers are needed. But in step three and four both groups are needed to bridge the gap between legal and technical world. The last step can be accomplished alone by legal experts.

Step 1: Law Pattern Based on the previously discussed structure of laws, we define a *law pattern* shown on the upper left-hand side of Fig. 1. The pattern consists of three parts: the dark grey part represents the Law Structure, the light gray part depicts the Classification to consider the specialization of the elements contained in the Law Structure in related laws or sections, and the white part considers the Context. We organize the mentioned hierarchies by Person Classifier, Activity Classifier, and Subject Classifier using *hierarchies*. Figure 2 shows example instances for all three hierarchies according to BDSG. The Context part of the law pattern contains the Legislator(s) defining the jurisdiction, and the Domain(s) clarifying for which domain the law was established.

As it is necessary to know in which context and relation a law is used, we introduce Regulation(s), which are Related To the section at hand. Regulation(s), Legislator(s), and Domain(s) can be ordered in hierarchies, similar to classifiers. For instance, Germany is part of the EU and consists of several states.

We now describe the instantiation process for our law pattern using Section 4b BDSG as an example. We explained the importance of this particular law in Sect. 1. Section 4b BDSG regulates the abroad transfer of data. The resulting instance is shown on the right-hand side of Fig. 1. Our process starts based on the first sections of the law to be analyzed. These sections are self-contained, i.e. they define all necessary elements of our Law Structure. Additionally, the Legislator(s) and Domain(s) can be instantiated according to the considered law (e.g. Germany and General Public in the Context part). Given a section of a law not yet captured by our law pattern, we identify and document the related laws and sections referred to by the given section (e.g. BDSG Sec. 1 in the Context part). Then, we search for the Law Structure directly defined in this section. In Section 4b BDSG, we find Abroad Transfer, and we use it to instantiate Activity(ies). Addressee(s), Target Subject(s), and Target Person(s) are not defined in Section 4b BDSG.

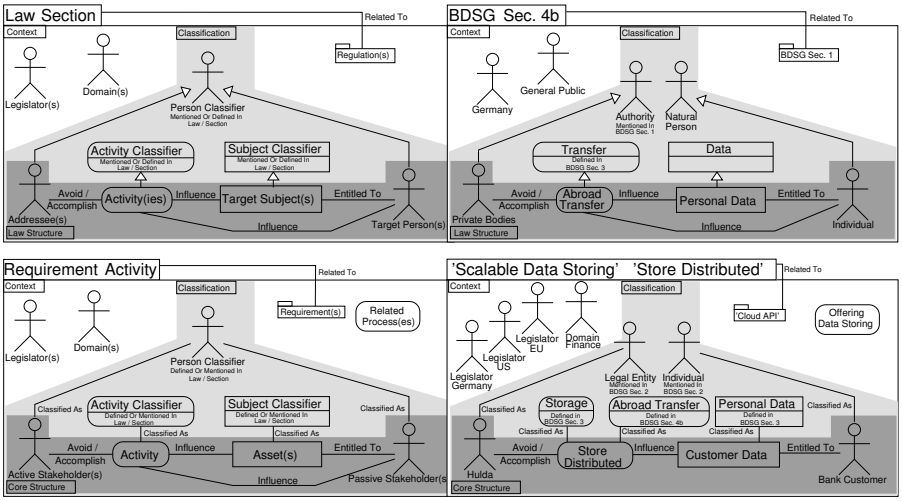

Fig. 1. Law Pattern (upper left) and Instance (upper right), Law Identification Pattern (lower left) and Instance (lower right)

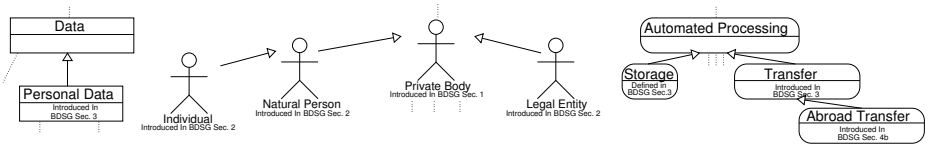

Fig. 2. Examples for Person (left), Subject (middle), and Activity (right) Hierarchies

Therefore, related sections defining these terms have to be discovered. In our example, we find Private Bodies for the Addressee(s), Personal Data for the Target Subject(s), and Individual for the Target Person(s) in Section 1 BDSG (according to BDSG Sec. 1 in the Context part). We arrange these specializations in the appropriate parts of the hierarchies in Fig. 2. The classifier is instantiated with the parent node of the corresponding hierarchy, which is for instance Transfer, defined in Section 3 BDSG, for Abroad Transfer.

Step 2: Law Identification Pattern Identifying relevant laws based on functional requirements is difficult, because functional requirements are usually too imprecise, they contain important information only implicitly and use a different wording than in laws. To bridge between gap of the wording and to facilitate the discussion between requirements engineers and legal experts, we define a *law identification pattern* to support identifying relevant laws

Figure 1 shows on the lower left-hand side our law identification pattern. The structure is similar to the law pattern on the upper left side of Fig. 1 to allow a matching of instances of both patterns. In contrast to the legal vocabulary used in the Law Structure of our law pattern, the wording for the elements in the dark gray colored Core Structure of our law identification pattern is based on terms known from requirements engineering. For example, the element Asset(s) in our law identification pattern represents the element Target Subject(s) in our law pattern.

Our law identification pattern takes into account that requirements are often interdependent (Requirement(s) in the Context part). Given a law relevant to a requirement, the same law might be relevant to the dependent requirements, too. Furthermore, the pattern helps to document similar dependencies for a given Activity using the Related Process(es) in the Context part.

As our example on the lower right-hand side of Fig. 1 shows, we select Hulda as the cloud provider, then we choose the functional requirement Scalable Data Storing. One of the activities associated with this requirement is the activity Store Distributed , which refers to the asset Customer Data of the Bank Customer. Moreover, we instantiate the elements Legislator(s) and Domain(s). In our example on the lower right side of Fig. 1, we include the legislators Germany, US, EU, and the domain Finance. In addition, we discover the related requirement Cloud API and the process Offering Data Storing, and document them in the instance of our law identification pattern.

Step 3: Establishing the Relation between Laws and Requirements To instantiate the Classification part, legal expertise is necessary. According to the Core Structure of the instance of our law identification pattern and the hierarchies built when instantiating our law pattern, legal experts classify the elements of the Core Structure. For example, the

activity Store Distributed is classified as Abroad Transfer based on a discussion between the legal experts and software engineers.

Step 4: Deriving relevant Laws The identification of relevant laws is based on matching the classification part of the law identification pattern instance (light gray part) with the law structure and classification part of the law pattern instance (light and dark gray parts), and thereby considering the previously documented hierarchies. If all elements match, the law is identified as relevant. For example, we find direct matches in the law pattern instance depicted on right side of Fig. 1 for the elements Abroad Transfer, Personal Data, and Individual contained in the law identification pattern instance shown on the lower right side of Fig. 1. Hulda is classified as Legal Entity and the only element that does not directly match with Private Bodies in the law structure of Section 4b BDSG. In this case, the hierarchy in Fig. 2 helps to identify that Legal Entity is a specialization of Private Bodies, and thus, we identify Section 4b BDSG as relevant.

Finally, we check for all laws identified to be relevant if Legislator(s) and Domain(s) are mutually exclusive. In our example, the legislator Germany contained in Context of the law pattern instance depicted on lower right side of Fig. 1 can be found in Context of the law identification pattern instance shown on the lower right side of Fig. 1. The domain General Public in the law pattern instance can be considered as a generalization of the domain Finance in the law identification pattern instance. The resulting set of laws relevant for the given development problem serves as an input for the next step.

Step 5: This last step covers the identification and specification of requirements based on laws identified to be relevant by our approach, e.g. using existing approaches such as the one from Breaux et al. [6].

3 Related Work

Breaux et al. [6] present a framework that covers analyzing the structure of laws using a natural language pattern. This pattern helps to translate laws into a more structured The approach has some drawbacks of formal logic analysis of laws we will discuss later in this section.

Siena et al. [7] describe the differences between legal concepts and requirements. The resulting process to align legal concepts to requirements and the given concepts are quite high level and cannot directly be applied to a scenario. In contrast to our approach they do not identify relevant laws and do not intertwine compliance regulations with already elicited requirements.

Álvarez et al. [8] describe reusable legal requirements in natural language. We believe that the work by Álvarez et al. complements our work, i.e., applying our law identification method can precede using their security requirements templates.

Bench-Capon et al. [9] make use of first order logic to model regulations. In contrast to our work, the authors assume that the relevant laws are already known.

4 Conclusions

We presented a *pattern-based method for identifying and analyzing laws*, which can be embedded in common system and software development processes. The novelty about

our approach is that we analyzed common methods lawyers use to identify and analyze laws. We captured this knowledge in patterns. We derived this pattern-based approach from the subsumption method, while other approaches use formal logic to formalize and analyze laws. Logic-based approaches seem to be more precise. However, legislators formulate laws imprecise by design [4, 3, 5, p. 298-99, p. 36-39, p. 32-33]. Hence, we decided to capture the modus operandi from lawyers in a pattern-based method. Biagioli et al. investigated Italian law and derived also a structure of dictates of justice, which is very similar to the structure presented in this work. [1, p. 247]. Thus, it is likely that the pattern is also applicable to further laws in the tradition of the Roman jurisdiction. The case law system, in the US or Great Britain, is another important legal system. We plan to adapt our method for the case law system, via case patterns that extend law patterns. We also aim to work on tool support for our approach, e.g. to store, load, and search for laws once they have been fitted to our law patterns. The tool support will be used for validation of our method. We are planning to use our approach on the entire BDSG, which has 48 sections. We estimate around 4 patterns are required per section on average, which will lead to a result of around 200 patterns for the entire law.

Acknowledgements

We thank Maritta Heisel and Christoph Sorge for their extensive and valuable feedback on our work.

References

1. Biagioli, C., Mariani, P., Tiscornia, D.: Esplex: A rule and conceptual model for representing statutes. In: Proceedings of the 1st international conference on Artificial intelligence and law. ICAIL '87, ACM (1987) 240–251
2. Otto, P.N., Antón, A.I.: Addressing legal requirements in requirements engineering. In: Proceedings of the International Conference on Requirements Engineering (RE), IEEE Computer Society (2007) 5–14
3. Schwacke, P.: Juristische Methodik mit Technik der Fallbearbeitung. 4. edn. Kohlhammer Deutscher Gemeindeverlag (2003)
4. Larenz, K.: Methodenlehre der Rechtswissenschaft. 5. edn. Springer (1983)
5. Beaucamp, G., Treder, L.: Methoden und Techniken der Rechtsanwendung. 2. edn. C.F.Müller (2011)
6. Breaux, T.D., Antón, A.I.: Analyzing regulatory rules for privacy and security requirements. IEEE Transactions on Software Engineering **34**(1) (2008) 5–20
7. Siena, A., Perini, A., Susi, A.: From laws to requirements. In: Proceedings of the International Workshop on Requirements Engineering and Law (RELAW), IEEE Computer Society (2008) 6–10
8. Álvarez, J.A.T., Olmos, A., Piattini, M.: Legal requirements reuse: A critical success factor for requirements quality and personal data protection. In: Proceedings of the International Conference on Requirements Engineering (RE), IEEE Computer Society (2002) 95–103
9. Bench-Capon, T., Robinson, G., Routen, T., Sergot, M.: Logic programming for large scale applications in law: A formalization of supplementary benefit legislation. In: Proceedings of the International Conference on Artificial Intelligence and Law (ICAIL), ACM (1987) 190–198