# A Pattern-Based Method for Identifying and Analyzing Laws in the Field of Cloud Computing Compliance*

[×]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
[*]Fraunhofer Institut for Software and Systems Engineering ISST
Jan-Christoph.Kuester@isst.fraunhofer.de

**Abstract.** Cloud computing offers highly flexible and scalable usage of IT resources, from which companies can benefit. Clouds are socio-technical systems with a high number of different kinds of stakeholders. Moreover, they are often geographically distributed, process critical data, and support sensitive IT processes. Therefore, aligning clouds to meet compliance regulations is a challenging task. Presently, this unsolved problem prevents companies from using clouds for critical tasks.
This paper presents a novel *method for identifying and analyzing laws for clouds*. The method makes use of different kinds of *patterns*, which help to systematically elicit relevant laws. We present *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. Our law analysis patterns make use of results generated by the application of a *cloud system analysis pattern* and different kinds of *stakeholder templates* that serve to understand and describe a given cloud development problem.
We illustrate our method using an online banking cloud scenario.

**Keywords:** law, compliance, cloud computing, requirements engineering

## 1 Introduction

Identifying relevant *compliance* regulations for a software system and aligning it to be *compliant* is a challenging task. The notion of compliance summarizes a large set of *regulations* such as national as well as international laws and domain-specific regulations. Typically, compliance regulations lead to non-functional requirements such as security requirements and also functional requirements. Several requirements engineering approaches for, e.g. security requirements exist [1]. However, 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. In addition, the construction of

* 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).

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 [3, 4].

In this paper, we consider compliance in the field of *cloud computing systems* (or short *clouds*), because using clouds to store and manage critical data and to support sensitive IT processes harbors several problems with respect to compliance. A PriceWaterhouseCoopers study from 2010 reveals that identifying compliance requirements is a significant challenge for compliance management in clouds.[1]

We present a *pattern-based method for identifying and analyzing laws for clouds*. We introduce *law analysis patterns* that allow legal experts and software and system developers to understand and elicit laws that are relevant for a given development problem. According to our example domain clouds, we make use of a *pattern for analyzing clouds*, which is complemented by templates to elicit knowledge about the different stakeholders contained in the pattern [5]. The pattern and especially the stakeholder templates are the basis for functional requirements engineering and the identification of activities, which serve as input for our law identification and analysis method.

Our approach can be applied during requirements engineering when constructing clouds, when moving existing IT landscapes to clouds, and it is also applicable to evaluate and assess existing clouds with respect to compliance.

We illustrate our approach using the example of a bank offering an online-banking service for their customers. This bank plans to source out the affected IT processes to reduce costs and scale up their system for a larger amount of 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.

The rest of the paper is organized as follows: Section 2 presents background on clouds, a short survey on compliance, and our pattern-based analysis approach for clouds. In Sect. 3, we outline how our approach can be used together with typical requirements engineering techniques. Then, we present patterns to deal with laws in requirements engineering in Sect. 4. We consider insights in Sect. 5 and related work in Sect. 6. In Sect. 7, we give a summary and directions for future research.

## 2 Background

In Sect. 2.1, we describe the main characteristics of clouds. In Sect. 2.2, we present a short survey about the main compliance problems. We focus on privacy concerns of data storage and processing in the cloud. We present our pattern-

[1] `http://www.pwc.de/en/prozessoptimierung/trotz-einiger-bedenken-der-virtuellen-datenverarbeitung-gehoert-die-zukunft.jhtml`

based approach for analyzing cloud computing systems in Sect. 2.3.

### 2.1 Cloud Computing Systems

According to the *National Institute of Standards and Technology (NIST)* cloud computing systems can be defined by the following properties [6]: the cloud customer can acquire resources of the cloud provider over *broad network access* and *on-demand* and pays only for the used capabilities. Resources, i.e., storage, processing, memory, network bandwidth, and virtual machines, are combined into a so-called *pool*. Thus, the resources can be virtually and dynamically assigned and reassigned to adjust the customers' variable load and to optimize the resource utilization for the provider [7]. The virtualization causes a location independence: the customers generally have no control or knowledge over the exact location of the provided resources. The resources can be quickly scaled up and scaled down for customers and appear to be unlimited, which is called *rapid elasticity*. The pay-per-use model includes guarantees such as availability or security for resources via customized *Service Level Agreements (SLA)* [7].

The architecture of a cloud computing system consists of different service layers and allows different business models: the *Infrastructure as a Service (IaaS)* layer, which is closest to the physical resources, provides pure resources, for instance virtual machines, where customers can deploy arbitrary software including an operating system. Data storage interfaces provide the ability to access distributed databases on remote locations in the cloud.

On the *Platform as a Service (PaaS)* layer, customers use an API to deploy their own applications using programming languages and tools supported by the provider. On the *Software as a Service (SaaS)* layer, customers use applications offered by the cloud provider that are running on the cloud infrastructure.

### 2.2 Compliance

Compliance management is a "broad term covering all activities and methods to ensure that a company follows all policies required by an external or internal regulation"[8]. Those regulations exist in form of laws, norms, standards, and best practices to control and support companies managing their risks in a responsible way.

Clouds raise a number of compliance issues. For reasons of space, we only list a few. First, cloud providers often are not able to provide detailed information on the location of their customers' data [9]. This is relevant e.g. to obey privacy laws. Second, an open question is how a cloud provider can prove that data has been deleted [10]. Third, cloud providers and customers are often located in different countries. In this case, the laws of the cloud provider's country are relevant. However, cloud providers and customers can agree on using the laws of one country for their cloud business. Furthermore, contracts have to fill the gap between the agreed law and the law of the other countries of the stakeholders [11]. Fourth, contracts are also used to define the ramifications of violations of the clouds's SLAs. Fifth, the previous issues multiply in complexity, when the cloud provider can use subcontractors, e.g., from another country. Moreover, it is hardly possible for cloud customers to recognize that their data has been

processed by a third party [10]. Sixth, the use of distributed computing environments, spread all over the globe, provides a challenge for auditing demands [10].

In our running example 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. In order to give an idea of the number of laws, regulations, and standards, that would have to be considered, we present the following list, which could be extended even further:

> *Law on Monitoring and Transparency in Businesses (KonTraG)*, *Stock Corporation Act (AktG)*, *German banking act (KWG)*, *Securities Trading Act (WpHG)*, *Minimum Requirements for Risk Management (MaRisk)*, *Commercial Code (HGB)*, *Tax Code (AO)*, *State Data Protection Acts (LDSG)*, *Telemedia Act (TMG)*, *Federal Data Protection Act (BDSG)*.

From this brief survey alone one can recognize, that even for our small running example, a huge number of laws might become relevant. This fact emphasizes the need for an engineering method for the identification of relevant laws and their analysis, which is based on re-usability in terms of patterns.

For simplicitie's sake, we focus in our running example on relevant compliance regulations for privacy. A good working definition for privacy according to Pfleeger et al. [12, p. 607] is that "privacy is the right to control who knows certain aspects about you, your communication and your activities". In 1995, the European Union (EU) adopted the *Directive 95/46/EC* on the processing of personal data [13] that represents the minimum privacy standards that have to be included in every national law. Germany implements the European Privacy Directive in the *Federal Data Protection Act (BDSG)*. According to the appendix of *Section 9 Sentence 1 BDSG* all organizations and companies that automatically process, store, and use personal data have to comply with the BDSG. IT systems have increased the feasibility of unwanted disclosure, because storage capacity and speed of computers allow to store, search and correlate data. *Section 9 Sentence 1 BDSG* states different requirements that have to be fulfilled by technical and organizational measurements for protecting personal data, e.g., physical and virtual access control to data and the separation of storing and processing data collected for different purposes. Furthermore, it must be verifiable whether personal data has been deleted and by whom and that data has only been processed with the permission of the customer.

Moreover, the EU law as well as *Section 4b Sentence 2 BDSG* forbid sharing data with companies or governments in countries that have weaker privacy laws. For exchange with companies in the United States (US), there exists the *Safe Harbour* agreement. But under the *US Patriot Act*, officials could access information about citizens of other countries, if that information is physically located within the US or accessible electronically. The priority of the Patriot Act has never been explicitly tested in court, but is a risk for bringing privacy-critical data into the cloud where data centers can be technically distributed world-wide. As cloud computing is considered as contracted data processing, the cloud customer is responsible to adhere to the complete BDSG, according to *Section 11 BDSG*. The law further defines the contract between customer and outsourcing provider. For example after ending the contract all data has to be deleted.

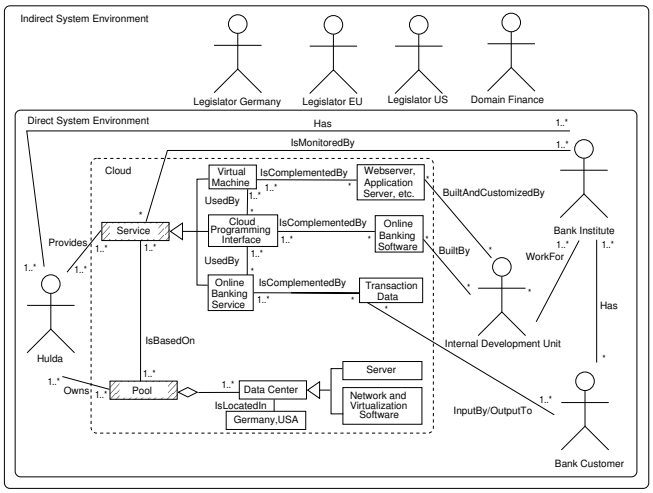


**Fig. 1.** Concrete Cloud Computing System for Online Banking Service

### 2.3 Pattern-Based Analysis of Clouds

We present a *cloud system analysis pattern* in [5] that provides a conceptual view on cloud computing systems and serves to systematically analyze stakeholders and requirements. We do not introduce the pattern in detail here; instead we present an instance of our cloud system analysis pattern in Fig. 1 regarding the online banking service example.

Basically, the online banking cloud service is embedded in an environment consisting of two parts, namely the Direct System Environment and the Indirect System Environment. The Direct System Environment contains stakeholders and other systems that directly interact with the cloud through associations, e.g. the Bank Customer. Moreover, associations between stakeholders in the Direct and Indirect System Environment exist, but not between stakeholders in the Indirect System Environment and the cloud. For example, the Legislator Germany is part of the Indirect System Environment. Typically, the Indirect System Environment is a significant source for compliance and privacy requirements.

We supplement the cloud system analysis pattern by templates to systematically gather domain knowledge about the direct and indirect system environments based upon the stakeholders' relations to the cloud and other stakeholders. When instantiating the cloud system analysis pattern, one also fills in the corresponding templates. These templates contain for example the following information: a description of a stakeholder, a stakeholder's motivation for using the cloud, relations to the cloud and to other stakeholders, and the assets of a stakeholder. Detailed information about stakeholder templates are in [5].

## 3 A Requirement Elicitation Method for Clouds

We describe how the concepts presented in the previous section can be integrated with typical requirements engineering techniques via a life cycle of our

requirement elicitation method for clouds. The starting point of our method is an informal problem description and the cloud system analysis pattern including the stakeholder templates as presented in Sect. 2.3.

*Instantiate Cloud System Analysis Pattern and Stakeholder Templates* The goal of this first step is to collect and structure knowledge about the envisaged cloud, especially about the involved stakeholders. We describe the instantiation procedure in [5] and we presented the results for our online banking example in Sect. 2.3. The output of this step consists of an instance of the cloud system analysis pattern and instances of stakeholder templates for each involved stakeholder. For steps 2–4, we use existing approaches. The instances of the cloud system analysis pattern and the stakeholder templates can be considered similar to, e.g. a *context diagram* in the problem-oriented requirements engineering approach of Jackson [14]. These instances are the basis to derive *functional requirements* in the second step of our method. Here, given approaches such as the problem decomposition approach by Jackson or an approach based on use cases can be applied. The third step covers the discovery of activities based on functional requirements, e.g. using UML[2] behaviour diagrams such as activity or sequence diagrams. This represents best practice in many object-oriented development methods such as the one by Cheesman and Daniels [15]. In the fourth step, assets are identified based on functional requirements and corresponding activities. Again, we reuse given approaches here, e.g. the one by Fernandez et al. [16], which makes use of UML activity diagrams to identify assets and threats, or the *misuse case* approach by Sindre and Opdahl [17]. Using the results of the artifacts generated in steps 2–4, we extend the direct stakeholder template instances initially developed in the first step by new rows "Functional requirements", "Activities", and "Assets". We illustrate the last two steps of our method in the next section as a part of our novel pattern-based law analysis approach.

## 4 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.

### 4.1 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 [18, 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 and the contained sections, which

[2] Unified Modelling Language: `http://www.omg.org/spec/UML/2.3/Superstructure/PDF/`

**Table 1.** Structure of law rules

| | | |
|---|---|---|
| **Addressee(s)** | has (have) to comply to the law. | |
| **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 * next to an element of the structure means the element is optional.

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* [19, 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 [19, p. 241].

We derived the structure of complete and self-containing dictates of justice and present the results in Tab. 1. 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 [20, 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 [19, p. 243-44], and an imperative has to be directed towards an addressee(s) [18, 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 [19, p. 260-64]. This refinement results in the basic elements *activities*, *target subjects*, and *target persons* [20, 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 [18, 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 [19, 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. The referenced dictates contain (parts of) the facts of the case or the legal consequences.
- *fiction dictates*, which equate different facts of the case. But this equation can be proved wrong within a law 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 [19, 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 [19, 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.

### 4.2 A Process for Identifying relevant Laws

Our general process for identifying relevant laws consists of five steps as depicted in Fig. 2. The first step is to set up a database of all laws which might be of relevance for a scenario. Therefore, laws have to be analyzed and stored in the structure of the law pattern. Thus, they are stored as pattern instances. Section 4.3 describes how to accomplish this step. This step is not needed if such a database already exists. The second step, described in Sect. 4.4, uses information from software requirements and their context to instantiate the core structure and the context of the law identification pattern. Instances of the cloud analysis pattern contain parts of the relevant information for this instantiation. Third, the relation between laws and software requirements has to be established (Sect. 4.5) to prepare the identification of relevant laws for the given software. Hence, a mapping between the terms and notions of the software requirements to legal terms and notions is derived. Fourth, the law pattern instances and law identification pattern instances have to be matched. This results in a set of laws

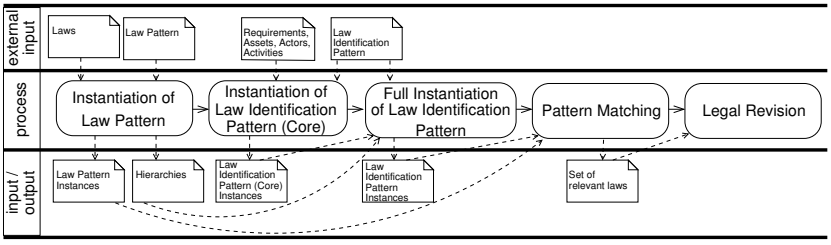


**Fig. 2.** Law Identification Process

which might be of relevance for the software. The resulting laws are only possibly relevant, because we use the subsumption method, as discussed in Sect. 4.1. The matching is described in Sect. 4.6. Fifth, the found laws are the basis for further investigations. 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.

### 4.3 Law Pattern

Based on the previously discussed structure of laws, we define a *law pattern* shown on the left-hand side of Fig. 3. 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 4 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 section in Sect. 2.2. The resulting instance is shown on the right-hand side of Fig. 3. 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

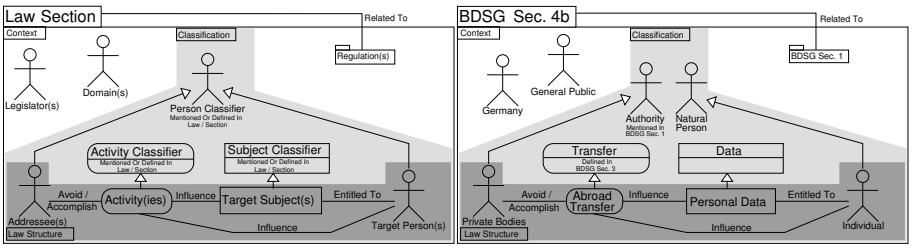


**Fig. 3.** Law Pattern (left) and Instance (right)

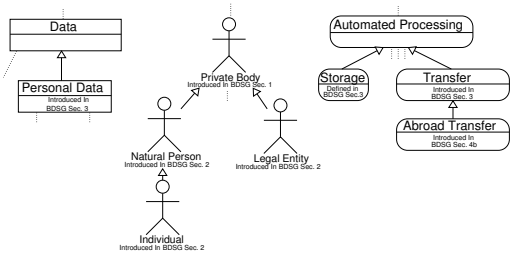


**Fig. 4.** Example Instances for Person (left), Subject (middle), and Activity (right) Hierarchies

it to instantiate Activity(ies). Addressee(s), Target Subject(s), and Target Person(s) are not defined in Section 4b BDSG. 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. 4. 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.

### 4.4 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. For example, a functional requirement like "The customer wants to withdraw money." might lead to different laws when searching for "customer" only, and most of them might not be relevant in this case (e.g. laws dealing with retail markets). Additionally, laws dealing with banks or Internet communication are not taken in consideration since the functional requirement does not contain adequate keywords. Moreover, it is difficult to discover that the amount of withdrawn money implicitly contains individual-related data, which has to be protected for privacy reasons. Formulating the requirement in a more comprehensive way, e.g. "The bank customer wants to withdraw money from his account using a web interface, which is offered via Internet by the bank using a cloud provider.", does not solve these problems.

To bridge the 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 based on the early steps of our method presented in Sect. 3. We especially use the laws captured with the law pattern presented in the previous section, and the knowledge collected using the stakeholder templates as mentioned in Sect. 2.3.

Figure 5 shows on the left-hand side our law identification pattern. The structure is similar to the law pattern on the left side of Fig. 3 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)

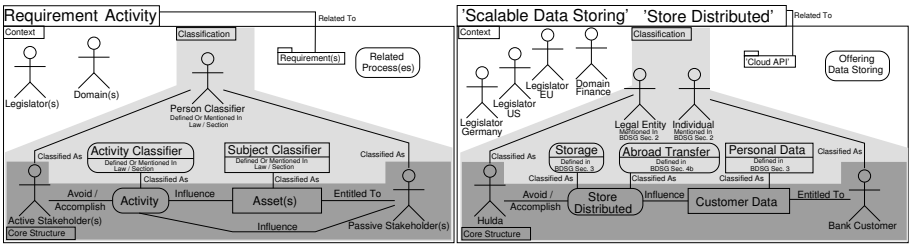


**Fig. 5.** Law Identification Pattern (left) and Instance (right)

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.

For identifying laws relevant for the online banking service example, we consider the activities documented in the direct stakeholder template instances of the instantiated cloud system analysis pattern as described in Sect. 3. As our example on the right-hand side of Fig. 5 shows, we select the template instance of the direct stakeholder Hulda, then we choose the functional requirement Scalable Data Storing (row "Functional Requirements" in the Hulda stakeholder template instance). One of the activities associated with this requirement is the activity Store Distributed (row "Activities" in the Hulda stakeholder template instance), which refers to the asset Customer Data (row "Assets" in the Hulda stakeholder template instance) of the Bank Customer. Moreover, we instantiate the elements Legislator(s) and Domain(s) according to the instantiated cloud system analysis pattern. In our example on the right side of Fig. 5, 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. So far, the instantiation process can be performed by a software engineer.

### 4.5 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.

### 4.6 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. 3 for the elements Abroad Transfer, Personal Data, and Individual contained in the law identification pattern instance shown on the right side of Fig. 5. 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. 4 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 right side of Fig. 3 can be found in Context of the law identification pattern instance shown on the right side of Fig. 5. 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 step 6 of our requirements elicitation method for clouds presented in Sect. 3. 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. [21, 22].

## 5 Insights

We recognized a common structure that covers German laws, presented in Sect. 4. 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. "We have tried to respect the distinction between legal actions[...], the active parties of the action[...], and the passive parties or objects of the action" [3, p. 247]. Thus, our law pattern is applicable for the Italian law as well. It is likely that the pattern is also applicable to further laws in the tradition of the Roman jurisdiction.

Our method requires adaptation for fundamentally different legal systems from the Roman legal system, e.g. the case law system in the US or Great Britain. The case law demands from legal experts exhaustive knowledge about numerous court rulings, which augment statute laws. Judges are able to dismiss a law if it is not a perfect match for a case. They can rule based upon previous rulings from judges on similar cases. Judges decide what cases to consider. However, Fikentscher states that also in countries with case law the basis for jurisdiction are the statute laws. He concludes, further, that statute law is part of the case law [23, p. 111 - 113]. Hence, our method covers this part of the case law.

We presented that several basic elements from laws have a relation to basic elements from our cloud analysis pattern. Hence, the information gathered in our instantiated cloud analysis pattern can be re-used for the instantiation of our law pattern (Sect. 4.4).

We derived a pattern-based approach from the subsumption method, while other approaches use formal logic to formalize and analyze laws. The subsump-

tion method is a common approach for analyzing laws among lawyers (Sect. 4.1). Logic-based approaches seem to be more precise. However, legislators formulate laws imprecise by design [19, 18, 20, p. 298-99, p. 36-39, p. 32-33]. This imprecision allows lawyers and judges to interpret laws. This design is not sustainable when using formal logic. Hence, we decided to capture the modus operandi from lawyers in a pattern-based method.

## 6 Related Work

Breaux et al. [21, 22] 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 restricted natural language and then into a first order logic. The idea of using first order logic in the context of regulations is not a new one. For example Bench-Capon et al. [24] made use of first order logic to model regulations and related matters. In contrast to our work, the authors of those approaches assume that the relevant laws are already known and thus do not support identifying legal texts. Their approach does not allow one to find dependent law sections. The approach also has the drawbacks of formal logic analysis of laws as discussed in Sect. 5.

Siena et al. describe in one publication [25] the differences between legal concepts and requirements. They model the regulations using an ontology, which is quite similar to the natural language patterns described in the approaches mentioned before. The ontology is based in the Hohfeld taxonomy [26], which describes the means and relations between the different means of legal texts in a very generic way. Thus Hohfeld does not structure a certain law at all but aims at the different meanings of laws. So the resulting process in [25] to align legal concepts to requirements and the given concepts are quite high level and cannot directly applied to a scenario. In a second work Siena et al. [27] try to bridge the gap between the requirements engineering process and compliance using a goal-oriented approach. In contrast to our approach they do not identify relevant laws and do not intertwine compliance regulations with already elicited requirements.

Álvarez et al. [28] describe reusable legal requirements in natural language, and based on the Spanish adaption of the EU directive 95/46/CE concerning personal data protection. We believe that the work by Álvarez et al. complements our work, i.e., applying our law identification method can preceed using their security requirements templates.

## 7 Conclusions and Future Work

We presented a *pattern-based method for identifying and analyzing laws for clouds*, 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. Our method comprises the following main benefits:

– Systematic pattern-based identification and analysis of laws

- Detection of dependent laws
- Considering legal requirements in systems and software engineering
- Bringing together legal experts and software and system developers
- Re-using cloud-specific context and stakeholder analysis based on patterns

The diverse structure of legal systems and laws in different countries around the world present a challenge for our approach. We are confident that the approach is feasible for common use. 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.

**Acknowledgements**

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

## References

1. Fabian, B., Gürses, S., Heisel, M., Santen, T., Schmidt, H.: A comparison of security requirements engineering methods. Requirements Engineering – Special Issue on Security Requirements Engineering **15**(1) (2010) 7–40
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. 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
4. Bobkowska, A., Kowalska, M.: On efficient collaboration between lawyers and software engineers when transforming legal regulations to law-related requirements. In: Information Technology (ICIT), 2010 2nd International Conference on. (june 2010) 105 –109
5. Beckers, K., Küster, J.C., Faßbender, S., Schmidt, H.: Pattern-based support for context establishment and asset identification of the ISO 27000 in the field of cloud computing. In: Proceedings of the International Conference on Availability, Reliability and Security (ARES), IEEE Computer Society (2011) 327–333
6. Mell, P., Grance, T.: The NIST definition of cloud computing. Working Paper of the National Institute of Standards and Technology (NIST) (2009)
7. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. SIGCOMM Computer Communication Review **39**(1) (2009) 50–55
8. Marwane, E.K., Stein, S.: Policy-based semantic compliance checking for business process management. In: Proceedings of the Workshops co-located with the Conference on Modellierung betrieblicher Informationssysteme (MobIS). Volume 420 of CEUR Workshop Proceedings., CEUR-WS.org (2008) 178–192
9. Jansen, W.A.: Cloud hooks: Security and privacy issues in cloud computing. In: Proceedings of the 2011 44th Hawaii International Conference on System Sciences. HICSS '11, IEEE Computer Society (2011) 1–10

10. Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., Molina, J.: Controlling data in the cloud: outsourcing computation without outsourcing control. In: Proceedings of the 2009 ACM workshop on Cloud computing security. CCSW '09, ACM (2009) 85–90
11. Duisberg, A.: Gelöste und ungelöste rechtsfragen im it-outsourcing und cloud computing. In Picot, A., Götz, T., Hertz, U., eds.: Trust in IT. Springer Berlin Heidelberg (2011) 49–70
12. Pfleeger, C.P., Pfleeger, S.L.: Security In Computing. 4th edn. Prentice Hall PTR (2007)
13. Hansen, M., Schwartz, A., Cooper, A.: Privacy and identity management. IEEE Security and Privacy **6**(2) (2008) 38–45
14. Jackson, M.: Problem Frames. Analyzing and structuring software development problems. Addison-Wesley (2001)
15. Cheesman, J., Daniels, J.: UML Components – A Simple Process for Specifying Component-Based Software. Addison-Wesley (2001)
16. Fernandez, E.B., la Red M., D.L., Forneron, J., Uribe, V.E., Rodriguez G., G.: A secure analysis pattern for handling legal cases. In: Latin America Conference on Pattern Languages of Programming (SugarLoafPLoP). (2007) `http://sugarloafplop.dsc.upe.br/wwD.zip`.
17. Sindre, G., Opdahl, A.L.: Capturing security requirements through misuse cases. In: Proceedings of the Norwegian Informatics Conference (NIK). (2001)
18. Schwacke, P.: Juristische Methodik mit Technik der Fallbearbeitung. 4. edn. Kohlhammer Deutscher Gemeindeverlag (2003)
19. Larenz, K.: Methodenlehre der Rechtswissenschaft. 5. edn. Springer (1983)
20. Beaucamp, G., Treder, L.: Methoden und Techniken der Rechtsanwendung. 2. edn. C.F.Müller (2011)
21. Breaux, T.D., Vail, M.W., Antón, A.I.: Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In: Proceedings of the International Conference on Requirements Engineering (RE), IEEE Computer Society (2006) 46–55
22. Breaux, T.D., Antón, A.I.: Analyzing regulatory rules for privacy and security requirements. IEEE Transactions on Software Engineering **34**(1) (2008) 5–20
23. Fikentscher, W.: Methoden des Rechts in vergleichender Darstellung. Band 2: Anglo-amerikanischer Rechtskreis. 1. edn. Volume 2. Mohr Siebeck (1975)
24. 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
25. 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
26. Hohfeld, W.N.: Fundamental legal conceptions as applied in judicial reasoning. The Yale Law Journal **26**(8) (1917) 710–770
27. Siena, A., Perini, A., Susi, A., Mylopoulos, J.: A meta-model for modelling law-compliant requirements. In: Proceedings of the International Workshop on Requirements Engineering and Law (RELAW), IEEE Computer Society (2009) 45–51
28. 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