Process and Tool-support to Collaboratively Formalize Statutory Texts by Executable Models

Bernhard Waltl, Thomas Reschenhofer, and Florian Matthes

Software Engineering for Business Information Systems
Department of Informatics
Technische Universität München
Boltzmannstr. 3, 85748 Garching bei München, Germany
b.waltl@tum.de, reschenh@in.tum.de, matthes@in.tum.de

Abstract. This paper describes a collaborative modeling environment to support the analysis and interpretation of statutory texts, i.e., laws. The paper performs a case study on formalizing the product liability act and proposes a data-centric process that enables the formalization of laws. The implemented application integrates state-of-the-art text mining technologies to assist legal experts during the formalization of norms by automatically detecting key concepts within legal texts, e.g., legal definitions, obligations, etc. The work at hand elaborates on the implementation of data science environment and describes key requirements, a reference architecture and a collaborative user interface.

Keywords: Formalizing Norms; Semantic Modeling; Computable Law; Data Analytics; Text Mining; Apache UIMA; Reference Architecture

1 Introduction

Although, several legislations have the ambitious goal that laws should be understandable by everyone (see [1]), understanding and interpreting legal norms is a difficult and complex task. Civil law systems have developed complex interpretation canons, that can be used during the interpretation of legal norms to avoid misinterpretations and to unveil and determine the intrinsic uncertainty and vagueness [2, 3].

Since the interpretation of laws is a repeating, data-, time-, and knowledge-intensive task, the lack of appropriate tool-support is counter-intuitive in several ways. On the one hand, providing tool-support to model and store the result of the interpretation process is a measure to unveil intrinsic vagueness, inconsistencies, complex regulations. On the other hand, the development of an application to collaboratively formalize and refine a semantic model of normative texts becomes—due to the advances in enterprise information systems engineering and because of text mining algorithms—more and more attractive [4, 5]. Up to now, there is a gap between the technological possibilities and the current support of legal interpretation processes, which holds especially for the legal domain in Germany. In 2015 the dutch researcher van Engers explicitly stated: “While many attempts to automate law [. . .] have been made before, hardly any attention has been paid to the 'translation process' from legal rules expressed in natural language to specifications in computer executable form." [6]

It is unlikely, that the interpretation process can be fully automated through algorithms. Instead, the advances in natural language processing (NLP) and detection of patterns in legal texts are considered to be supportive to legal (data) scientists and practitioners. The paper's contribution can be stated as follows:
(i) How does a data-driven process aiming at the analysis and interpretation of normative texts look like?
(ii) How can prevalent textual representation with normative content, e.g., laws, be transformed into models?
(iii) What are requirements for an application supporting the collaborative derivation of models using NLP technologies?

2 Analysis and Interpretation of Normative Texts

The potential of tool-support during the analysis and interpretation of normative texts and the subsequent formalization in semantic models can be described by a process model shown in Figure 1.

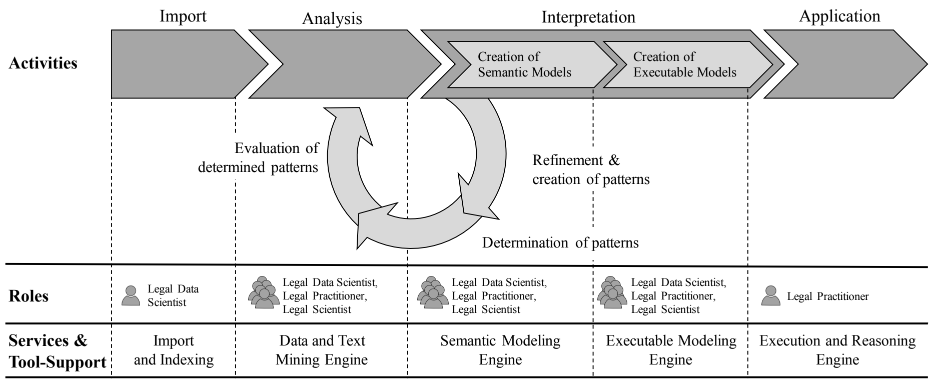

Import
Analysis
Interpretation
Application
Activities
Creation of Semantic Models
Creation of Executable Models
Evaluation of determined patterns
Refinement & creation of patterns
Determination of patterns
Roles
Legal Data Scientist
Legal Data Scientist, Legal Practitioner, Legal Scientist
Legal Data Scientist, Legal Practitioner, Legal Scientist
Legal Data Scientist, Legal Practitioner, Legal Scientist
Legal Practitioner
Services & Tool-Support
Import and Indexing
Data and Text Mining Engine
Semantic Modeling Engine
Executable Modeling Engine
Execution and Reasoning Engine

Fig. 1. Structured and data-centric process of collaboratively analyze, interpret, and model normative texts with specification of clear roles and potential tool-support.

The process is built upon the legal theory and interpretation canon proposed by Larenz and Canaris (see [2]) and consists of four steps:

1. **Import** Integration of relevant literature into the system requires a generic data model representing the particularities of normative texts, such as structural information. Depending on the case, that a legal expert wants to deal with, an import of various literature, e.g. laws, contracts, judgments, commentaries, etc., needs to be performed.

2. **Analysis** During the analysis step, legal experts explore and investigate the key norms and relationships between them. They implicitly determine the relevance of norms and their applicability by considering particular linguistic and semantic patterns which indicate whether a norm is relevant in a case, respectively a set of cases, or not.
3. **Interpretation** During the interpretation step the textual representation of norms is transferred into a coherent semantic and normative model (see Figure 2 and 3). This could be an explication of the mental model of an experience lawyer or legal scientist might has. The example in Section 3 illustrates how the determination of obligations, exclusions, prohibitions, etc. can be assisted during the interpretation process. It supports the legal experts to access the content of the law and prevents him of missing important norms.
4. **Application** The models arising from the textual interpretations can be stored in order to provide access to different collaborating users. Based on the interpretation and creation of the model the system automatically generates forms which allow end-users to describe a particular case. Thereby, the user inputs the known evidence from a fact and the system automatically infers the solution represented by derived attributes.

3 The Product Liability Act

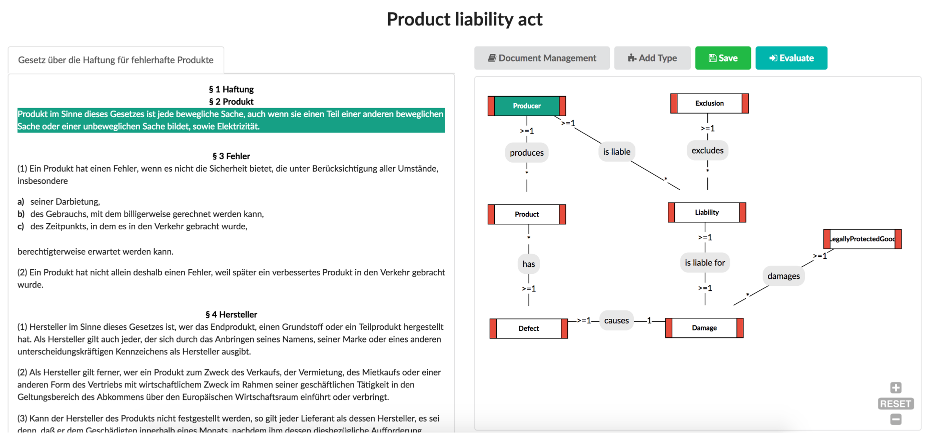
3.1 Semantic modeling to represent the structure and semantics of legal norms

Based on linguistic and semantic analysis of legal texts, it is possible to support the interpretation by proposing important semantic legal concepts.

The act explicitly states that there has to be a product (§2) with a defect (§3) causing a damage (§3) on a legally protected good (§1). This might be the life (in case of death) or the physical or psychological integrity (in case of injury) of a person or items that are possessed by a person. If this is the case, then the producer of the product (§4) becomes liable for the damage. The law also specifies reasons releasing the producer from his liability (§1). For sake of simplicity, we omit attributes of the types described by the legislator, such as the manufacturing date of a product. Obviously, it is possible—and depending on purpose of the modeling necessary—to deviate from the textual representation, by either modeling more or less information than provided in the text, e.g., interpretation. However, the proposed modeling approach does not make constraints regarding the quality of the model. This is intentionally left to the user.

3.2 Executable models representing decision structures and behavioral elements

Beside the semantic model representing types with their attributes and the relationships among them, the decision structure of norms has to be represented.



Gesetz über die Haftung für fehlerhafte Produkte

§ 1 Haftung
§ 2 Produkt
Produkt im Sinne dieses Gesetzes ist jede bewegliche Sache, auch wenn sie einen Teil einer anderen beweglichen Sache oder einer unbeweglichen Sache bildet, sowie Elektrizität.

§ 3 Fehler
(1) Ein Produkt hat einen Fehler, wenn es nicht die Sicherheit bietet, die unter Berücksichtigung aller Umstände, insbesondere

a) seiner Darbietung,
b) des Gebrauchs, mit dem billigerweise gerechnet werden kann,
c) des Zeitpunkts, in dem es in den Verkehr gebracht wurde,

berechtigterweise erwartet werden kann.

(2) Ein Produkt hat nicht allein deshalb einen Fehler, weil später ein verbessertes Produkt in den Verkehr gebracht wurde.

§ 4 Hersteller
(1) Hersteller im Sinne dieses Gesetzes ist, wer das Endprodukt, einen Grundstoff oder ein Teilprodukt hergestellt hat. Als Hersteller gilt auch jeder, der sich durch das Anbringen seines Namens, seiner Marke oder eines anderen unterscheidungskräftigen Kennzeichens als Hersteller ausgibt.

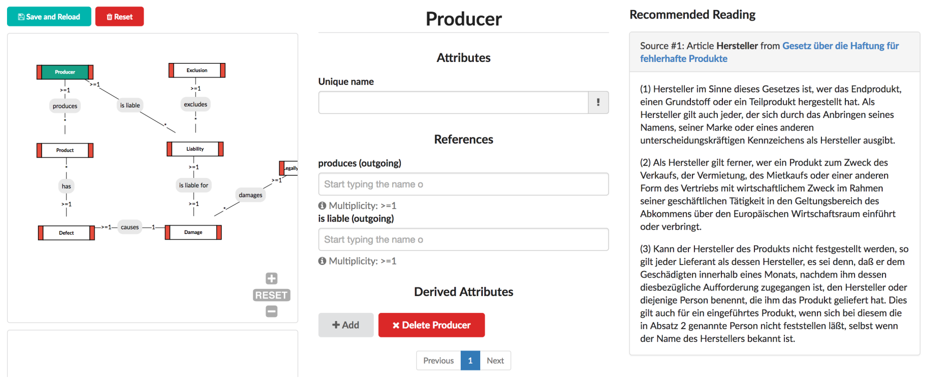
(2) Als Hersteller gilt ferner, wer ein Produkt zum Zweck des Verkaufs, der Vermietung, des Mietkaufs oder einer anderen Form des Vertriebs mit wirtschaftlichem Zweck im Rahmen seiner geschäftlichen Tätigkeit in den Geltungsbereich des Abkommens über den Europäischen Wirtschaftsraum einführt oder verbringt.

(3) Kann der Hersteller des Produkts nicht festgestellt werden, so gilt jeder Lieferant als dessen Hersteller, es sei denn, daß er dem Geschädigten innerhalb eines Monats, nachdem ihm dessen diesbezügliche Aufforderung

Fig. 2. Model based formalization of the product liability act in a web based environment. Types, attributes, and relations, can be linked with text, which is then highlighted.

This so-called executable model contains the decision structures, i.e. derived attributes. Consequently, the models aim at capturing the executable (or computable) logic that can be used to decide whether someone has a claim or not. Those derived attributes are expressed in a domain specific language (MxL) and are evaluated at run time based. Those statically type-safe expressions do not only support the specification of first-order propositional logic but also higher-order logic and almost arbitrarily complex arithmetical operations.

Figure 2 shows semantic model of the product liability act. Each type in the model contains different attributes and relations to other types.



Unique name

produces (outgoing)
Start typing the name o
Multiplicity: >=1
is liable (outgoing)
Start typing the name o
Multiplicity: >=1

Recommended Reading

Source #1: Article **Hersteller** from Gesetz über die Haftung für fehlerhafte Produkte

(1) Hersteller im Sinne dieses Gesetzes ist, wer das Endprodukt, einen Grundstoff oder ein Teilprodukt hergestellt hat. Als Hersteller gilt auch jeder, der sich durch das Anbringen seines Namens, seiner Marke oder eines anderen unterscheidungskräftigen Kennzeichens als Hersteller ausgibt.

(2) Als Hersteller gilt ferner, wer ein Produkt zum Zweck des Verkaufs, der Vermietung, des Mietkaufs oder einer anderen Form des Vertriebs mit wirtschaftlichem Zweck im Rahmen seiner geschäftlichen Tätigkeit in den Geltungsbereich des Abkommens über den Europäischen Wirtschaftsraum einführt oder verbringt.

(3) Kann der Hersteller des Produkts nicht festgestellt werden, so gilt jeder Lieferant als dessen Hersteller, es sei denn, daß er dem Geschädigten innerhalb eines Monats, nachdem ihm dessen diesbezügliche Aufforderung zugegangen ist, den Hersteller oder diejenige Person benennt, die ihm das Produkt geliefert hat. Dies gilt auch für ein eingeführtes Produkt, wenn sich bei diesem die in Absatz 2 genannte Person nicht feststellen läßt, selbst wenn der Name des Herstellers bekannt ist.

Fig. 3. The form is automatically generated and evaluated by the reasoning engine. End-users have to provided the facts, i.e., filling the form.

Thereby, it is necessary that the set of facts contains a product (§2) and a producer (§4) who has manufactured the product. In addition, there needs to be a defect (§3) causing a damage. In principle, producers are liable for damages that a product manufactured by him has caused. Of course, there are several exclusions that release the producer from his liability (§1). The existence or absence of those exclusions decides whether there is an effective liability or not. So if the producer is effectively liable and if there is a legally protected and damaged good, then the plaintiff has a claim according to the product liability act.

4 Technological requirements

The requirements (listed in Table 1) served as the base line for the implementation of the data science environment. To ensure an easy extension and adaptability, the system follows the principle of high cohesion and low coupling regarding its components. Technologically, the environment was implemented as a web application and the programming language used in the back-end was Java. Elasticsearch serves as the data storage, which allows an efficient handling of a large amount of textual data. The execution and reasoning engine, which is already existing and maintained at our research group, is accessed via a REST API. It fulfills all technological requirements to store the models [7]. The execution and reasoning engine integrates a DSL, i.e. MxL, which allows the specification of complex queries as well as logical and arithmetic operations [8].

5 Concept and implementation of a collaborative web application

Based on a case study of the product liability act presented in Section 3 and the process shown in Figure 1, several requirements can be derived that have to be met by a text mining and semantic modeling environment to foster collaboration. On this fundament we propose a reference architecture and an implementation.

5.1 Reference architecture

Based on the requirements from Table 1 and on the framework proposed in [9]) it is possible to define a reference architecture focusing on the analysis of legal texts and the creation of semantic and executable models.

The data and text mining engine is the central component of the platform supporting the modeling process by unveiling linguistic and semantic patterns. Since the main task consists of the creation of semantic and executable models based on textual descriptions that are usually of normative character, the assistance during the analysis and interpretation consists in parts of the automated detection of relevant sentences and phrases (patterns).

Within the data and text mining engine, several components have to be provided, e.g., dictionaries and pattern definitions (see also Section **??**). The

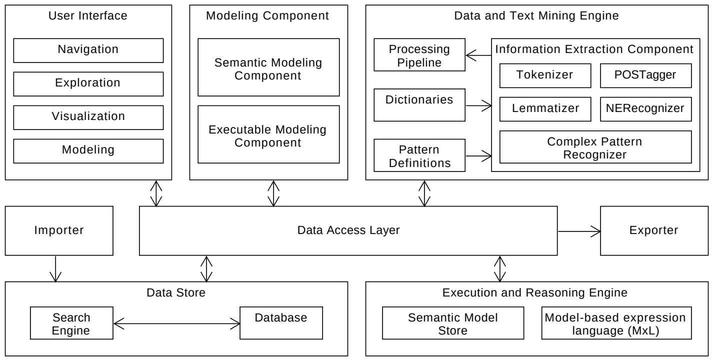


Fig. 4. The reference architecture for collaborative modeling of normative texts based on linguistic and semantic text analysis (extension of [9]).

legal language has some particularities that make it well suited for the analysis by NLP [10], for example the usage of particular linguistic patterns (phrases or words) that indicate particular legal ideas or concepts. This is what makes pattern determination valuable for the legal domain (see also [11]). In order to determine patterns which are more elaborate than common regular expressions, it is necessary to integrate a component that allows the specification of patterns which can easily nest, reuse and recombine those pattern specifications (rules). Our implementation integrates Apache Ruta (rule-based text annotation), which shows some great potential as a linguistic pattern definition technology [11].

Modeling component The modeling engine offers the required functionality to access, create, refine, maintain, and delete models. The differentiation between the semantic model and the executable model is necessary. In contrast to the semantic model, the executable model requires the definition of the executable semantics, i.e., functions and operators, between model elements, i.e., defining derived attributes based on atomic attributes. Figure 2 shows the types which are input, namely producer, product, etc., whereas "legally protected good" is a type, which is the output (relevant intermediate result) of an operator. The semantics of these operators can be expressed using an existing model-based expression language (as described above).
The modeling engine is also capable of managing the associations between model elements and text phrases. It offers proper technical functions and services, and is also the component that observes changes in the textual representation that might lead to changes within the semantic and executable models.

Execution and reasoning engine The execution of the model is done by an existing reasoning engine. Heart of the engine is a model-based domain specific expression language (MxL), that was developed at our research group [8]. It is a type-safe functional expression language, implemented in Java and hosted within a web application that can be accessed via a REST API. This expression language allows the specification of almost arbitrarily complex logical and numerical operations. These operations accept typed input values (integer, string, boolean, model elements, etc.) and compute a typed output. The operations are part of the interpretation process and can therefore be linked to the text.

6 Conclusion

In this paper we have developed a collaborative data-driven process that structures the analysis and interpretation of normative texts by leading to executable models of statutory texts, i.e., laws.

We have identified four phases that are required to formalized legal norms: Import, analysis, interpretation, and application (see research question i). For each step we have identified and implemented tool-support.

In a case study we have shown the approach by formalizing a basic claim arising from the product liability act. Based on the proposed data- and tool-centric process we have derived key requirements. Finally, we showed how the co-existence of these semantic models and the corresponding textual representation can be implemented in a collaborative web environment.

References

1. Bundesministerium des Innern, "Gemeinsame Geschäftsordnung der Bundesministerien," Berlin.
2. K. Larenz and C.-W. Canaris, *Methodenlehre der Rechtswissenschaft.* Berlin [u.a.]: Springer, 1995.
3. J. Hage, "A theory of legal reasoning and a logic to match," *Artificial Intelligence and Law*, vol. 4, pp. 199–273, 1996.
4. A. Rotolo, *Legal knowledge and information systems: JURIX 2015 : the twenty-eighth annual conference*, ser. Frontiers in artificial intelligence and applications. Amsterdam: IOS Press, 2015.
5. *ICAIL '15: Proceedings of the 15th International Conference on Artificial Intelligence and Law.* New York, NY, USA: ACM, 2015.
6. T. M. van Engers and R. van Doesburg, "First steps towards a formal analysis of law," *Proceedings of eKNOW*, 2015.
7. T. Reschenhofer, M. Bhat, A. Hernandez-Mendez, and F. Matthes, "Lessons Learned in Aligning Data and Model Evolution in Collaborative Information Systems," *Proceedings of the International Conference on Software Engineering (accepted for publication)*, 2016.
8. T. Reschenhofer, I. Monahov, and F. Matthes, "Type-safety in EA model analysis," *IEEE EDOCW*, 2014.
9. B. Waltl, F. Matthes, T. Waltl, and T. Grass, "LEXIA: A data science environment for Semantic analysis of german legal texts," *Jusletter IT*, 2016.
10. E. Francesconi, Ed., *Semantic processing of legal texts: Where the language of law meets the law of language.* Springer, 2010.
11. M. Grabmair, K. D. Ashley, R. Chen, P. Sureshkumar, C. Wang, E. Nyberg, and V. R. Walker, "Introducing LUIMA: An Experiment in Legal Conceptual Retrieval of Vaccine Injury Decisions Using a UIMA Type System and Tools," in *ICAIL '15: Proceedings of the 15th International Conference on Artificial Intelligence and Law.* New York, NY, USA: ACM, 2015, pp. 69–78.

Import		
1	Flexible import structure	Baseline for the analysis and interpretation is the consideration of various literature (laws, judgments, contracts, commentaries, etc.) that is present in different sources (xml, html, pdf, etc.).
2	Mapping and indexing of legal data	The legal literature has to be indexed and mapped to a data model, that does not only preserve the content, i.e. text and metadata, but also structural properties, such as references and nested content.
Analysis		
3	Preserving textual representation	Enabling users to access the content, i.e. legal literature. The visualizations of legal literature has to show the structural information, such as nestedness and links between articles and documents.
4	Collaborative creation and maintenance of patterns	The creation, refinement and deletion of the required pattern definitions should be done collaboratively in the application, so that different users are able to share their knowledge and contributions.
5	Lifecycle management of pattern descriptions	Support of the full lifecycle of the pattern specifications, namely creation, refinement, evaluation, and maintenance.
6	Automated pattern detection	Automated identification of linguistic and semantic patterns through data and text mining components.
7	Reuse of existing NLP components	Building of NLP pipelines, that allow the easy reuse and sharing of highly specified software components for NLP.
8	Evaluation of annotation quality	Possibility to view the annotations, to examine precision and recall manually, or to export this information to compare against a manually tagged corpus.
9	Manually annotating and commenting of legal texts	Users should be able to manually add relevant semantic information and comments to the legal literature.
10	Storing of annotations	Storing and indexing the automatically determined and manually added annotations.
Interpretation		
11	Creation of semantic and executable model elements	Step-wise definition of model elements (types, attributes, relationships, operators) for semantic and executable models.
12	Lifecycle support for semantic models	Defining, maintaining and storing of static model elements, such as types, attributes, relationships.
13	Lifecycle support for executable models	Defining, maintaining and storing of executable model elements, such as types, relationships, operators.
14	Connecting model elements with text phrases	Creation of connections between model entities and the relevant (interpreted) text. Thereby various levels of the interpreted text should be linkable to model elements, such as words, phrases, sentences, sections, and documents.
15	Domain specific language (DSL) to express semantics of operators	Specification of the operations and executable semantics of relationships with a model-based expression language.
Application		
16	Access to existing models	Viewing and exploring of semantic and executable models to grasp the result of prior interpretation processes.
17	Application of decision models	Executing the defined models through intelligent form-based or spreadsheet-based reasoning.

Table 1. Main requirements for collaborative tool-support to model the semantics of statutory texts structured into four phases.