An Integration of Software Engineering Methods and Semantic Technologies for Drafting and Modeling Statutes and Legal Rules

Martin Stabauer[1] **Gerald Quirchmayr**[2] **Johann Höller**[1]

[1] Department of Data Processing in Social Sciences, Economics and Business
Johannes Kepler University Linz
Linz, Austria
Email: {`martin.stabauer, johann.hoeller`}`@jku.at`

[2] Research Group Multimedia Information Systems
University of Vienna
Vienna, Austria
Email: `gerald.quirchmayr@univie.ac.at`

Abstract

The semantic representation and modeling of legal texts has for a long time been a significant research challenge. While approaches from both, software engineering and semantic modeling, have led to impressive results, some gaps are still remaining. This paper tries to bridge the gap between generalizability and applicability by combining semantic modeling with traditional software engineering processes.

A framework for drafting legislation was implemented in OWL, SWRL, various web-based technologies and Java using the Jena framework. Links to external ontologies were made via semantic relations, following the principles of linked open data. A base model including ontologies, semantic rulesets and additional algorithms were developed and amended by a general development process for implementing diverse fields of law.

The base model and process suggested in this paper were then tested in an extensive case study, which clearly documented the approaches benefits, such as increased efficiency of the modeling process, automatic consistency checking, compatibility with established standards in legal semantics, and the reusability of base classes underlying the developed models. The case study addresses curricula based on Austrian legislation in depth, which are exemplarily covered as a whole.

Keywords: Semantics; legal semantics; legislative texts; curricula; law-making processes; linked data.

1 Motivation and Background

The issue of expressing legal information and rules is traditionally solved by the usage of legislative texts. Developments in the field of semantic modeling have led to the question whether written text really is the one and only, optimal solution for the task of making the addressee understand the intentions of the legislator. Extensive research carried out over the past decades has led to very promising results based on formal logic and artificial intelligence technologies.

While early research suggested that the usage of expert systems in law would provide members of legal professions with extensive knowledge and interpretation of law and therefore would reduce dependence on experts and even replace them in certain cases (Susskind 1987), later work especially in the U.S. focused more strongly on technologies like case-based reasoning (e.g. Ashley 1991, Rissland et al. 2003).

Further pioneering work include the publications of Thorne McCarty (McCarty 1977), Marek Sergot and Robert Kowalski (Sergot et al. 1986) and Trevor Bench-Capon (Bench-Capon et al. 1987). Leading French and German research is very adequately described by Bourcier (1995), Fiedler et al. (1984) and Fiedler & Traunmüller (1986), respectively. Other examples of representative research carried out in the 1980s and 1990s are presented in Gordon (1987), Greenleaf et al. (1987) and Van Noortwijk et al. (1991).

The continuing importance of research in this field is shown in Schefbeck (2009). The author claims that structuring legal norms in XML and contextualizing them in RDF or OWL will be the future. From a jurisdictional point of view, the main task will be to describe legal norms and their relations to other norms and context elements, whereas in a global scope there will be the need to model cross-language, normative equivalences. All of that is said to happen in a mostly IT-supported way.

Some major advantages of semantic systems in comparison to the written textual communication of legal texts are listed in Höller & Ipsmiller (2009):

- Better quality of legal norms through faster unveiling of inner contradictions and incompleteness.
- Improvements in the law-making process by common understanding and terminology. Individuals with different levels of information have the chance to learn about certain topics in a very specific way.
- Improvements in general communication. Words often have several meanings, which can lead to misunderstandings and barely comprehensible texts. By contrast, semantic models claim to be unambiguous.
- In case of changes in the legal norms in question an impact analysis can be done more easily and effectively.

- A better and easier implementation of IT systems that are based on the legal norms in question due to a more direct transmission of information.

Building on the available literature, this paper now tries to close one of the remaining gaps by presenting a practical approach to modeling both the content of the field of law in question and the legal text, which it is represented with. While not going so far as to try and automatically draft legislation, like described in e.g. Winkels & Den Haan (1995), the proposed framework does enable a model developer to combine the benefits of legal texts with state-of-the-art technologies in software engineering and the advantages of semantic models.

2 Related Work

The research presented in this paper is intending to close the gap between the current approaches of very generic legal ontologies and the traditional means of software engineering, which are usually aimed at very specific issues. Legal semantics were developed with the intention in mind that they should fit every thinkable branch of law and every thinkable field of application. This aspiration requires a high level of abstraction, which in turn results in most of these ontologies not being directly applicable.

One of the most widely recognized variants of these very valuable contributions is LKIF, the Legal Knowledge Interchange Format developed by the EU-funded Estrella Project ('European project for Standardized Transparent Representations in order to Extend Legal Accessibility'). Breuker et al. (2008) describes the 15 modules included, which are meant to keep the system clear and tidy. Figure 1 shows an overview.

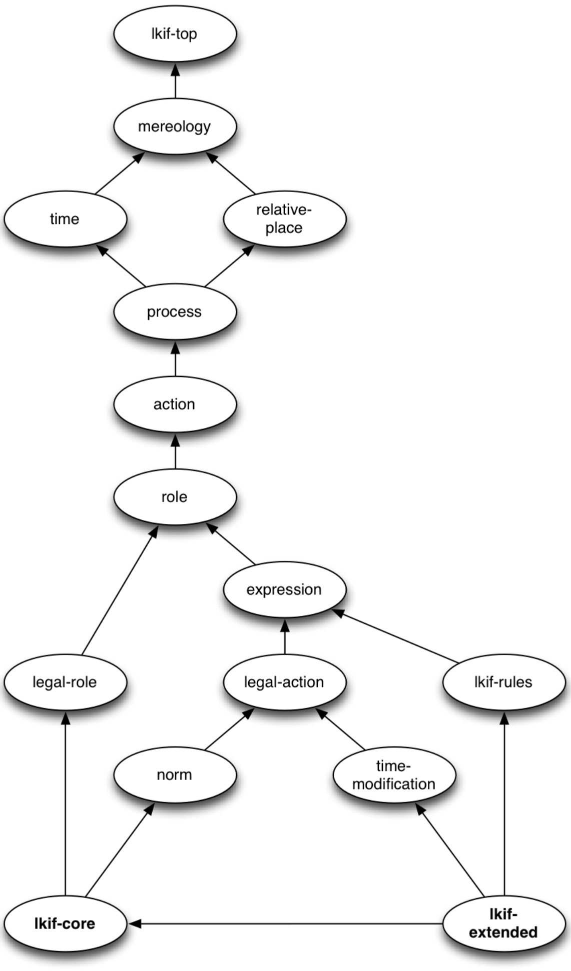

Figure 1: LKIF Overview

LKIF allows legal knowledge bases to be represented in OWL, the Web Ontology Language (McGuiness & Harmelen 2004). This includes specific terminology, rules and normative statements. The rules in LKIF extend the Semantic Web Rule Language standard (SWRL) described in Horrocks et al. (2004), which itself is a combination of OWL and RuleML (Boley et al. 2001).

Apart from this, two generally applicable main categories of IT systems in the field of law have been identified in the early 1980s (e.g. McCarty & Sridharan 1982, Bing 1984): storing and consequently retrieving legal information on the one hand and legal analysis systems on the other. Building on the experience gained by a series of projects in the late 1980s the issue of logic based representations of laws was again stressed in (Kowalski 1995). Concerning text analysis, the works of Erich Schweighofer (e.g. Schweighofer 1999) belong to the most remarkable.

With ontologies being more expressive than previous modeling mechanisms and IT infrastructures offering substantially greater power, some of the ideas introduced by researchers in the 1980s and 1990s are now becoming practically relevant. According to Lauritsen (2013) this leads to a series of interesting new research challenges.

The importance of the field is again underlined by Springer planning to publish a handbook on legal reasoning and argumentation in the near future (Bongiovanni et al. 2015).

3 Semantic Model and Modeling Process

The major issue of models used today is that most of them are either very abstract or not reusable. The lack of reusability often is a result of modeling a specific field of knowledge without considering additional areas of application. How this gap (see Figure 2) is closed by our integrated model is outlined in this section of the paper. While the direct approach using either semantic modeling tools or software engineering methods in an isolated way leads to a series of impasses, the combined model yields the desired results with considerably less effort. By integrating the semantic base model with a software-engineering oriented process model the implementation of various legal regulations becomes possible, reducing the effort through employing generalized base classes and following a guided process. To illustrate the benefits of this strategy, the results of an extensive case study are presented in section 4.

The approach used in this research consists of a semantic base model that can be used to model various fields of law, and a matching development process model to help implement a specific solution for a selected application domain. A supporting framework was developed to help ensuring consistency with existing standards and external ontologies. This framework combines a semantic ruleset with elements of traditional software development processes.

3.1 Semantic Base Model

The semantic base model lays the foundation for the modeling process described in this section. It is constructed as a connection of base classes that are inte-

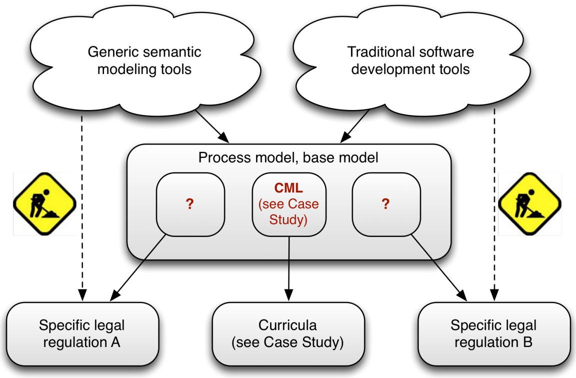
Generic semantic modeling tools
Traditional software development tools
Process model, base model
?
CML (see Case Study)
?
Specific legal regulation A
Curricula (see Case Study)
Specific legal regulation B

Figure 2: Solution through an integration of semantic models and software engineering

grated with an ontology. The underlying formal concept is inspired by the LKIF model briefly described in chapter 2. Depending on the modeled field of law extensions can be added as necessary. In most cases it is these extensions that make the model directly applicable. The effort needed to add them to the base model highly depends on the desired application. The more structured and isolated the specific legal regulation is, the easier it is to be implemented in our base model. The benefit is highest for fields of law where automatic calculations build the focus. An example could be tax law, where the ontology itself could deliver answers to questions like ‘What is the tax payable for individual X?’.

The model includes both the contents of the field of law in question and the conceptual structure. Therefore, it is not only possible to use the legal knowledge covered in the ontology for whatever application is desired, but also to automatically create and restore the semantically correct and complete human-readable legal text, even in multiple languages.

Another feature of the model is that it allows the connection to external ontologies via suitable semantic relations. These links follow the principles of linked open data as referred to in Bizer et al. (2008).

An important decision was, which sublanguage of OWL (Hitzler et al. 2012) to select. We decided in favor of OWL DL, as it allows a reasonable compromise between elegant, high-level expressiveness and decidability in the ontology. As future implementations of the base model might not want to use all the expressions included in OWL 2 Full, it seems a good idea to restrict the base model to a certain amount of expressiveness in exchange for guaranteed decidability and the availability of practical reasoning algorithms.

The following figure gives an outline of the base models structure and selected connections to external ontologies.

This base model is used for checking the consistency of specific implementations and for modeling and creating the domain specific legal text, as shown in more detail in the case study in chapter 4.

3.2 Automated Creation of Legal Text

The automated creation of legal text is mainly achieved by a specific semantic structure that is used as input for algorithms written in Java. These algorithms then recursively construct a human-readable version of the legal ontology in the form of a classic legal text. As the basic conceptual structure of the modeled law is part of the ontology, it can be used to create the structure and hierarchy of the legal text. Therefore a set of basic classes is introduced, which can be used for a wide range of fields of law.

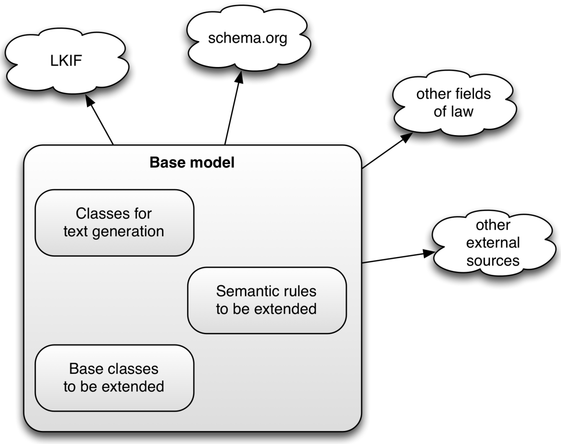
LKIF
schema.org
other fields of law
other external sources
Base model
Classes for text generation
Semantic rules to be extended
Base classes to be extended

Figure 3: Base model

In essence, these basic classes represent the basic building blocks of legal regulations such as sections, paragraphs, subparagraphs etc. and their hierarchy. All these classes as well as an additional utility class called *Content* are subclasses of the class *Textblock*, which provides the basic functionalities for numbering and organizing the legal text. These functionalities are modeled via included meta information and relations to other *Textblock*s.

The following example illustrates the creation of legal and statutory texts using the example of one sample paragraph of a Master curriculum. Figure 4 shows the structure of this paragraph in the ontology, which can be turned into legal text when combined with straightforward Java algorithms. These create the hierarchy of paragraphs, subparagraphs etc. recursively, place the correct objects and relations at the corresponding positions and produce the accurately formatted legal text.

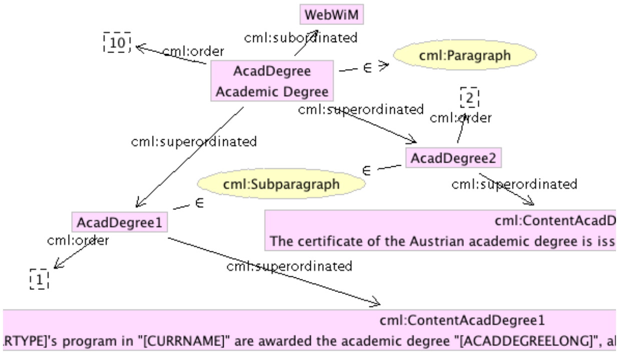

Figure 4: Automated Text Creation

The main algorithms for a textual output within the classes of the legal regulation itself and all sorts of *Textblock* then include specific formatting functions and a recursion like follows:

```
StringBuffer sb = new StringBuffer();
Collections.sort(textblocks);
for(Textblock b : textblocks) {
    sb.append(b.toString());
}
```

In the next step a development process model is needed to ensure the satisfaction of requirements to guarantee the executability of the text-generating algorithms, semantic rules and consistency checks.

3.3 Development Process

The development process described in this section focuses on guiding a modeler through the development of a domain specific implementable representation of a law or statute. Drawing on experience gained in the field of software development, this process is based on the classic spiral model, described in Boehm (1988). The major advantage of the spiral model in our context is that it covers the necessity of representing continuously expanding legislation in a stepwise approach. Every time there are changes in the respective field of law, a new iteration starts by refining and/or modifying the last iteration.

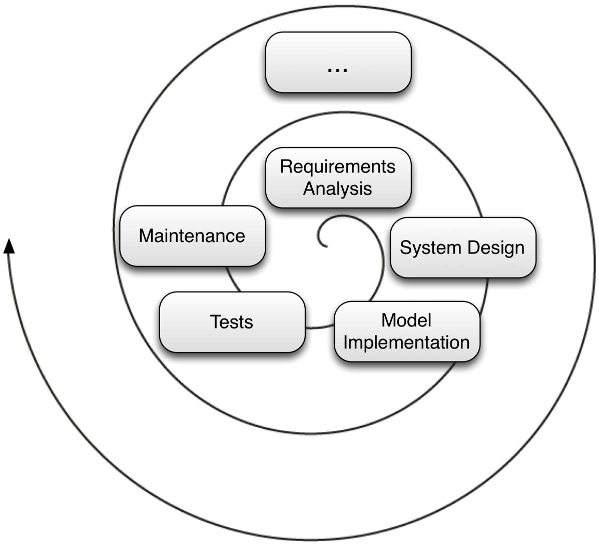

Figure 5: Spiral model

Figure 5 shows the process model, which consists of the following activities:

- Requirements analysis

  As a first step, the needs and requirements of all the involved parties need to be analyzed. Often, there are several user groups with diverging technical and juristic understanding. In order to completely capture and consider all these requirements, Bourque & Fairley (2014) suggest the following structure, which is very suitable for our demands:

  - Requirements elicitation
  - Requirements analysis
  - Requirements specification
  - Requirements validation

  As a minimum, the requirements of the legislator and the addressee are to be analyzed. However, often there are several more stakeholders involved in the process. Furthermore, legal prerequisites have to be regarded, and there could be special needs like multilingualism, version control, accessibility, etc.

- System design

  The next step is the design of the overall system with the main objectives of reduction of complexity and minimization of the risk of maldevelopment. For example, the following questions need to be answered when implementing the model in a specific field of law:

  - Should the whole field of law be modeled or just parts of it?
  - What is the structure of the model, how are relations to other legal regulations implemented?
  - Are there external sources (e.g. already modeled regulations) that can be used?
  - Which semantic language, which rule language, which programming language shall be used? Which frameworks can be used?
  - Is there a need for one or more user interfaces?

- Model implementation

  The next task is to develop the semantic model, all required rulesets and user interfaces. The variation between the respective fields of law is greatest in this phase: The outcomes can differ a lot, depending on the specific legal regulation to be implemented. This is due to highly diverse legal prerequisites, different external relations and varying structures. Nonetheless, the base model ensures that certain basic elements are in place so that compatibility is guaranteed.

- Tests

  One first test is the concretization of the model itself: A specific legal regulation is implemented and the results are validated. A successful implementation can be seen as a sign for a correct base model; this is shown in the case study in chapter 4. In many cases, additional tests are useful, Bourque & Fairley (2014) specify various tests especially designed for the field of software engineering.

- Maintenance

  There are two main types of maintenance that are of high relevance in this context (*IEEE Standard Glossary of Software Engineering Terminology* 1990):

  - Corrective and perfective maintenance aims to improve the software, rules and/or ontology by rectifying errors, enhancing algorithms or models or implementing new functional features.
  - Adaptive maintenance tries to cope with changes in the systems environment by modifying the software, rules and/or ontology. Some legal regulations are modified regularly and the implemented model has to handle these changes. This way the models life cycle can be extended in a setting very likely to change.

The modeling process described above will be applied to a challenging and complex case study in the next section of this paper. This case study will test our approach against a hierarchy of different laws and statutes and a highly complicated regulation drafting process.

4 Case Study: Curriculum Development based on Austrian Legislation

As an example for the base model described above a concrete implementation was developed. This implementation acts as proof of concept for the base model. It shows that it is possible to model a whole coherent legal text using an integration of the aforementioned techniques.

4.1 Environment and Setting

The Austrian legislation regarding universities and other institutions of higher education has gone through enormous changes during the last years. A new university law was introduced, giving the universities greater flexibility and full legal capacity.

One of the main changes was the new duty to enact statutes, in which each university applies its own regulations including those regarding curricula. As a consequence, an IT system handling curricula has to cope with all these laws and statutes.

Another important influence of the law-making process of curricula is the so-called Bologna process, which introduced the framework of Bachelor and Master degrees as well as the European Credit Transfer System ECTS (*ECTS Users' Guide* 2009). The ECTS credits standardize study attainment and make the performance of students of higher education comparable across the EU.

4.2 Implementation of the Base Model

As a proof-of-concept of the base model described in chapter 3.1 a new set of ontologies was implemented. It is called CML ('Curriculum Modeling Language') and allows modeling curricula based on Austrian legislation. In order to act as an implementation of the base model all of the base classes and relations were included and enhanced in CML.

4.2.1 Ontologies

Table 1 shows the ontologies modeled in CML.

Ontology	Content
cml.owl	base structure and imports of external ontologies
cml_uni.owl	models universities and other institutes of higher education
cml_curr.owl	models a curriculum and its components
cml_templ.owl	templates, exemplary data and text, translations
cml_rules.owl	SWRL rules for automated validation
cml_ext.owl	mapping to external ontologies

Table 1: CML ontologies

4.2.2 Semantic Rules

CML makes extensive use of SWRL. Three main types of rules are used in CML to validate a modeled curriculum:

- Requirements that must be fulfilled for a curriculum to be valid. E.g. Bachelor curricula compliant with the statutes at Johannes Kepler University Linz have to include at least one course with a gender-related subject and at least 3 ECTS. This can be validated by the following SWRL rule, which categorizes a correct curriculum as *CurriculumWithGenderCourse*:

$$\begin{aligned}&Curriculum(?curr) \wedge Course(?c)\\ &\wedge indirectSub(?c, ?curr) \wedge genderTopic(?c, true)\\ &\wedge ects(?c, ?ects) \wedge swrlb: greaterThan(?ects, 2)\\ &\Rightarrow CurriculumWithGenderCourse(?curr)\end{aligned}$$

- Rules that draw the attention of the relevant authorities to certain aspects. E.g. a Bachelor curriculum normally includes 180 ECTS, but this can be changed under certain conditions. The following SWRL rule checks if the sum of ECTS credits of a curriculum matches the standard. If the numbers differ, the rule categorizes the curriculum as *CurriculumSpecialECTS*:

$$\begin{aligned}&Curriculum(?curr) \wedge hasType(?curr, ?type)\\ &\wedge ects(?curr, ?ects) \wedge ects(?type, ?standard)\\ &\wedge swrlb: notEqual(?ects, ?standard)\\ &\Rightarrow CurriculumSpecialECTS(?curr)\end{aligned}$$

- The third category consists of rules that cannot be modeled reasonably using semantic technologies. In most cases this is because of the open-world assumption (OWA) that is made in OWL and other languages of the semantic web. The main idea of OWA is that if a statement is not known it is not automatically false but simply unknown. This way one can distinguish false statements from missing statements (Grimm 2010).

  Numerous rules in this category are sum-based, like the following example: We want to know, whether the ECTS of a curriculum match the sum of the courses subordinated. Figure 6 shows the issue: In OWA we do not know if the courses really are the only ones subordinated, there could always be another course formerly left out of consideration. A possible solution for this problem would be to add closure axioms, thereby creating an artifical closed-world. This adds a lot of axioms to the ontology and every change has to be done at various places.

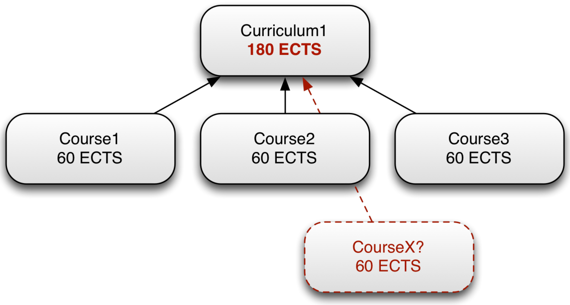

Figure 6: Issues with OWA

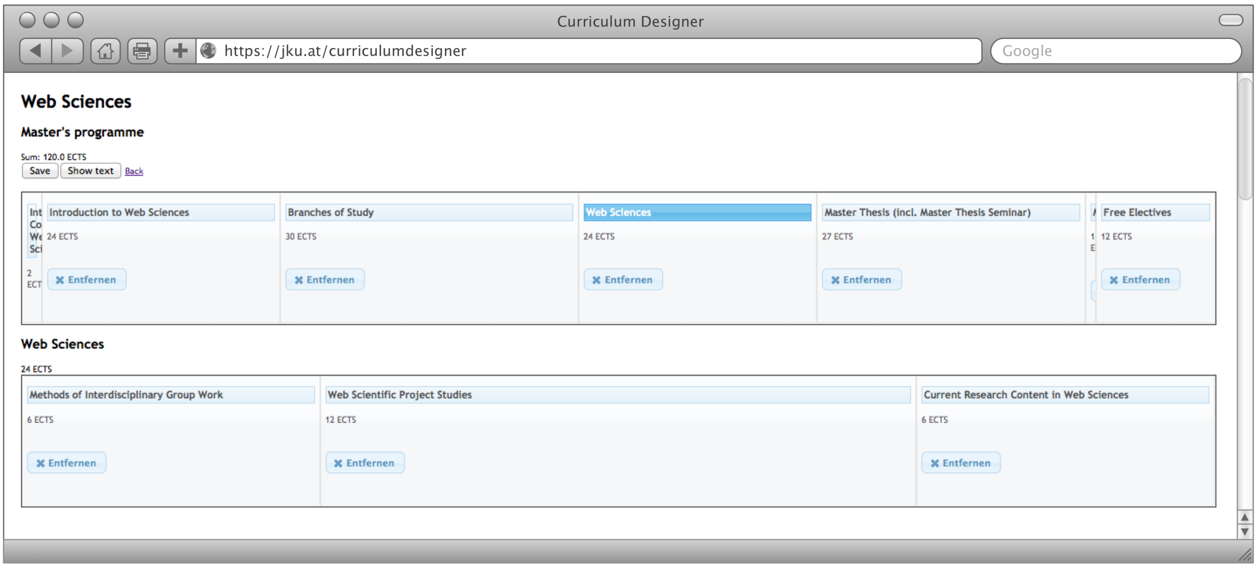

Figure 7: Curriculum Designer

Another solution would be the usage of SQWRL, which allows querying an ontology as it could be done in a closed-world environment (O'Connor & Das 2009). In this case we use SQWRL to sum up the ECTS of the courses and compare them with the curriculum.

$$\begin{aligned}
&Curriculum(?curr) \land Course(?c) \\
&\land sub(?c, ?curr) \land ects(?curr, ?currects) \\
&\land ects(?c, ?ects) \land sqwrl : sum(?sum, ?ects) \\
&\land swrlb : notEqual(?sum, ?currects) \\
&\Rightarrow CurriculumWrongECTS(?curr)
\end{aligned}$$

As SQWRL is not yet fully supported by some of the most widely recognized frameworks, our approach makes use of simple Java algorithms to perform the desired checks.

We now have demonstrated the implementation of different categories of semantic rules needed to validate a modeled curriculum. The next section covers an exemplary GUI developed to make life easier for the various users of the CML ontology. Understandably, such GUIs are not a necessity in all thinkable applications of the base model.

4.2.3 Graphical User Interfaces

Although it is possible to create and edit a model conform to CML by using generic tools like Protégé (*Protégé* 2014), their complexity and wide range of functions might confuse the not too advanced users. There are numerous groups of users, all of which have been determined during the requirements analysis and who have their own needs, technical and juristic understanding. This arises the necessity for specific user interfaces for each user group.

One exemplary GUI that has been developed for the study commissions in charge of creating a new curriculum from scratch is the Curriculum Designer. This web-based piece of software is written in Java and JSP and allows the modelers of curricula to create the basic building blocks of a new curriculum and output a coherent and complete model in OWL. Figure 7 shows a screenshot of the Curriculum Designer.

4.3 Results

The results of the case study are very promising. We turned several curricula into semantic models and back into legal texts that turned out to be equal to the original material and therefore appear to be compatible with human-readable text. The basic structure including the most important information of a typical Austrian Master curriculum including around 300 to 400 course classes results in an ontology of around 4000 to 5000 axioms, which can easily be handled by the algorithms and frameworks described. Naturally, the number of axioms increases as more information regarding courses or more comprehensive rulesets are added or when additional languages are included. However, this increase is moderate and happens linearly, so that no performance issues arose at any time.

5 Conclusion

In this paper we have presented an approach for modeling legal texts based on an integration of semantic models and traditional software engineering technology. This has allowed us to bridge the gap between abstract and often too generic representations and very specific implementations that have little value for reuse. In the case study the applicability to a very complex domain was shown that comprises constitutional legislation, simple laws, administrative statutes and local university regulations. Hence, this experiment has also demonstrated the ability of our approach to cope with different hierarchies of legislative and statutory materials.

Eventually, the major contribution of the research described in this paper is the provision of a tool-supported method that significantly simplifies the representation of legislation in an executable form and also increases the reusability of the developed models.

References

Ashley, K. D. (1991), *Modeling legal arguments: Reasoning with cases and hypotheticals*, MIT press.

Bench-Capon, T. J. M., Robinson, G. O., Routen, T. W. & Sergot, M. J. (1987), Logic programming for large scale applications in law: A formalisation of supplementary benefit legislation, *in* 'Proceedings of the 1st international conference on Artificial intelligence and law', ACM.

Bing, J. (1984), *Handbook of legal information retrieval*, Elsevier Science Inc.

Bizer, C., Heath, T., Idehen, K. & Berners-Lee, T. (2008), Linked data on the web, *in* 'Proceedings of the 17th international conference on World Wide Web', ACM, pp. 1265–1266.

Boehm, B. W. (1988), 'A spiral model of software development and enhancement', *IEEE Computer* **21**(5), 61–72.

Boley, H., Tabet, S. & Wagner, G. (2001), Design rationale of RuleML – a markup language for semantic web rules, *in* 'Proceedings of the Semantic Web Working Symposium', pp. 381–401.

Bongiovanni, G., Postema, G., Rotolo, A., Sartor, G. & Walton, D., eds (2015), *Handbook of Legal Reasoning and Argumentation*, Springer Netherland.

Bourcier, D. (1995), *La décision artificielle: le droit, la machine et l'humain*, Presses universitaires de France.

Bourque, P. & Fairley, R. E., eds (2014), *SWEBOK 3.0 – Guide to the Software Engineering Body of Knowledge*, IEEE Computer Society.

Breuker, E. et al. (2008), 'Estrella project – LKIF', http://www.estrellaproject.org.

ECTS Users' Guide (2009), http://ec.europa.eu/education/tools/docs/ects-guide_en.pdf.

Fiedler, H., Barthel, T. & Voogd, G. (1984), *Untersuchungen zur Formalisierung im Recht als Beitrag zur Grundlagenforschung juristischer Datenverarbeitung: (UFORED)*, Westdt. Verlag.

Fiedler, H. & Traunmüller, R. (1986), Formalisierung im Recht und juristische Expertensysteme, *in* 'GI – 16. Jahrestagung II', Vol. 127, Springer Berlin Heidelberg, pp. 367–382.

Gordon, T. F. (1987), Oblog-2: A hybrid knowledge representation system for defeasible reasoning, *in* 'Proceedings of the 1st international conference on Artificial intelligence and law', ACM.

Greenleaf, G., Mowbray, A. & Tyree, A. L. (1987), Expert systems in law: The datalex project, *in* 'Proceedings of the 1st international conference on Artificial intelligence and law', ACM.

Grimm, S. (2010), 'Knowledge representation and ontologies', *Scientific Data Mining and Knowledge Discovery – Principles and Foundations* pp. 111–137.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F. & Rudolph, S. (2012), 'OWL 2 web ontology language primer (second edition)', http://www.w3.org/TR/owl2-primer/.

Höller, J. & Ipsmiller, D. (2009), Semantik – was wir immer schon wussten: Semantische Technologien in der Unterstützung von Rechtsetzungsprozessen, *in* E. Schweighofer, ed., 'Semantisches Web und Soziale Netzwerke im Recht: Tagungsband des 12. Internationalen Rechtsinformatik Symposions', pp. 141–146.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B. & Dean, M. (2004), 'SWRL: A semantic web rule language – combining OWL and RuleML', http://www.w3.org/Submission/SWRL.

IEEE Standard Glossary of Software Engineering Terminology (1990), number 610.12-1990, IEEE.

Kowalski, R. A. (1995), 'Legislation as logic programs', *Informatics and the Foundations of Legal Reasoning* pp. 325–356.

Lauritsen, M. (2013), 'Are we free to code the law?', *Communications of the ACM* **56**(8), 60–66.

McCarty, L. T. (1977), 'Reflections on taxman: An experiment in artificial intelligence and legal reasoning', *Harvard Law Review* **90**(5), 837–893.

McCarty, L. T. & Sridharan, N. (1982), A computational theory of legal argument, Technical Report LRP-TR-13, Laboratory for Computer Science Research, Rutgers University.

McGuiness, D. L. & Harmelen, F. v. (2004), 'OWL web ontology language overview', http://www.w3.org/TR/owl-features.

O'Connor, M. J. & Das, A. K. (2009), SQWRL: A query language for owl, *in* R. Hoekstra & P. F. Patel-Schneider, eds, 'OWLED'09'.

Protégé (2014), http://protege.stanford.edu.

Rissland, E. L., Ashley, K. D. & Loui, R. (2003), Ai and law: A fruitful synergy, *in* 'Artificial Intelligence 150.1'.

Schefbeck, G. (2009), Per Anhalter durch das Legiversum – Rechts- und Legislativinformatik 2.0, *in* E. Schweighofer, ed., 'Semantisches Web und Soziale Netzwerke im Recht: Tagungsband des 12. Internationalen Rechtsinformatik Symposions', pp. 47–55.

Schweighofer, E. (1999), *Legal knowledge representation: automatic text analysis in public international and European law*, Vol. 7 of *Kluwer Law International*, Kluwer.

Sergot, M. J., Sadri, F., Kowalski, R. A., Kriwaczek, F., Hammond, P. & Cory, H. T. (1986), 'The british nationality act as a logic program', *Communications of the ACM* **29.5**, 370–386.

Susskind, R. E. (1987), *Expert systems in law: A jurisprudential inquiry*, Oxford University Press.

Van Noortwijk, C., Piepers, P. A. W., van der Wees, J. & De Mulder, R. (1991), The juricas-system: new applications and future development, *in* 'Proceedings of the 3rd international conference on Artificial intelligence and law', ACM.

Winkels, R. & Den Haan, N. (1995), Automated legislative drafting: Generating paraphrases of legislation, *in* 'Proceedings of the 5th international conference on Artificial intelligence and law', ACM, pp. 112–118.