Requirements for Logical Models for Value-Added Tax Legislation

Nielsen, Morten Ib; Simonsen, Jakob Grue; Larsen, Ken Friis

Publication date:
2008

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (APA):
Nielsen, M. I., Simonsen, J. G., & Larsen, K. F. (2008). Requirements for Logical Models for Value-Added Tax Legislation. Paper presented at International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Doha, Qatar.

Requirements for Logical Models for Value-Added Tax Legislation

Morten Ib Nielsen* Jakob Grue Simonsen
Ken Friis Larsen*
Department of Computer Science, University of Copenhagen, Denmark
{mortenib|simonsen|kflarsen}@diku.dk

Abstract

Enterprise resource planning (ERP) systems are ubiquitous in commercial enterprises of all sizes and invariably need to account for the notion of value-added tax (VAT). The legal and technical difficulties in handling VAT are exacerbated by spanning a broad and chaotic spectrum of intricate country-specific needs. Currently, these difficulties are handled in most major ERP systems by customising and localising the native code of the ERP systems for each specific country and industry. We propose an alternative that uses logical modeling of VAT legislation. The potential benefit is to eventually transform such a model automatically into programs that essentially will replace customisation and localisation by configuration by changing parameters in the model. In particular, we: (1) identify a number of requirements for such modeling, including requirements for the underlying logic; (2) model salient parts of the Danish VAT law in Web Ontology Language (OWL) and in Configit Product Modeling Language (CPML).

1 Introduction

Virtually all businesses employ an *enterprise resource planning* (ERP) system[1] to help them keep track of the economical aspects of their business. Most of these ERP systems are designed and programmed in a generic fashion and later heavily adapted to particular geographical regions and specialised industries, indeed often heavily adapted to the specific company at which the system is deployed.

The adaptations to an ERP systems can be divided into two kinds: *customisations*, requiring changes to the source code of the systems, and *configurations*, requiring changes to be interpreted in an overlay system at runtime.

The adaptations made to an ERP system when it is to be deployed in a specific country is called *localisation*. Localisation is both the translation of the user-visible strings in system and, more importantly, the changes to the system with respect to the local legislation. Localisation can be performed both as customisation and as configuration.

We suggest to make all VAT localisation configuration, and to give this configuration in the form of a explicit logical *VAT model*. The benefits of this approach is to (1) drastically cut down the number of domain experts involved in the design process, (2) alleviate the implementation tasks of the programmer (as sizable chunks of the necessary code can be auto-generated), and (3) make the handling of VAT an explicit data object that, for exapmle, can be subjected to static analysis, for instance, it could be model-checked against a formalisation of the VAT legislation.

In this paper, we take the initial steps towards this goal by identifying a number of requirements needed by logic-based tools for VAT modeling. We detail our experiments with two such tools and evaluate them against the requirements thus laid out.

2 Modeling of VAT Legislation

Legal rules are usually accessible through legal documents such as the EU directive on the common system of value added tax, [5], or through official guidelines (that also carry legal weight) such as [3]

*Supported by the Danish National Advanced Technology Foundation through the 3gERP project.

[1] We use the term ERP system in a broad sense in this article.

and [10]. While the (legal) semantics of legal documents and guidelines are (at least intended to be) the same, the latter are usually easier to understand for laymen but carry less (formal) structure than legal documents. We use the term *legal sources* to refer to both legal documents as well as official guidelines.

Structure in legal documents plays an important role for instance in connection with references between exceptions and main rules, which are often separated from each other (and might even occur in different documents). Another use of structure is in connection with meta-principles referred to as *rules of collision* in legal doctrine. The purpose of these principles is to disambiguate in situations where legal rules are in conflict with each other. In this paper we only focus on *Lex Specialis* (the principle that particular norms suppress general norms. For example, that country-specific legislation in a particular EU country will in general override legislation common to all EU countries in case of non-crucial conflict).

3 Desired Properties of a Logical Model

This section accounts for some of the desired properties of logical models of legal rules. We use F to denote a formal language and F-model to mean a model of legal rules written in F.

Good Tool Support We envision that F-models are created in part by domain experts without a strong background in computer science or modeling. Thus modeling in F should be easy and provide good domain support. It is not reasonable to expect that domain experts interact with F directly, therefore we envision a GUI as a mediator between domain experts and F-models. Even though (usability) design of a GUI is outside the scope of this paper the interplay between GUI and F is important. One way to design a GUI is by using graphical notation as sugar for common constructions in F (e.g., like it is done in [11]).

Support the isomorphism principle This principle, as presented in [9], states that:

> A representation of a legal system is an isomorphism if (1) Each legal source is represented separately. (2) The representation preserves the structure of each legal source. (3) The representation preserves the traditional mutual relations, references and connections between the legal sources. (4) The representation of the legal sources and their mutual relations is separate from all other parts of the model, notably representation of queries and facts management.

For our application that means at least support for structure and references. And the loose coupling of main rule and exceptions seems to be the most difficult to achieve.

Conform to legal doctrine As described in Section 2, it is important to be able to express rules with exceptions, and to support meta principles such as Lex Specialis.

Be transferable Modeling of legal sources will most likely involve many people with different legal specialisations. Therefore, the combination of F and its accompanying editor should support good software engineering principles, including support for transfer of the model between persons with different areas of responsibility.

Sufficient expressive power F must have the ability to express the semantics of relevant rules and properties of the problem domain. It is clear that F must be able to handle at least Presburger Arithmetic (amongst other things due to the need for multiplication with VAT percentage constants) on integers. Good domain support also requires the use of variables over general finite domains, which is the reason why we have not considered using a SAT solver (even though any finite domain can be coded).

1. *Samples and items used for advertising. No VAT should be added to items used for advertising or the like if the value (purchase price) is less than 100 kr. exclusive of VAT nor to samples you hand out for free.* Translated from [10, p. 8].

2. *Sales outside EU (3rd countries). No VAT should be added to goods delivered to destinations outside the European Union, or to the Faroe Islands or Greenland. This fact ordinarily also applies to services, but VAT should be added to certain services.* Translated from [10, p. 9].

3. *Sales in other EU countries. No VAT should be added to goods delivered to companies in other EU countries, provided that the companies are registered for VAT. In this case you must acquire the VAT registration number of the company.*
Translated from [10, p. 8].

4. *If the net VAT payable calculated to the nearest penny, is less than £1, no payment need be made.* From [3, p. 100].

Figure 1: Examples on legal rules.

Amenable to static analysis It is desirable that F supports decidable reasoning and can be subjected to static analysis such as dead rule detection and inconsistency checks.

Have tractable reasoning Another crucial feature is the existence of a well-maintained reasoner with the ability to handle large models (the EU Directive on VAT is 118 pages) and simultaneously process many transactions (queries) at runtime—in short reasoning must be tractable. We note that expressive power and tractability of reasoning are opposite forces.

We believe that the need for expressive power is best analysed by evaluating existing formalisms and tools. We report on such an evaluation below.

4 Evaluation of Frameworks

To evaluate a framework, we consider a list of legal definitions and rules which we believe are representative for the legal domain. In order to give the reader a hint about the flavour of legal rules we present a few in Figure 1.

In a framework we designed a model with the explicit purpose of testing the model on a suite of basic, common tasks associated with VAT handling in ERP systems: To calculate the selling price without VAT, the VAT component and the VAT percentage of a given supply in a line-by-line fashion similar to Microsoft Dynamics NAV. A line corresponds to a quantity of one particular supply. Thus if a quote or invoice contains multiple supplies it must have several lines. The reason why not only one of the VAT component and the VAT percentage suffices is that the VAT component might be zero even though the selling price without VAT and the VAT percentage are not. An example is rule 4 in Figure 1.

5 Modeling in Two Logic-based Tools

In this section we report on our experiences using the two logic-based frameworks: OWL (description logic) [1] and Configit Product Modeling Language (CPML) [2] to model select legal rules *with particular emphasis on scoring the two frameworks against the requirements identified in Section 3.*

	Tool support	Isomorphism	Legal doctrine	Transferability	Exp. Power	Static anal.	Tractable reas.
OWL	Adeq.	Adeq.	Low	High	Low	Adeq.	Adeq.
CPML	High	Adeq.	Adeq.	Adeq.	Adeq.	High	High

Figure 2: Suitability of OWL and CPML for VAT modeling. Each tool could potentially score "None", "Low", "Adequate", or "High" in each category, reflecting our—subjective—assesment of how *fast* and *easy* the task of creating a model satisfying the requirements and being able to handle a suite of tasks associated with computing VAT rates was.

We have chosen OWL and CPML because they seemed prime candidates due to their (on the surface) satisfying most of the requirements of the last section, especially on the side of tractable reasoning and tool support. In case of OWL we have used the Protégé editor [4] and the Racer Pro reasoner [8] while CPML comes with its own complete tool suite. We can only give a brief account of our modeling in this paper—full models including all code employed are available for download at `http://www.diku.dk/~mortenib/2008/lpar/models.zip`.

Figure 5 shows the result of our experiences with the tools as regards the requirements laid out in Section 3.

5.1 OWL

OWL, short for Web Ontology Language, is an ontology language designed to be compatible with the World Wide Web and the Semantic Web[1, 7, 6]. The most salient abstraction in OWL is a concept axiom called a *class*. Each class has a list of necessary conditions and zero or more equivalent lists of necessary and sufficient conditions governing membership of the class.

Structure of the model: We created our OWL model in an iterative way starting from a small framework of classes central to VAT Modeling of the rules then progressed by first extending the model with needed classes and data type properties etc. and, for each rule, adding a class with necessary and sufficient conditions corresponding to the rule in question.

Evaluation and discussion: The primitives in OWL and the tool support we had did not fit well with modeling of legal rules. We often found that the addition of a new rule would trigger a global change in the model due to ill-preservation of the structure in the legal source. However, OWL provided good support in the modeling of definitions and their interrelationships. A salient problem with OWL was the lack of support for inequalities on integers which prevented us from modeling rules 1 and 4 in Figure 1. This taken together with the (state of the art) reasoners ill support for individuals and the lack of support for hasValue ($\ni$) and quantifier restrictions ($\exists$) on datatype properties makes us conclude that reasoning about the legal domain is not tractable using OWL.

5.2 CPML

The *Configit Product Modeller* uses a language henceforth referred to as *CPML* [2]. The Product Modeller is a graphical tool for building *CPML models* consisting of variables with finite domains together with rules, relations and functions. Rules and relations specify dependencies between variables in propositional logic.

The purpose of a model is to describe all legal assignments of values to variables and to help a (human) user to arrive at a legal combination (a configuration). In general, this problem is NP-complete, and in order to give good runtime guarantees Configit has isolated the NP-hardness of the problem in an offline compilation step.

Structure of the VAT model: On the top level our model consist of two groups: `Supply` containing

variables (and sub-groups) describing a supply and `VATrules` containing a sub-group for each of the rules we have modeled.

Modeling precedence: All rules in Figure 1 describe situations where the VAT component is zero even though other rules conclude differently, due to Lex Specialis. Our solution was to create a variable named *LHS* for each of the rules, which is true exactly when the corresponding rule applies, and to overwrite the VAT component with the content of a rule specific variable named *RHS*.

Evaluation and discussion: CPML comes close to realising the isomorphism principle but cannot separate main rules from exceptions. Furthermore, the Product Modeller is easy to work with but does not easily support the legal domain. However, the expressive power of CPML is sufficient and reasoning is tractable provided that the model compiles (which is NP complete but not a problem in practise).

6 Conclusion and Future Work

We have proposed using logical modeling of VAT legislation, assessed the benefits of this approach, and have identified a number of requirements for such modeling. To support our proposition we have modeled salient parts of the Danish VAT law in two different logic-based tools: Web Ontology Language (OWL) and Configit Product Modeling Language (CPML), scoring them against the requirements identified.

The most salient future work will be full modeling of the EU VAT directive and the VAT legislation of several large European nations, preferably using a suite of different frameworks to gauge their suitability for this task. Furthermore, integration into an industrial-strength ERP system is needed to validate our claim that all localisation can be made into configuration.

References

[1] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. Owl web ontology language reference.

[2] Configit Software, Kristianiagade 7, DK-2100 Copenhagen Ø. *Configit Product Modeler User's Guide*, June 2008.

[3] H. R. . Customs. Notice 700: The VAT guide–all sections. HMRC, April 2002.

[4] V. Haarslev and R. Möller, editors. *Proceedings of the 2004 International Workshop on Description Logics*, volume 104. CEUR-WS, 2004.

[5] E. Heinäluoma. Council Directive 2006/112/EC of 28 November 2006 on the common system of value added tax. *Official Journal of the European Union (English edition)*, Volume 49 11(L 347), December 2006.

[6] I. Horrocks and P. F. Patel-Schneider. Reducing owl entailment to description logic satisfiability. *Lecture Notes in Computer Science*, 2870/2003:17–29, September 2003.

[7] A. R. R. S. M. Horridge, H. Knublauch and C. Wroe. A practical guide to building owl ontologies using the protégé-owl plugin and co-ode tools edition 1.0. http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf, 2004.

[8] Y. Sure and Ó. Corcho, editors. *EON2003, Evaluation of Ontology-based Tools, Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools at ISWC 2003*, volume 87 of *CEUR Workshop Proceedings*. CEUR-WS, 2003.

[9] F. C. T.J.M. Bench-Capon. Isomorphism and legal knowledge based systems. *Artificial Intelligence and Law*, 1(1):65–86, March 1992.

[10] ToldSkat. Moms - fakturering, regnskab mv., January 2005.

[11] W. van der Aalst and M. Pesic. Decserflow: Towards a truly declarative service flow language. In *The Role of Business Processes in Service Oriented Architectures*, number 06291 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006. `http://drops.dagstuhl.de/opus/volltexte/2006/829`.