Alexander Boer et al., 'A Content Management System based on an Event-based Model of Version Management Information in Legislation' in T. Gordon (ed.), *Legal Knowledge and Information Systems. Jurix 2004: The Seventeenth Annual Conference*. Amsterdam: IOS Press, 2004, pp. 19-28.

A Content Management System based on an Event-based Model of Version Management Information in Legislation

Alexander Boer Radboud Winkels Tom van Engers
Emile de Maat
Leibniz Center for Law, University of Amsterdam, The Netherlands

Abstract. Understanding electronic legislation requires a comprehension of its lifecycle and the events and corresponding transformations that make up this lifecycle. A metadata description attached to the legislative resource represents a time point in this lifecycle. In this paper we advocate an event-based model of version management for legislation. We are currently developing a Content Management System (CMS) based on this model. This paper discusses the event-based model, sketches the infrastructure of the CMS, and relates it to timestamping in METALex documents.

1 Introduction

Governments and legal publishers nowadays usually make legislation available in some special purpose XML format or XHTML annotated with metadata describing what version of which legislation it is. These electronic documents are taken from largely autonomous and heterogeneously organized repositories of legislation, and versioning metadata is closely tied to the organization of the repository it came from. In this paper we investigate how interoperability and integration of versioning metadata from different content providers can be achieved. Understanding the relationship among multiple metadata descriptions begins by understanding the resources they purport to describe. Understanding the resource requires a comprehension of its lifecycle and the events and corresponding transformations of the resource that make up this lifecycle. An electronic document and its metadata description represents a time point in this lifecycle.

The values of metadata attributes may change over the lifetime of a legal document, even if the document itself does not. Also, metadata is often not extra: it mostly concerns information already contained in the document itself, or in another document that refers to it. In earlier publications about the METALex XML schema for legislation[1] we have explained the use of the World Wide Web Consortium's Resource Description Framework[2] (RDF) in a separate information store to encode information about METALex XML documents with this observation about document metadata (e.g. [2, 1]).

RDF can be used to store a reified description of the events through which the resource with the attached metadata description comes about. This approach has recently been formalized as the ABC model of metadata interoperability by Lagoze et al. in [7], and there are instances in AI & Law where modification of legislation has been treated as an independent event (e.g. [4]) instead of relying on static metadata categorizations of the state of the document. An simple example of such an event that applies to all documents is the 'translation'

[1] http://www.metalex.nl
[2] http://www.w3.org/RDF/

event that relates two documents that purport to contain the same information written in two different languages, of which only one is the authentic one.

In this paper we describe a simple event-based model for version management information in legislation metadata. We are currently performing a feasability study for the Dutch Tax and Customs Administration (DTCA) relating to the following two questions:

- Is it possible to store the information the DTCA collects about legal sources over time in such a way that content bought from content providers can be automatically inserted into the existing internal infrastructure for software applications, without requiring manual linking each time they deliver an update?
- To what extent is it possible to extract such information – in particular relating to version management and relations between different types of sources such as legislation, case law, and commentaries – automatically using a parser that directly parses the text of the legal sources?

The DTCA is usually directly involved in drafting tax legislation, and it may at first seem surprising that it is interested in buying tax legislation from legal publishers. It is not as such. A number of Dutch legal publishers sell electronic products that contain legislation, related case law, and commentaries from experts relating to these legal sources. The DTCA is primarily interested in making commentaries from the different sources available in internal applications for its civil servants, and is confronted with the problem that these products are all organized in a different way and frequently updated, and sometimes even reorganized, by the publishers. Versioning, structuring, and naming practices are different. This makes it difficult to establish the exact identity of the legal sources contained in a product without continuous human intervention. The result is that the DTCA currently invests a lot of energy in making all these commentaries available in the right places in internal search engines and applications. The DTCA ordered this feasability study to determine whether it is possible to build a Content Management System (CMS) that can make this process more efficient.

This "identity" problem is much harder for the consumer of legal sources from different producers, than for the producer itself. From the point of view of the producer of documents solving the version management problem comes down to centralizing access to the source. The readers of this paper may use such systems: The Concurrent Versions System (CVS) for programmers for instance. The versioning approach described in this paper is an essential ingredient for a versioning system that aggregates legal information from different producers.

To demonstrate feasability we are building a prototype CMS for these purposes, based on the mechanisms we developed for METALex XML, and a prototype parser that recognizes standard model sentences for auxiliary provisions dealing with version management, the identity of legislation, and delegation of legislative competence in some legislative domain (somewhat similar to [3]). This paper discusses the event-based model, sketches the infrastructure of the CMS, and relates it to (static) timestamping in METALex XML documents.

1.1 Types of Timestamps in Legislation

A shared feature of legislation in all jurisdictions known to us is that all use discrete 'timepoints' in the form of a date. Roughly, one can distinguish three kinds of timestamps used in legislation:

Version management These timestamps define the validity of the document for reference, and often the validity for application in the (narrow) sense that the document can be applied by a competent decisionmaker in the time-interval in which it is active.

Legislative Drafting These timestamps relate to the procedures that have to be followed by the legislator. In addition to spurious timestamps for certain events (e.g. signing), there are usually rules for minimum time intervals that must have elapsed etc.

Application to cases These timestamps try to define what objects in the outside world legislation refers to. Rules relating to traffic for instance usually deal with immediate events that take negligible time, but other legislation often deals with persistent 'objects' like cars, mortgage loans, and pension arrangements, or delayed payoffs of choices (for instance financial products exempt from certain kinds of taxation). The latter type of legislation has to deal with transitory regimes to minimize the damage of changes in legislation to society. Notions like 'retroactive application' generally belong in this category.

These timestamps show up prominently in a metadata or XML standard for legislation. Any detailed discussion of timestamping practices beyond the simple rules exemplified by METALex XML is unfortunately jurisdiction-specific. In this paper we will therefore only use examples from the Netherlands Guidelines for legislative drafting (AR), but we know that similar practices occur in other jurisdictions.

Wellknown design patterns like *retroactive application* (e.g. AR, aanwijzing 167), *immediate application* (AR, aanwijzing 166), *delayed application* (to allow evasive action; AR, aanwijzing 169), and *'honouring' application* (where a certain existing state of affairs is honoured, while the rules apply only to 'new' cases; AR, aanwijzing 169) apply to time attributes of persistent objects in the regulated world, and not the lifecycle of legislation itself. These patterns belong in the third category and will not be explained in this paper.

2 Legislative Events and the Lifecycle of Legislation

Legislative events are those events that can conceivably have a version of legislation as output. The lifecycle of legislation usually consists of four phases, or *states* the legislation can be in:

Fixed At some point in time a design of legislation becomes an official proposal and is at this point fixed in the sense that it cannot any longer be modified by legislative drafters in the normal way (by opening the document in an editor, changing it, and saving it). Usually this is some "sign" or "pass" event. Some kind of legislation is required to change the text of the proposal. In the Netherlands this point coincides with the date of signing of legislation by the monarch. In the Netherlands, at this point auxiliary provisions that may be contained by it that set the date of publication, date of enactment, official name of the legislation, official acronym of the legislation, and delegate legislative competence, are enacted and become valid law.

Knowable At some point in time the proposed legislation is made publicly known. This is the date of publication. This date is usually prescribed by law. Legislation may set a date, or require publication relative to some other event (e.g. it is published when some other, closely related law, is published). This is an alternative date at which any auxiliary provisions mentioned before, except for the one prescribing the date at which it is published which must have been set before, may enter into force. National legislation in the Netherlands is published in the Staatskrant (State Gazette).

Active At this point in time, the date of enactment, the legislation becomes valid law and may be applied. This date is usually prescribed by law. Legislation may set a date, or require enactment relative to some other event (e.g. it is enacted when some other, closely related law, is enacted, or after some fixed time interval after the date of publication). By default it may be applied to events happening after this point in time, and states of

affairs existing at this point in time. There are models for retroactive application and delayed application, but these deal with when events in the regulated domain happen, and not the date at which the legislation may be applied. There may be legal contraints on the minimum time interval elapsed between the date at which legislation was fixed or published, and the date at which it is enacted. In the Netherlands several such constraints exist, depending on the type of legislation.

Repealed At some point in time the proposed legislation is repealed. This date is again usually announced by law. Legislation may set a date, or require cancellation relative to some other event (e.g. it is repealed when some other legislation is enacted, or after some fixed time interval after the date of publication in the case of temporary provisions for a transitory regime). If legislation is repealed it does not 'disappear' for document management purposes: it can still be referenced by its official name and acronym. Temporary legislation is subject to a number of common legislative errors in temporal reasoning, and may therefore be subject to special contraints. In the Netherlands it is for instance a requirement to recount the text of legislation in the modifying legislation after a temporary modification is repealed to prevent misunderstandings relating to the order in which modifications of a text are applied.

Some jurisdictions, in particular lower legislators, may not distinguish between the 'Fixed' and 'Knowable' states, and this distinction is therefore not part of METALex XML. Events have input and/or output, and if they are actions they have an actor in a certain role (e.g. legislator) and optionally instruments (e.g. a legislative competence). There are five types of event that cause a transition that brings a new version of (a part of) a legislative text into being:

Fix This event is initiated by a legislator, using as instrument (a part of) legislation attributing legislative competence, and produces fixed legislation as output.

Publish This event is initiated by the same legislator, and uses a publication channel and optionally legislation requiring the publication as instrument. Input is fixed legislation and output is published legislation.

Enact This event is initiated by a legislator, using as instrument (a part of) legislation requiring enactment of the legislation. Input is published legislation and output is enacted legislation.

Repeal This event is initiated by a legislator, using as instrument (a part of) legislation requiring cancellation of the legislation. Input is enacted legislation and output is repealed legislation.

Modify This event is initiated by a legislator, using as instrument (a part of) legislation requiring modification of the legislation by replacing its text with text quoted in the modifying, or in rare case external, legislation. Usually (parts of) articles are modified, but larger divisions may also be modified by inserting or removing an article. Input is legislation and output is amended legislation.

A novel use of this event-based framework is the use of legislation in an instrumental role. All relevant dates are in fact attached to the event (and if there is a corresponding source that contains the specific date, it is in *external* legislation in the instrumental role, and not the legislation to which the date is usually attached as metadata). To decide for instance when a specific version of (a part of) legislation comes into being the following extremely simple procedure can be applied:

1. Find all modify and enact events that have the piece of legislation as output, that have a part of it as output, or that have legislation as output of which it is a part.
2. Retrieve the list of known dates at which these events happened. Events that happen at an unknown date are excluded.
3. Pick the last date: this is the date the version came into existence.

This method produces 'virtual' consolidated versions on a date of interest that correspond with the intuitions users have of when legislation has changed, and does not produce any more versions than there actually are. If you select the whole law, you get a version for each time even one article of it changed. If you select a part of it, you only get the different text versions that exist of that part. The annotated 'consolidated' versions produced by publishers usually correspond with the number of versions of the legislation as a whole. In addition, publishers may publish versions for interesting dates like the 1st of january or the opening date of the academic year (for educational products).

Another concrete benefit of this approach is that it signals 'gaps' in the consolidated version history in the material provided by the content providers, because the modification event is known but there is no known document in the CMS that corresponds with its output.

Note that 'modify' events are very complex to handle. Note for instance that the modifying provision may also be changed. It is often not easy to find out *when* a modification provision goes into effect (regulated in the Netherlands by AR, aanwijzing 171-173, 175-177, 183), and often a tie-breaking rule is required if multiple modifications have to be applied on the same text on the same date. Modifications are often 'stacked' on the same date to minimize the number of different versions of legislation (which may in the worst case have been active for one day). In the Netherlands the tiebreaking rule is AR, aanwijzing 173A.

3 Design of the CMS

In earlier publications about the METALex XML schema we have explained the use of RDF in a separate information store to encode information about METALex documents (e.g. [2, 1]). This method is applied to content in the DTCA CMS. METALex standardizes structure and designation of identity in legislation. The XML ID attribute can be attached to elements that represent document structure and the METALex XML structure can be translated to RDF conforming to the METALex RDF Schema. Logical constraints on the RDF graph, within a document and between documents, are defined with the Web Ontology Language (OWL[3]) RDF schema (e.g. [6]).

RDF makes the document structure graph explicit and de-couples the identity of elements from the documents in which they are serialized, positioning the element only in a namespace – which may or may not correspond to a document. If a document element is described with RDF statements – triples of a *subject*, *predicate*, and *object* – it can be both subject and object of statements regardless of what document it is serialized in. Figure 1 shows a description of the repeal of some legislation (an article, or a complete act for instance). Traditionally, in a document centered XML format like METALex XML, the date of repeal would be attached to the repealed document as metadata. In the store it is instead attached to a repeal event. Note that the actual date is usually going to be in another piece of repealing legislation, which is the input to the repeal event. In some cases the date may not explicitly occur anywhere: the repealing legislation may for instance state that the legislation will be repealed if some other event occurs.

Content for the DTCA CMS is not usually delivered in METALex XML by the publishers, and it is not translated to it. Instead the document is parsed to discover its identity and

[3] http://www.w3.org/2004/OWL

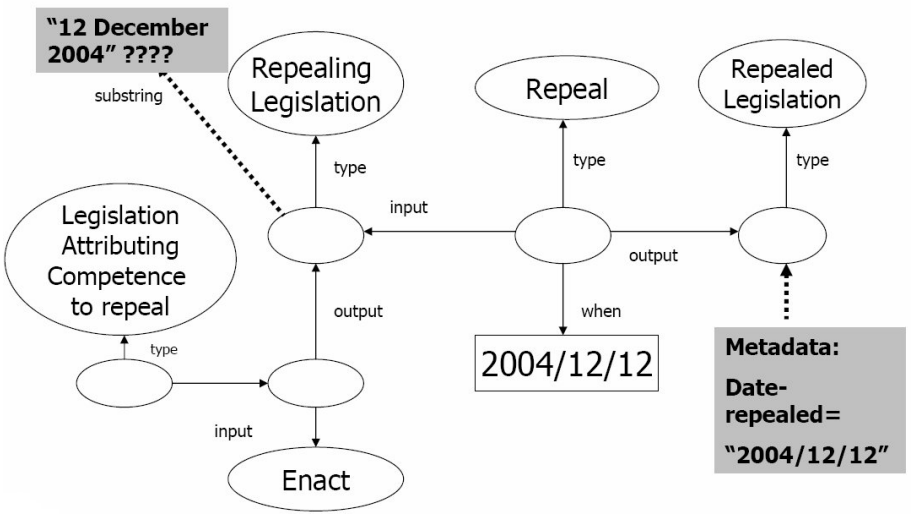

Figure 1: Repeal of a Law in *subject*, *predicate*, and *object* triples. The labels of the edges are the subjects, and squares indicate XML Schema datatypes instead of resources.

structure which is then stored in the CMS in RDF form directly, with a relation to the corresponding content URLs. Each structural element of a regulation may be linked to as many as 6 different serializations in HTML or XML from different content providers. Useful metadata is also extracted in this phase. It is parsed again to try to:

1. classify the text in each document element as definitions, enactment or repeal provisions, deeming provisions, transitional provisions, evaluation provisions etc,
2. discover direct and indirect relations to other legislation, and
3. to classify the role of the referred legislation (attributor of competence, defining, constraining, etc.).

Key functions of the CMS are organizing document display relative to a point in time and the role of a user, relating jurisprudence and commentaries to legislation, and relating specific regulations to the legislation that attributed the legislative competence those regulations are based on.

See figure 2 for an overview of the infrastructure. The RDF information is stored in a Jena-based[4] RDF store. Jena is a wellknown Java library for processing RDF. The CMS is a J2EE application that is approached with HTTP GET and HTTP POST requests over the Internet. In front of the CMS, Cocoon[5] is used to define a logical 'sitemap' in order to offer the functions of the CMS through predictable static URL's designating identity. Cocoon is also used to define user interface forms for HTTP POST functions and to apply XSL sheets to filter and translate replies in XML, resulting in suitable HTML pages if the system is accessed directly through the Internet browser instead of an XML-aware client application. The CMS will usually be accessed by other (server or client-side) applications.

Attached to Jena is a remote OWL reasoner module based on the description classifier RACER approached through HTTP POST commands containing DIG XML[6]. The CMS architecture is separated in three layers and the HTTP interfaces between the layers meet the requirement of *idempotency* – making a request for a particular URL once or multiple times is no different. Each layer can be separately installed on a machine, and easily scaled up to multiple machines with a simple load balancing proxy.

[4]See http://jena.sourceforge.net

[5]See http:// cocoon.apache.org

[6]See http://jena.sourceforge.net/how-to/dig-reasoner.html

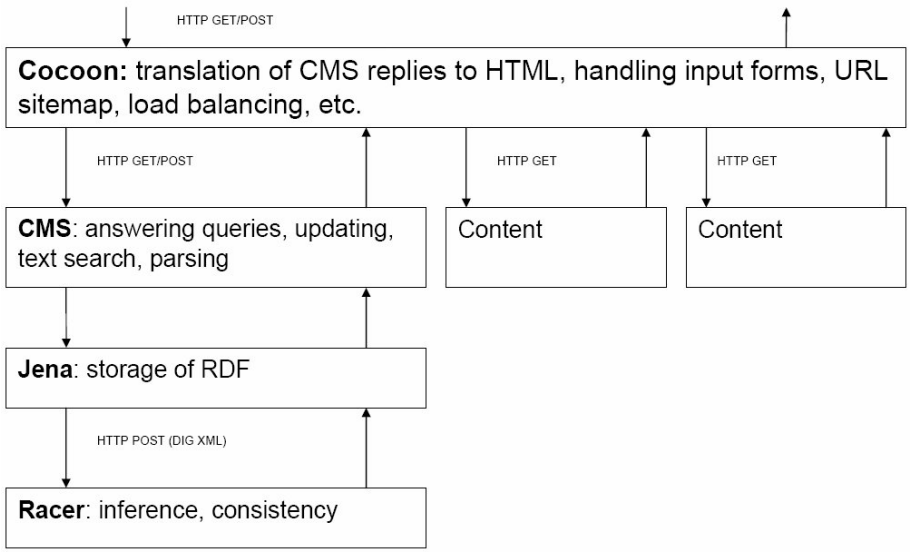

Figure 2: The layers of the CMS infrastructure.

The RACER description classifier (see [5]) guards the consistency of the CMS with the constraints defined in OWL ontologies, classifies the objects in the CMS, and answer queries. There are three kinds of OWL reasoners considered to be "correct": the OWL Lite, DL (for description logic), and Full reasoner. Practical Full reasoners do not exist (because Full is known to be undecidable, and that is an undesirable feature of a Semantic Web technology). The RACER description classifier is a modal logic theorem prover for a modal logic with a number of extensions. The logic is known in the description logic community as a $SHIQ$ (cf. [6]) or $ALCQHIR+(D)-$ (pronounce ALC-choir-D-minus) logic, i.e. standard ALC (a modal logic with labels – Role names – on modal operators) plus qualified number restrictions, role hierarchies, inverse roles, transitive roles, and restricted concrete domains. RACER is considered a full theorem prover for OWL-DL. It is based on the largest subset of interesting description logic comstructs currently considered decidable in the Description Logic community ([5]). The description classifier contains ontologies for the following subjects:

Mereology Parts and wholes and the transitive nature of the relation between these. This theory is required by a number of other theories.

Action and Thematic Roles Actions, and related roles like the point in time the action happens, the actor, instruments, (re)sources, and the product or patient of the action.

Documents and Legislation Document identity, structure, and internal ordering of its parts. Includes classification of legislation and legislators by type and level. This theory corresponds roughly with METALex XML, with some minor extensions and generalizations.

Legislative action Signing, publishing, enacting, repealing, and modifying legislation, and the necessary products, sources, and legislative competence as instrument.

Legal persons and concept-definitions Legislation addresses natural and legal persons by a taxonomy of roles (e.g. taxpayer), and defines certain concepts. These concepts and roles play an essential intermediary role in connecting jurisprudence to legislation.

Temporal validity Relates legislative action and its requirements to the resulting documents and establishes validity of legislation, and jurisprudence dependent on it, at a certain time interval.

The following is an example – in the more readable Knowledge Representation System

Syntax (KRSS), and translated to English – of the kind of OWL document management axioms one may encounter in the RACER description classifier:

```
(equivalent part-of-active-legislation (and legislation (some part-of-legislation
(and knowable-legislation (or enactment-legislation publication-provision
citationtitle-provision (or (exactly 1 (inv product) enact) (some (inv product)
(and enact (some enactment-date integer)))))))))
```

4 Timestamping Document Elements in METALex XML

METALex XML is treated differently than METALex RDF in a repository. To keep track of versions METALex XML provides a number of attributes for every structural XML element in the document that can be identified, selected, and thus changed. It is very important to realize that the requirements for timestamping a document that is serialized are slightly different from the requirements for the CMS, because the XML serializations are intended for exchange of electronic documents. There are two notable differences:

- Because the document lacks a context, all elements of the document must be timestamped and not just the root element of the serialization. The timestamps are thus 'virtual', and many more timestamps are required for METALex XML than for METALex RDF in the CMS.
- The serialized document represents a belief at a certain time point. Consider the example of serializing a document now that represents a piece of legislation as it is going to look like on the first of januari next year: because the document may be changed in between the valid law on the first of januari may be different from the one serialized now. The legislative competence it is based on may also in the meantime be retracted. The CMS would be aware of that if it is sufficiently complete, but the serialization still exists, with faulty information in it.

METALex XML distinguishes 6 time-related attributes: *date-publication, date-enacted, date-repealed, date-effective, date-version, and ref-date*. The *date-version* attribute represents the date the correctness of the content and other date attributes of the METALex element was last verified. An element is considered verified for a certain date if it is certain that the content of the element and the dates set in its attributes and attributes of child elements (inasfar as date attributes are set) are a correct representation at that date for the regulation or decision encoded. As a rule contained texts from the past relative to the date-version of the XML serialization are correct.

A document can only be verified correctly against the whole collection of legislation that (however indirectly) refers to it. If a document element refers to another element, it is important to verify whether the text in the element repeals or enacts the other element for instance. But a classification based on the text of such a provision alone cannot be taken for granted: the legislator obtains the competence to legislate from a specific assignment by law, or from a delegation or mandate decision. Legislation usually becomes inactive if this basis becomes inactive. This rule is at least true for regulations from public bodies in the Netherlands (AR, aanwijzingen 243-245). Small changes in a law may invalidate dozens of pages of lower regulations at once.

A reference to (a part of) a regulation is also optionally timestamped with the *ref-date* attribute. If a *ref-date* attribute is present, then the reference only refers to the version of the document at *ref-date* – which is usually equal to the date the referring document was published. If the *ref-date* is not present, then the reference at any time refers to the version at that time. Dutch legislation for legislative drafting (AR, aanwijzing 92) describes *static* and *dynamic* references based on the same principle.

The *date-effective* attribute can be set to specify that a new provision is applied to certain past events in cases pending at, or brought after, the time of the regulation's enactment if no real modeling of the content of the legislation takes place. As stated in section 1.1, it is not intended to mean that a court can apply the regulation before it is published or enacted. Obviously, ex post facto laws are regarded as subversive to justice. As David Hume put it (discussing the trial of Strafford in 1641[7]): *"Better to live under no law at all, and conform ourselves the best we can, to the arbitrary will of a master, than fancy we have a law on which we can rely, and find at last, that this law shall [..] try us by maxims unheard of till the very moment of prosecution"*.

5 Discussion

A lesson we learned when designing METALex XML is that you are often left with almost trivial schemes when you exclude everything that is jurisdiction-specific. Content management systems that can deal with the 'legal' update rules will necessarily be jurisdiction-specific, unless they manage to distinguish between a sufficiently general vocabulary and separate sets of rules of a specific jurisdiction. We hope that others will apply the schema to new legislative documents and report problems with our jurisdiction-neutral vocabulary to the METALex discussion mailing list. We already noted that *date-effective* is a severely limited solution, but our experience is that legislators do have different solutions for that (and are not even consistent across time and domains).

A lesson we learned while juggling with dates is that all dates are relative to the date today, which cannot be left out of the equation if one of the involved dates is in *the future*. In principle it is not possible to produce future versions of legislation, and a serialization of legislation that does not make clear when it was serialized is almost useless without contextual knowledge. It is surprising how many publishers forget (or 'forget' maybe) to include the date of serialization in products. This is also the context where one gets into trouble with applying modifications: there is no reason to assume that applying modifying provisions that are not yet enacted will result in an unambiguous text for the future today. This is also one of the reasons why legal publishers occasionally confuse their customers by including consolidated versions of legislation in a collection which, judged with the benefit of hindsight, never become law.

There are clear rules for national legislation in the Netherlands, solving even the 'tie-breaking' problem of applying modifications, but this is not always the case unfortunately. Guidelines for legislative drafting of the EU, for instance, are underspecified. Most municipal governments in the Netherlands do not even keep track of previous versions of regulations and attributions of legislative authority at all. The best source for historical versions of municipal regulations in the Netherlands is probably the database of the Council of State, which is the court for administrative conflicts.

Time-related problems with commentaries and case law have not been discussed in this paper, even though classifying the commentaries is central to the study. Classifying which version of legislation commentaries (or doctrine) by legal experts refer to is even more problematic if the commentary refers to future legislation, because it may comment on invalid future versions of legislation. At the same time one does not want to throw away old commentary before one is sure that it is stale. The cause of the problem is that many commentaries are dated only by the date of last modification of the electronic resource and refer in the text to such things as 'the new income tax law'. If the publisher decides it contains faulty information, it will repair the fault and timestamp and redistribute the commentary again, making it impossible to discover from the text alone what version of legislation it is about and –

[7] Letters of David Hume to William Strahan, ed. G. Birkbeck Hill (Oxford: Clarendon Press, 1888).

even worse – still potentially containing other faults. Because the commentary has now been moved into the future, it now appears to apply to another version of the involved legislation.

Case law in itself is trivial as far as version management is concerned. There is only one relevant version of a court decision. The two issues with case law are deciding whether the *same case* has been annulled by a higher court (which is doable in a jurisdiction where case law is not considered to be semi-legislation), and deciding whether the legal grounds for the decision have disappeared because legislation has been repealed.

The relation between an original court decision and a higher one which disagrees on the same case is obviously interesting content for the CMS, and not very hard to find in the available sources. No clearcut interpretation is possible of how annulling a decision affects *similar* cases if the court decision itself is not written with the intent to state a new rule[8].

To solve the problem of deciding whether case law has become irrelevant it is important to find out whether references to repealed legislation in the decision refer to norms or import concepts, because if they refer to concepts then the concept may have been copied from the old place to a new place and judgments relating to the interpretation of the term remain valid. A more interesting issue is whether one can automatically deduce that a reversal of a decision making trend is occurring in case law. This obviously would require much more information about the content of a verdict than we try to extract for the CMS.

A provisional conclusion is that it is feasible to build a CMS according to requirements for legislation and case law, but that there are limits to what can be done with commentaries unless publishers add better time-related information.

Acknowledgements

The METALex XML initiative was started during the E-POWER project. E-POWER was partially funded by the EC as IST Project 2000-28125. We are working on a feasability study for a knowledge management infrastructure including document and version management functionality based on METALex for the Dutch Tax and Customs Administration. This paper was written in the context of that project. Some of the problematic scenario's with applying future modifications to legislation were already pointed out to one of the authors of this paper on the Jurix conference in 2002 by Monica Palmirani.

References

[1] A. Boer, R. Hoekstra, R. Winkels, and T. van Engers. ^{META}lex: Jurisdiction and Language. In Monica Palmirani, Tom van Engers, and Maria A. Wimmer, editors, *Proceedings of the E-Government Workshop in conjunction with JURIX 2003*, pages 54–66. Universitätsverlag Rudolf Trauner, December 2003.

[2] A. Boer, R. Hoekstra, R. Winkels, T. van Engers, and F. Willaert. ^{META}lex: Legislation in XML. In T. Bench-Capon, Aspassia Daskalopulu, and R.G.F. Winkels, editors, *Legal Knowledge and Information Systems (Jurix 2002)*, pages 1–10, Amsterdam, 2002. IOS Press.

[3] Andrea Bolioli, Luca Dini, Pietro Mercatali, and Francesco Romano. For the automated mark-up of italian legislative texts in xml. In T. Bench-Capon, Aspassia Daskalopulu, and R.G.F. Winkels, editors, *Legal Knowledge and Information Systems (Jurix 2002)*, pages 21–30, Amsterdam, 2002. IOS Press.

[4] A. Gangemi, D.M. Pisanelli, and G. Steve. A formal ontology framework to represent norm dynamics. In R.G.F. Winkels and R. Hoekstra, editors, *Proceedings of the Second International Workshop on Legal Ontologies (LEGONT)*, Amsterdam, Netherlands, 2001.

[5] Volker Haarslev and Ralf Möller. RACER system description. *Lecture Notes in Computer Science*, 2083:701–??, 2001.

[6] I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From shiq and rdf to owl: The making of a web ontology language, 2003.

[7] Carl Lagoze, Jane Hunter, and Dan Brickley. An event-aware model for metadata interoperability. *Lecture Notes in Computer Science*, 1923:103–??, 2000.

[8] We are referring here to the tautological nature of stating that two cases are the same because they decided the same, and different because they are decided different, or vice versa.