An XML Standard for Legislation in the Netherlands: Representation of Norms in XML

Alexander Boer and Radboud Winkels and Rinke Hoekstra

February 22, 2002

Abstract

This (draft) report is part of an effort to define an XML Standard for Legislation and Public Decisions in the Netherlands. The research is carried out in the context of the ePOWER project and the LEXML initiative for XML standards for the various European jurisdictions. This report describes a mapping from CLIME/ONLINE style norms to DAML+OIL, an ontology specification language for XML/RDF. The document focuses on describing the interpretation of normative assessment in DAML+OIL, not (yet) the specifics of defining the conceptual primitives (rule, norm, case, generic case) in DAML+OIL syntax. Some important repercussions of maintaining a mapping between UML and DAML+OIL are also discussed.

1 Introduction

This (draft) report is part of an effort to define an XML Standard for Legislation and Public Decisions in the Netherlands. The research is carried out in the context of the ePOWER project[1].

The standard intends to provide a generic and easily extensible framework for the XML encoding of the structure and contents of written public decisions in the Netherlands. We roughly distinguish three different viewpoints on how we look at such documents:

Form In the Netherlands a *wet* is 'recognized' by certain required phrases and formulas (*Allen die deze zullen zien of horen lezen, saluut!*). These formal requirements mostly come from the grondwet, Awb, and AR. The formal requirements are influenced mostly by considerations of consistency of language and ease of access for the reader. The formal structure of the document provides a context for the interpretation of the content of the document.

Role Although we may look at the phrases and formulas in a written decision to classify a document as a *wet*, we know that it is not the structure of the document that makes it a *wet* or *gemeentelijke verordening* but the role the document plays in the activities of public bodies, most importantly the activities that produced the document.

[1] IST Project 2000-28125; includes the DTCA (Belastingdienst) and this department

Content We also classify documents depending on what its contents mean: It represents a type of decision or *besluit*. If it is a *beschikking* its meaning is limited to a particular occurrence or case. If it is a *algemeen verbindend voorschrift* its meaning extends to general class of occurrences or cases. If it is a *beleidsregel* it does not classify generic or particular cases as such; It proposes a general standard, a value theory, for judging the quality of decisions.

The form, role and content classifications overlap: An *algemene maatregel van bestuur* is for instance a *koninklijk besluit* that contains norms, while a *koninklijk besluit* can also be limited to a simple appointment decision. We can distinguish the *koninklijk besluit* on formal criteria from other types of documents, but its 'identity' criterium is in the end a role-based citerium.

This report is concerned with the content of documents, in particular with the representation of what norms mean. Norms (or 'deontic' modalities of reasoning) have been subject to research by philosophers, legal theorists, and logicians for millenia. In this report we assume a degree of familiarity with the main deontic concepts (obligations, permissions, prohibitions, rights, power) and the notation of propositional modal logic.

This (draft) report describes a mapping from CLIME/ONLINE style norms to the XML/RDF ontology specification language DAML+OIL. This is a mapping on a logical level, not syntax level.

The document focuses on describing the interpretation of normative assessment in DAML+OIL, not the specifics of defining the conceptual primitives (rule, norm, case, generic case) in DAML+OIL syntax. This mapping will be specified in an XHTML document. To map the original CLIME/ONLINE norms to DAML+OIL we have to solve the problem of the missing existential variables for queries and the more practical problem of verifying DAML+OIL grounded facts (cases).

The technique described in this paper is partially validated with actual examples specified in OILedit and executed in description classifier FaCT. Adequate DAML+OIL classification in FaCT is used as a criterium for a workable standard. Another important consideration is a potential mapping of POWER's UML models to DAML+OIL (and vice versa). Some important differences between UML and DAML+OIL are discussed in the next section.

The difficult aspects in the translation of CLIME/ONLINE norms relate to the concept of a case and generic case, and instance classification of cases and generic cases relative to each other. An intriguing question is whether cases can subsume generic cases or other cases: This is basically the problem of the 'missing relevant facts in the case that can change the normative classification of the case' (and maybe provides insight in the relation between norms and 'actual' court decisions). This paper treats cases differently – as 'leafs' in the classification hierarchy – but it may be possible to extend the argument for generic cases to cases (with a computational penalty).

1.1 Terminology

Considering in particular that this report is written in English, we have included a listing of some of the more salient concepts discussed in this document:

Public Decision Translate as *besluit*. Any written decision of a public body (*bestuursorgaan* or other public body with public power (*bevoegdheid*) to make decisions including the bodies in *artikel 1:1, tweede lid Awb*.

Legislation Translate as *wet in materiele zin*. A type of written public decision that enacts norms. It is based on a public power to legislate.

Norm Translate as *algemeen verbindend voorschrift*. A deontic judgement of a general nature; A formula that classifies certain cases as *allowed* or *disallowed* – regardless of whether it assigns responsibility or allows the possibility of sanctions.

Generic Case The type of thing to which a case description in a norm refers. It may be a static situation (e.g. in traffic), a design (e.g. of a house), an action or transaction (e.g. a sale, murder, delegation of a legislative power, a public decision).

Case A specific occurence of a generic case.

1.2 Preliminaries

DAML+OIL is an extension to $RDFS$ that imposes a much stronger interpretation of RDF graphs – in DAML+OIL one can define fairly precisely what graph structures are valid. This level of preciseness is achieved at a high cost; The RDF graph structures that *can* be defined as valid constitute a small fragment of RDF.

The most complete description classifier available for DAML+OIL is FaCT. FaCT is a description classifier for an ALC or $SHIQ$ Tbox, and consists of a definition and a query language. An ALC Tbox is a set of assertions about concepts of the form $C_1 \sqsubseteq C_2$ (and $C_1 \doteq C_2$ as shorthand for equivalence) where C_1 and C_2 are concept definitions conforming to the syntax $C ::= N \mid \top \mid \bot \mid \neg N \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall R.C \mid \exists R.C$ where N is a concept name with the usual interpretation. The remarkable thing about FaCT is that it can support transitive and functional roles in ALC, and proves very efficient. Roles can be defined with primitive role definitions $N \sqsubset R$ where R is a role comforming to the syntax $N \mid R_1 \sqcap R_2$ and N is a role name with the usual interpretation (including frame identifiers for transitivity, inversions, and functions). DAML+OIL itself is a $SHIQ(d)$ Tbox, or $SHIQ$ and (xsd) datatypes.

Work is underway on an Abox for FaCT. In the meantime there is no way to verify a set of facts (and therefore cases) in FaCT. There is therefore also no provision for variables in the Tbox query language (since ALC is a propositional modal logic K4). The query language basically allows us to verify expressions of the form $C_1 \sqsubseteq C_2$ against the TBox. Contrary to DAML+OIL, FaCT does not support definition of concepts by explicit enumeration of instances and cardinality restrictions on roles (except for the functional restriction). Another consideration to keep in mind is that FaCT cannot retract a definition at runtime, which limits its usefullness to batch verification of ontologies (and knowledge bases).

2 ONLINE and CLIME

In *ONLINE* (developed at this department) cases were represented as grounded instances, and generic cases were in fact queries to be submitted to the description classifier ($LOOM$, which had a rich query language but was seriously incomplete and unpredictable). Since the queries themselves could not be classified (even by $LOOM$, obviously) all norms had to be executed as queries to determine the normative classification of a case; A huge waste of the one main strength of the description classifier: classifying concepts.

The *CLIME*[2] assessment engine was able to classify cases (of the same level of expressiveness as ONLINE) but was backed up by a simple handcrafted concept description language for instance classification with a much lower level of expressiveness than ALC and no support for concept classification. This design does allow for extension with a complete description classifier.

2.1 Querying Legislation

Most information retrieval systems for legal documents only try to capture domain-specific 'concepts' in regulations, assuming that the user wants regulations 'about' those concepts. The CLIME assessment engine is an attempt to capture also the 'normative' meaning of the regulation and apply it to the information in the query.

A norm qualifies a generic case as either *allowed* or *disallowed*. Norms are modelled everywhere: In planning and design systems they usually occur as requirements (that must be met) and constraints (that must not be violated). Norms occur in three main variants in natural language: obligations, permissions, and prohibitions. A fictional example:

1. *Tanks adjacent to the hull are not used to store fuel oil.*
2. *If the ship is fitted with a double hull, tanks adjacent to the hull can be used to store fuel oil.*

Straightforward approaches to modelling norms translate this to $if \ldots then \ldots$ rules that derive a normative predication for some fact from a set of conditions. The problem with such an approach is that it lacks a coherent theory of how norms are composed into coherent normative systems. It can only work reliably if the normative rules constrain only one ideal world. The previous example, however, tells us conflicting things if I have a ship fitted with a double hull and a fuel oil tank adjacent to the hull. Obviously rule 2 must be a normative 'exception' to rule 1. If an IR system would have returned rule 1 first, a naive user might have drawn the wrong conclusion.

In regulations these exceptions are often used for compactness; A so-called qualification model (a full, monotonic paraphrase, cf. [4]) of regulations is often much harder to understand and use. If explicit and intended, the exception relation is often marked in the text in some way, but the general line of reasoning is always the same. In principle the more specific regulation is superior, unless it is of lower order or of equal order but issued earlier (both properties of the document that contains a regulation). These principles impose a priority ordering on norms (not worlds) and go by the names *Lex Specialis*, *Lex Superior*, and *Lex Posterior* respectively in legal theory. The CLIME assessment engine assigns a normative qualification to the 'whole' prime implicant of the generic case in the user's case (the minimal model of the case that entails the generic case; cf. generally [8]). This complication is necessary because we want to determine automatically which norm has a 'more specific' generic case.

For a more precise notion of a valid argument in a normative system, let $\mathcal{R}$ be the set of modelled regulations, $\mathcal{C}$ the resulting set of cases, set $\mathcal{Q} = \{allowed, disallowed, silent\}$, and the relation $\mathcal{N} \subset \mathcal{R} \times \mathcal{C} \times \mathcal{Q}$ the set of norms. Cases are expressed as closed first-order formulas, built inductively using monadic or dyadic predicates $P_1, P_2, \ldots, P_n$, variable letters $x, y, z, \ldots$, and logical connectives $\neg$ and $\wedge$. All variables are existentially quantified and distinct variable letters are assumed disjoint within one formula.

[2] CLIME, Computerized Legal Information Management and Explanation, Esprit Project EP25414; Including this department

A predicate letter denotes a term from the domain ontology. Normalized to a cube (conjunction) of literals a case $\varphi \in \mathcal{C}$ is written as:

$$\varphi = [P_1 x_1, P_2 x_2, \ldots, P_n x_3]$$

A case φ entails another case ς iff any interpretation using the domain ontology that makes φ true also makes ς true. If the user supplies a case ς for assessment, there is a norm $(r, \varphi, \rho) \in \mathcal{N}$, and $\varsigma \models \varphi$, then obviously $(r, \varsigma, \rho) \in \mathcal{N}$. If, for example user case $\varphi = [ship(x), doubleHull(y), fuelOilTank(z), fittedWith(x, y), fittedWith(x, z), adjacent(y, z)]$ and the previous example regulations are identified as r_1 and r_2 then it is possible to infer $(r_1, \varphi, disallowed)$ and $(r_2, \varphi, allowed)$ if the domain ontology is sufficiently complete. Sofar we have established no good reason to distinguish allowed and disallowed from true and false. There is however a source of nonmonotonicity in the normative system that is separate from normal 'terminological' exceptions; A norm may be an exception to another norm if it strictly disaffirms the other norm as defined here:

Definition 1 ***(Strict Disaffirmation)*** *A norm $(r_1, \varphi, \rho_1) \in \mathcal{N}$ strictly disaffirms a norm $(r_2, \varsigma, \rho_2) \in \mathcal{N}$ iff. $\varphi \models \varsigma$ and $\varsigma \not\models \varphi$ and $\rho_1 \neq \rho_2$.*

This relation between norms can be computed offline and stored in an noncircular partial ordering $\mathcal{E} = \{\mathcal{N}_2, \prec\}$. The example norms are thus resolved to $(r_1, \varphi, disallowed) \prec (r_2, \varphi, allowed)$ and φ is allowed if r_1 is not of higher order or issued later and of equal order. The resulting exception graph can be used to guide dialog and it is input to a bigger noncircular partial ordering $\mathcal{P} = \{\mathcal{N}_2, \prec\}$ which is the overall priority ordering on norms used to establish whether a case is allowed or disallowed by $\mathcal{P}$, or whether $\mathcal{P}$ is silent about the case. This priority ordering is considered separate from any other exception mechanism used for terminological knowledge because:

- Normative conflicts that are not resolved by $\mathcal{P}$ do occur regularly and are a normal part of legal reasoning (and a major source of litigation). There is nothing wrong with conflict; It is not a truth-functional 'contradiction'.
- The ordering on norms does not imply an ordering on (ideal) worlds; No decision-theoretic assumptions like preferential independence between generic cases can be made. Violating two norms is for instance not necessarily less ideal than violating one in actual regulations (even if legislative drafters feel that this definitely is a desirable property of well-drafted regulations).

Ideal worlds tell us very little about what to do. If deontic sentences are interpreted as logical sentences describing an ideal world, they cannot be used to express what ought to be done in a sub-ideal world; This is the essence of the so-called Chisholm and good Samaritan paradoxes. It shows that deontic sentences can presuppose a sub-ideal world. Strict belief in one ideal world leads to suspect 'formal' arguments like the argument that the distinction between just and unjust wars (ius ad bello) affects the validity of rules in war (ius in bello) if the war is unjust and the situation therefore sub-ideal. This 'logical' argument has been used to justify nuclear war and some actions of the Allies in world war II.

In reality, one must be able to establish a preference ordering on situations to determine what one ought to do given a choice between worlds. There is unfortunately no reason to expect that a generic recipe for establishing a total preference ordering on ideal worlds exists that faithfully represents legal reasoning. The existence of a total

preference ordering is a desirable formal property of a law and a good criterium for evaluation of its effectiveness and transparency, but it cannot be taken for granted that existant law meets criteria of this sort. In fact, you can be sure it does not. You just cannot prove it.

2.2 Assessing situations

To 'solve' the normative qualification of a case, the case must be complete: all relevant facts must be there. The case presented by a user is however almost necessarily incomplete. It cannot be assumed that all necessary facts are supplied at the right level of detail, because the user cannot determine in advance what the relevant facts are. The question of the user is usually focused on some specific aspect of the rules, and which facts are supplied is based on the presuppositions the user made about what is relevant to the normative qualification of the presented case. If the user for instance frames a question around a description of a *tanker*, instead of a *liquid gas carrier*, he may obtain a misleading normative qualification. The CLIME assessment engine is therefore able to initiate dialog to fill in the missing facts or present the open questions that may still change the normative qualification of the case. In this respect it models the legal advisor who helps the client in determining whether the case in question is allowed through a simple strategy of continually trying to apply norms that change the verdict from allowed to disallowed, or vice versa (cf. generally [13, 3]). The order in which norms are checked is based on the exception graph. The presentation of potential exceptions can be fine-tuned further by determining the prime implicant of the conjunction of all missing facts (to remove logical redundancy in the line of questioning; cf. [12]) and making smart 'frame' assumptions (like assuming that the design and construction of a ship is in compliance with regulations if the query concerns the operation of a ship). Of course smart assumptions are domain- and user-specific, and not generic for legal reasoning.

2.3 The MILE System

MILE is an Internet-based IR tool for regulations in technical domains that was developed in the CLIME project. The demonstrator includes ship classification regulations and Annex I and II of the MARPOL treaty. These documents are available in several translations and isomorphic in the sense that each translation of a rule imposes the same legal qualification on the same situations.

MILE offers two functionalities: *conceptual retrieval* and *normative assessment*. The user does not enter *search topics*, but a description of the actual or hypothetical *case* the user is interested in: a concrete situation involving objects, events, agents, and their attributes. This presumes some mental effort on side of the user, since he has to rephrase more general question into one or more concrete situation descriptions. To facilitate this use of the system, the user can edit previous queries to produce variants of his case. The case representation cannot be handled by familiar user interface controls (a mathematician would resort to drawing pictures), so MILE uses a controlled natural language interface, and a direct manipulation *WYSIWYM* [9] interface that restricts the user to the subset of natural language supported. The natural language generator uses a unification grammar that allows for mixing proper grammar rules and fixed phrases. This controlled natural language is translated to the right fragment of first order logic. The editing interface has also been applied to other domains and other mappings to

logic fragments. Both the input case and the answer can be translated along with the interface between English, French, and Italian.

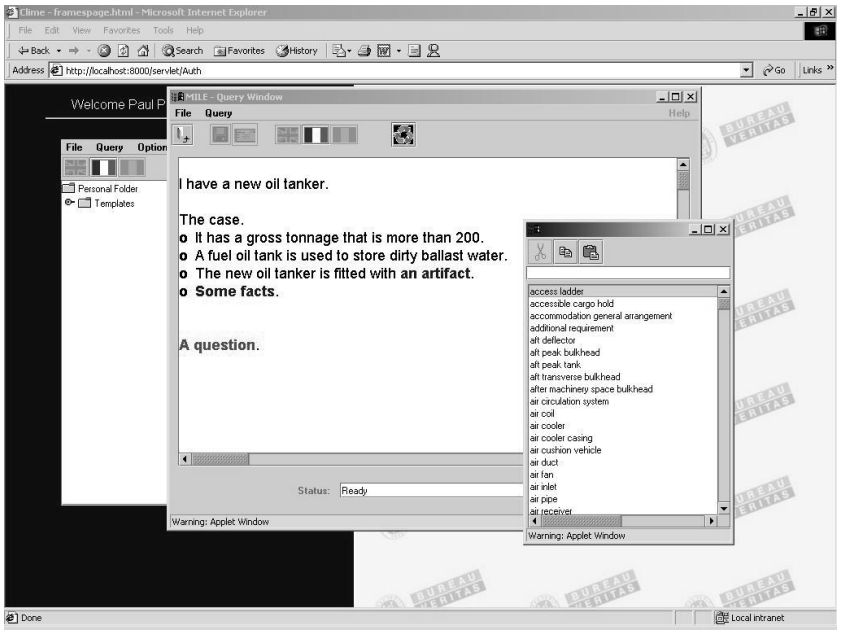

Figure 1: Editing a MILE query.

Figure 1 shows an example of a MILE query applet in an Internet browser. The common controls for copying, cutting, and pasting are used to edit existing objects. In figure 1 a *fuel oil tank* is being pasted into *an artefact*, and *an artefact* will change to *the fuel oil tank* in the feedback text. If appropriate it may change to for instance *it*. The WYSIWYM system may also decide to rephrase other elements as the text is being built.

If you start typing a word in the text field, the term list at the bottom narrows to those terms containing it as a sub string. Typing *tank* will for instance produce i.a. *fuel oil tank*. Composite lexical elements, for instance *A container is used to store a substance*, represent binary predicates, where *a container* and *a substance* can be subsequently specialized or made to refer to an existing individual.

Simple expansion of the case with terminological inferences results in the conceptual retrieval function of MILE (evaluated in [14]). The presentation of assessment models legal argumentation and advice as introduced in the previous section. Relevant missing information in the case is presented as a listing of possible exceptions to the argument constructed by MILE. Figure 2 shows how a normative assessment is presented by the MILE WYSIWYM language generator. The 'dominant' normative qualification is presented first, together with a paraphrase of the way in which the user's case matched the regulation that was dominant in the priority ordering. Other relevant rules whose relation to the user's case could not be automatically established are also presented – including the regulations that match with the user's case but are defeated by the dominant case.

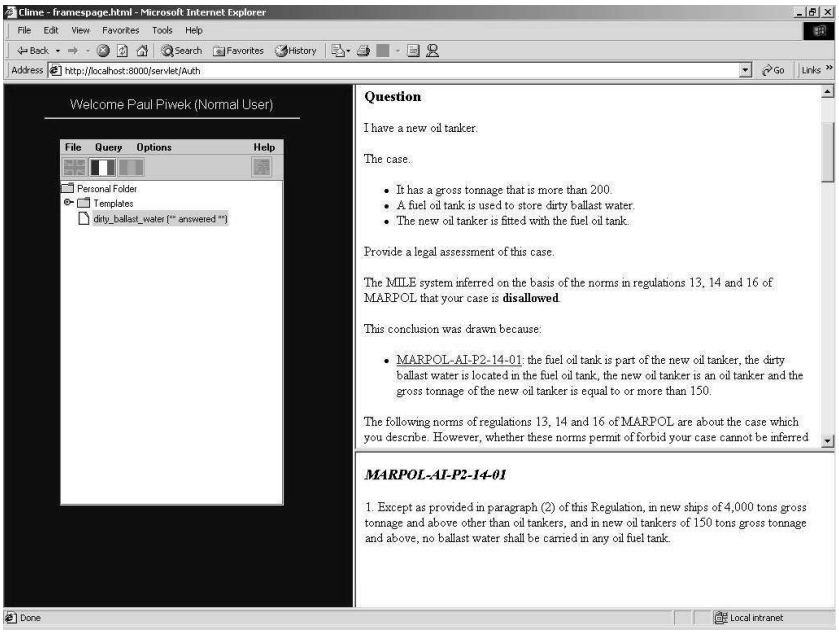

Figure 2: Normative assessment of a case.

2.4 Knowledge acquisition & tools

In CLIME, knowledge acquisition was conducted in an *incremental* fashion; norms are modelled using the terms and relations of a fully developed domain ontology. This approach not only enables the engineer to focus on the specific problems of the type of knowledge that is being represented, but also allows for rapid usability of the KB for less knowledge-intensive applications such as CR.

The ship classification domain ontology was built without reuse of any external knowledge representations. The ontology itself was not only reused in KDE, but also in an experiment to measure the difference between reuse and a clean start in representing another maritime domain (MARPOL convention). This experiment showed a significant bootstrapping effect on the speed of encoding: from 1 page in three hours to 2.5 pages per hour.

The ontology has now reached a size of 3377 concepts and 11897 relations (including is-a). Between concepts and regulations 8289 'modlinks' are defined and the entire system now covers about 15% of the ship classification rules and Annexes I and II of MARPOL. During the encoding process, the size of the knowledge base was measured various times. The resulting graph (figure 2.4) shows the same bootstrapping effect: as more regulations are covered by the representation, less concepts and relations need to be added.

The modelling tools for the ontology, the Legal Encoding Tools (LET), were developed parallel to the building of the ontology itself. Available in both a Java application and an applet version, the LET offer a view of the legal sources that are to be represented from within the tools. This allows for an intuitive direct interaction between the ontology and the rules during encoding.

Specifying norms from within the LET is not yet supported; they are to be entered by hand in a text-editor. The norms can be imported into the knowledgebase after

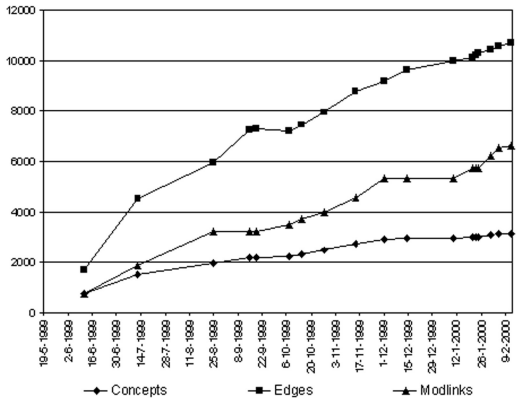

Figure 3: Modelling progress over time

which the computed exception structure can be investigated through a web-browser. The assessment engine can be directly queried through the same interface.

The knowledge base can be represented as a labelled, directed graph, with a fixed set of labels, and is trivially translated to RDF. The domain ontology has been translated to a *Protégé-2000* [7] knowledge base and *RDFS* for reuse in the KDE[3] workflow-management environment. In KDE the domain ontology was used to enhance search in arbitrary technical and legal documents used in business tasks associated with ship classification and survey (cf. [6]).

The reuse of the ontology in the KDE-project resulted in the development of new tools that exploited the representation of the ontology in RDF(S). The KWDE Applet is embedded in an agent-oriented workbench in which agents communicate to other agents through XML. It has a more flexible, concept-centred user interface that allows for the integration of multiple, separate and different types of (RDF) resources into the knowledgebase. This opened-up CR to other documents than a single legal text.

2.5 Limitations of the Encoding

The norm representation presented in this paper suits its purpose, but it does not capture all legal knowledge or all uses of the contents of legal documents. One element that is not covered is the assignment of blame for violation of the rules. This element has been explicitly omitted, because it is less relevant to the purpose of the system and the policies of a classification society. This means that the system is not particularly suitable for the preparation of defensive arguments after a disaster, for instance.

There are also complex deontic concepts like *right*, *liability*, *power*, *disability*, and *immunity* whose deontic impact cannot be encoded definitively in this deontic encoding. Known interpretations as they arise can be encoded, however. Observe for instance that the right of a certain category of agents in the domain may create certain permissions for these agents, but also certain prohibitions and obligations for other agents in dealing with the agents protected by the right. Liability of a certain category of agents creates permissions for other agents, whereas immunity creates certain prohibitions for other agents. A disability creates certain prohibitions for a certain category of agents,

[3]KDE, Knowledge Desktop Environment, Esprit Project EP28678

and a power grants certain permissions to a certain category of agents that other agents do not have. Encoding specific impacts of these concepts is possible, but rules introducing these concepts often remain indefinite in the sense that new consequences can be discovered and encoded in the future. The fact that you have a right to vote, for instance, creates a certain prohibition for others to interfere. This can be naively encoded, of course. What it means to "interfere" remains ill defined, however, because the commonsense knowledge needed to determine in what ways one could possibly prevent a vote is unbounded. When a certain category of cases is discovered to be a violation of this prohibition, this case can be encoded and used as a prohibition in the future. Note that written law also works in this way: If a court finds that some fundamental right is violated by some category of cases in a very non-obvious way, this category of cases may be explicitly prohibited in later law. Yet the existence of the fundamental right was apparently sufficient to come to the same conclusion in absence of this prohibition.

In regulations for very technical domains this practice seems ubiquitous: Regulations from a classification society are often merely concrete and clear-cut interpretations of what is prescribed in the abstract – and in a teleological flavour – in the annexes of conventions signed by the members of the United Nations International Maritime Organization (IMO). Another element that is conspicuously missing in the MILE representation format is a representation of how and why new interpretations come into existence. To capture this element to some extent one needs a *teleological perspective* on rules: Rules change because the rules did not serve their intended purpose, increasing safety or preventing pollution, well enough. Recent research in AI and Law has acknowledged the importance of this element in explaining interesting court decisions [2], but we do not think it is very useful to add this element to the MILE system. These concepts become relevant only in complex cases in court, when routine legal advice of the kind MILE, or a classification society, wants to offer has already failed.

The caveats in the knowledge base of interpretations of norms also apply to terminology. The terminological knowledge base is also never definitive because concepts are by necessity *open-textured* to a certain degree: Concepts are more rigorously defined as time progresses and new interpretations are established. New ways to mishandle systems on a ship are discovered regularly, and that influences the interpretation of the concept *arrangements such as to prevent any mishandling*, for instance. They also change because societal values or the available technology changes [10]. Concerns with maritime pollution, for instance, led to the introduction of the *segregated ballast tanker* in 1973 as a disjunct in the definition of *tanker*, effectively modifying this concept (or at least its enunciative definition) from then on.

3 POWER, UML, and OCL

The *POWER*[4] method represents the law in UML and OCL subject to a number of modeling restrictions to enforce some degree of uniformity on the model. The modeling restrictions include i.a. the possibility to define the interpretation of roles (no supertypes allowed, even though frame identifiers for functions, inverses, and even cardinality are allowed) and UML restricts the domain of a role to one concept (i.e. only one $\forall R.C$ or $\exists R.C$ restriction allowed in concept definitions for each R), which unfortunately results in a proliferation of unrelated roles not representing a distinct role

[4]This section is incomplete and does not pretend to be a description of the qualities of POWER as the previous section, only of UML as a 'logical' language. See [11] for more on POWER.

in the domain. This difference, and some other incompatibilities (see [1]), result from the difference in conceptualization between UML associations and RDF properties. In RDF a property is a self-contained resource, just like the DAML+OIL class, that can have properties (like supertypes, and domain and range restrictions). In UML an association only exists relative to its domain class. This difference can be solved with an extension to the UML MOF (cf. [1]), or a rather verbose mapping to RDF (cf. http://www-db.stanford.edu/ melnik/rdf/uml/).

This model is potentially richer in some aspects than the others, but lacks a complete interpretation for classification – UML subsumption and the satisfiability of OCL constraints is clear, but the combination of both is not. The combination of UML and OCL is not problematic for instantiation and retrieval of instances, but concept and instance classification are less relevant in UML. This is understandable because UML is a software specification language and most target platforms for implementation use a fixed concept hierarchy and only allow instances to change type (C++ etc.) at runtime with a relatively large performance (and coding effort) penalty. This can lead to a completely different modeling style. A positive consequence of this restriction is that cardinality restrictions and explicit enumeration of instances are not at all problematic. Another positive aspect of OCL is that it is sufficiently flexible to add interpretation constraints where needed. The software engineer adds OCL at his own risk – consistency of UML and free OCL combined is most likely undecidable for any reasonable interpretation.

A negative consequence for modeling style can be for instance that a deduction $Triangle \doteq (Polygon \sqcap \exists sides : \{3\}), i : Polygon, (i, 3) : sides \models i : Triangle$ where needed at runtime may be modeled with an ad hoc solution like

$$Polygon \sqsubseteq \exists isTriangle : \{true|false\}$$

Moves in the virtual instance classification hierarchy can then be effected with OCL constraints in the right places to assign a value $(i, true) : isTriangle$ if $i : Triangle$ is inferred. It is up to the programmer to decide where the right places are (but assistance from verification tools is possible). It is needless to say that UML class hierarchies are often flatter and less meaningful than concept hierarchies in a description classifier. Of course, definitions in a description classifier can also be abused to model other things than concept definitions.

The unrestricted use of the assignment operator makes OCL an imperative programming language (based on propositional dynamic logic), not just a description logic (which itself can be understood as a propositional dynamic logic where propositions are concepts, states instances, and roles transitions). This means that UML/OCL models can be only partially translated to a description logic, and only partially verified (although different verification algorithms or translations can handle different fragments).

On the other hand UML itself is a good choice for DAML+OIL ontology development and deployment because:

- It is a ”best practice” graphical representation. Many years of experience in software analysis and design by a variety of companies contributed to the formulation of UML.
- It is widely adopted in industry and taught in many university and professional development courses.
- It is supported by mature CASE tools. Therefore, existing GUI and consistency checking technology can be leveraged for DAML+OIL ontology tools.

- It is an open standard maintained by the Object Management Group. There is a systematic process for evolving UML based on public input.
- It has a documented semantic metamodel. Explicit constraints guide the development of well-formed UML models.
- It has extension mechanisms. Subclasses of UML elements can be defined (stereotypes) and new properties can be added to UML elements (tagged values). User defined constraints can be applied to UML elements.
- It has been effectively used to design XML DTDs and schemas. DAML ontologies are an extension of XML DTDs and schemas.
- A UML to XML mapping standard has been defined. This standard is called XML Metadata Interchange (XMI).

The *UML Based Ontology Toolset* (UBOT; http://ubot.lockheedmartin.com/) project is part of the DARPA Agent Markup Language (DAML) Program. It aims to define the interpretation for UML to DAML+OIL transformation, and software (AeroDAML, ConsVisor) for the production and verification of DAML+OIL ontologies in UML CASE tools. The primary goal of the mapping is to support translation from UML class diagrams to DAML ontologies. The assumption is that the UML class diagrams were created specifically for the purpose of designing DAML ontologies. This mapping does not attempt to map all elements on a class diagram to DAML. Legacy class diagrams that were not originally intended for DAML applications can be partially translated or they can be modified for DAML purposes. Figure 4 shows a part of an e-commerce ontology in UML and the result of translating it into DAML with the UBOT mapping. Figure 5 illustrates graphical annotation of a sentence in a web page and the result of translating it into DAML. The sentence says "Trey Clever is the CEO of the B2B company acronym.com".

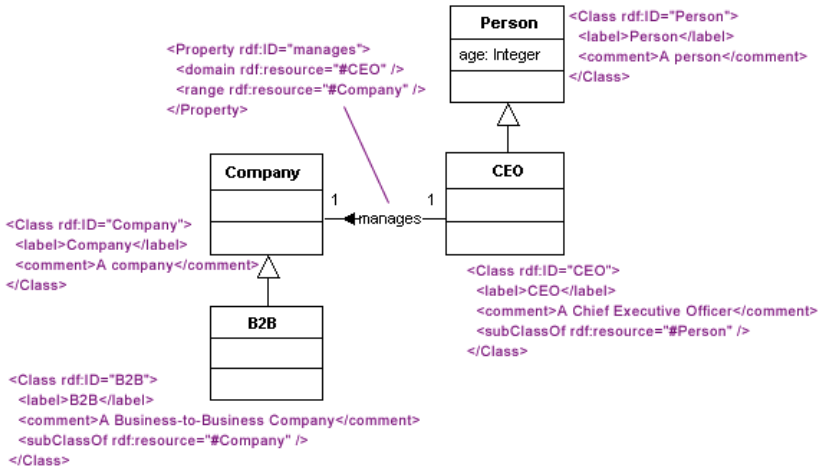

Figure 4: Mapping of UML to DAML+OIL Tbox.

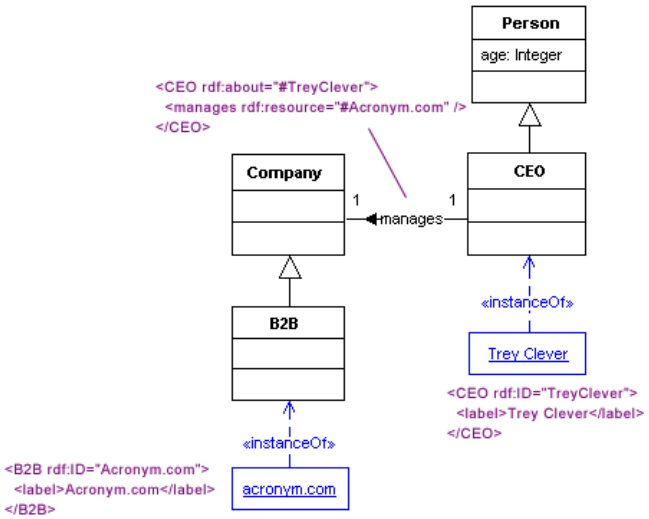

Figure 5: Mapping of UML to DAML+OIL Abox.

Tools that are already avalable are *DUET* and *VisioDAML*. The DAML-UML Enhanced Tool (DUET) provides a UML based environment for the development and manipulation of DAML ontologies. Core DAML concepts are being mapped into UML through a UML profile for DAML. DUET is available as an addin for Rational Rose. An ArgoUML version is forthcoming. VisioDAML consists of a Visio template and stencil. The Visio stencil contains shapes that represent the DAML+OIL language contructs, and can be used to make DAML drawings (but cannot import or export DAML XML).

4 Representing Cases and Generic Cases

To map the original CLIME/ONLINE cases and generic cases to DAML+OIL we have to solve the problem of the missing existential variables for queries (generic cases) and the more practical problem of the missing grounded facts (cases) in FaCT. The second problem, discussed in the next subsection, is part of the solution of the first.

4.1 Representing Cases as Concepts

If the description classifier language includes a *one-of* operator (an operator that allows definition of a concept as an explicit enumeration of its member instances), instances can be defined indirectly by defining an individual – a concept with only one member. The case of *a ship fitted with a main engine and two fire pumps, one driven by that main engine* or more formally $\{i_1 : Ship, i_2 : MEng, i_3 : FP, i_4 : FP, (i_1, i_2) : FW, (i_1, i_3) : FW, (i_1, i_4) : FW, (i_3, i_2) : DB\}$ can be transformed for practical purposes to a concept $Case \sqsubseteq (\{i_1\} \sqcap Ship \sqcap \exists FW : (\{i_2\} \sqcap MEng) \sqcap \exists FW : (\{i_3\} \sqcap FP \sqcap \exists DB : (\{i_2\} \sqcap MEng)) \sqcap \exists FW : (\{i_4\} \sqcap FP))$. On a conceptual level this can be understood as blurring the distinction between instance classification and

concept classification. Each instance now maps directly to a corresponding 'individual' concept. In the example above the instances are all folded into one, but clearly i_3 has been mapped to $(\{i_3\} \sqcap FP \sqcap \exists DB : (\{i_2\} \sqcap MEng))$. A case is always mapped to a conjunctive concept definition.

Unfortunately FaCT has no *one-of* operator (but DAML+OIL does). The function of the operator is to enforce co-reference, or to make sure that the interpretation of the 'individual' always maps to exactly one instance – the right one. If it is removed it can become impossible to distinguish between two different firepumps I_1 and I_2 that are both driven by the same main engine ('individual' I_3) if, in the absence of distinguishing information, $I_1 \doteq I_2$. Since there is always potential distinguishing information (cases are never the same, only *equivalent* for some purpose), all instances should therefore always be defined as $N \sqsubseteq C$ where N is the unique identifier for the instance. For purposes of case assessment this solution for the representation of cases is elegant enough: The distinction between an instance and its most specific concept description is immaterial for legal assessment. The advantage of the aproach is that it uses the description classifier, without any reconstruction afterwards. In addition it allows the case to be defined as a concept node itself, allowing multiple cases in one knowledge base at the same time without any external computation. Furthermore there is the added benefit that a set of cases can be (re)classified at the same time after a change to the ontology.

A slight disadvantage is that it requires all instances (even in different cases) to have a *unique* identifier, but that is also a RDF requirement for resources. This is why RDF toolkits usually include a unique URN generator. This is a good reason to ignore concept identifiers and use the human-readable *rdfs:label* instead for a descriptive tag. Given the flexibility of RDF, it is off course possible to link the concept describing the case to the RDF instance that represents it. This makes the representation compatible with Tbox-only validators like FaCT, and Abox and TBox rule engines like JESS or cwm.

4.2 Representing Generic Cases as Concepts

Representing generic cases as Tbox concepts instead of (generally much more expressive) $LOOM$ Abox queries may seem like a loss, but it is actually a very interesting idea: We get a generic subsumption procedure for generic cases for free. This means that we can define a standard for which a sound DAML+OIL document verification guarantees that the document (or collection of documents) contains a consistent and satisfiable norm system. Unfortunately this is not completely possible. Still, it seems more natural to look at the generic case (or the norm) as a concept (and at the same time an instance of 'generic case' ('norm' respectively)).

The folding procedure for facts sketched in the previous section can be generalized to existential variables in conjunctive queries (cf. [5]). Queries and the variables occuring in them can also be translated to concepts. In this case the reasoning required is unfortunately more complex and requires some reconstruction after classification to verify the result and to filter 'false positives'. Still a lot of the reasoning effort can be moved to the DAML+OIL validator and we know that the only errors remaining will be false positives: Cases classified under a generic case without actually matching them in the CLIME/ONLINE sense.

The general idea is fairly simple. Let $Case \sqsubseteq (Ship \sqcap \exists FW : ME_1 \sqcap \exists FW : (FP \sqcap \exists DB : ME_1) \sqcap \exists FW : FP)$ be a case of *a ship fitted with a main engine and two fire pumps, one driven by that main engine,* $ME_1 \sqsubseteq MEng$ an instance of

a main engine, and $N_1 \doteq (Ship \sqcap \exists FW : ME \sqcap \exists FW : (FP \sqcap \exists DB : MEng))$ and $N_2 \doteq (Ship \sqcap \exists FW : FP_1 \sqcap \exists FW : FP_2)$ be norms, where $FP_1 \doteq FP$ and $FP_2 \doteq FP$ are variables that can be matched with an instance of a fire pump. As expected, $Case \sqsubseteq (N_1 \sqcap N_2)$.

The problem is that $(N_1 \sqsubseteq N_2)$ is also true (because $FP_1 \doteq FP_2 \doteq FP$): We can enforce co-reference, but we cannot force FP_1 and FP_2 to match different instances of firepumps. Adding to the Tbox that $(FP_1 \doteq (\neg FP_2))$ merely makes the Tbox inconsistent, proving that it really means something different than $(FP_1 \neq FP_2)$. This assertion, $(FP_1 \neq FP_2)$, is usually taken for granted in query languages, but it is usually specified explicitly in standard logics.

The problem can be solved with an algorithm that obtains the set of generic cases that subsume the case, and for each generic case obtains the facts subsumed by each variable in the generic case, and finds a mapping from all variables to distinct instances in the case. Inputs to this algorithm are the concept names for the case, all instances occuring in it, the generic case, and all variables in it. All mappings are matches from the case to the generic case. Rule engines like JESS or cwm can be used to implement this small extension.

The human-readable *rdfs:label*, the descriptive tag, can be used to embed the text of the law in the generic case and variables.

5 Defining Norms and Cases

Not yet ready.

6 Discussion

In the examples I used cases and generic cases that are subtypes of ship (*the case is a ship fitted with* ..). This is different from ONLINE and CLIME, which represent generic cases as lists of propositions – more or less modeling the opinion that all facts are equally important. An ALC definition is always tree-like though (because Tbox assertions cannot define cyclical models), and if each role has an inverse it does not really matter which 'variable' is put at the root. A case or generic case each consist of a number of concepts.

FaCT does not yet support cardinality, numbers, or arithmetic functions in DAML+OIL and XML Schema. These constraints can be checked for instances by generation of JESS rules from DAML+OIL.

Earlier I noted that to translate UML to DAML+OIL, either the UML metamodel must be extended, or we have to use rather verbose mapping to RDF. Let's assume we use a mapping to UML to RDF that reifies the associations and association-ends of UML in RDF. A consequence of this would be that DAML+OIL axioms would also become very verbose and difficult to verify. A simple constraint $\forall R.C$ would have to be translated to $\forall R_1.(R \sqcap \forall R_2.C)$ where R is now a reified role, and R_1, R_2 an accessibility relation from domain to association, and association to range. This leads to a radically different style of modeling in DAML+OIL, and may lead to a reduction in expressiveness – this is an open question.

References

[1] Kenneth Baclawski, Mieczyslaw Kokar, Paul Kogut, Lewis Hart, Jeffrey Smith, William Holmes, Jerzy Letkowski, and Mike Aronson. Extending UML to support ontology engineering for the semantic web. In *Fourth International Conference on UML (UML 2001)*, Toronto, Canada, 2001.

[2] T. Bench-Capon and G. Sartor. Using values and theories to resolve disagreement in law. In J.A. Breuker, R. Leenes, and R.G.F. Winkels, editors, *Legal Knowledge and Information Systems (JURIX-2000)*, Amsterdam, 2000. IOS Press. ISBN 1.58603.144.9.

[3] A. Boer. The Consultancy Game. In J.A. Breuker, R. Leenes, and R.G.F. Winkels, editors, *Legal Knowledge and Information Systems (JURIX-2000)*, pages 99–112, Amsterdam, 2000. IOS Press. ISBN 1.58603.144.9.

[4] N. den Haan and J. Breuker. Constructing Normative Rules. In *Proceedings of JURIX'96*, pages 135–147, 1996.

[5] Ian Horrocks and Sergio Tessaris. A conjunctive query language for description logic aboxes. In *AAAI/IAAI*, pages 399–404, 2000.

[6] W. Jansweijer, J. Breuker, J. van Lieshout, E. van der Stadt, R. Hoekstra, and A. Boer. Workflow directed Knowledge Management. In *E-work and E-commerce: Novel solutions and practices for a global networked economy*, pages 755–762, Amsterdam, 2001. IOS-Press.

[7] N. F. Noy, R. W. Fergerson, and M. A. Musen. The knowledge model of Protégé-2000: Combining interoperability and flexibility. In *2th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000)*, Juan-les-Pins, France, 2000.

[8] L. Pallipoli, F. Pirri, and C. Pizzuti. Algorithms for Selective Enumeration of Prime Implicants. *Artificial Intelligence*, 111:41–72, 1999.

[9] P. Piwek, R. Evans, L. Cahill, and N. Tipper. Natural Language Generation in the MILE system. In *Proceedings of the IMPACTS in NLG Workshop*, Schloss Dagstuhl, Germany, 2000.

[10] E. Rissland and T. Friedman. Detecting change in legal concepts. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law (ICAIL-99)*, pages 127–136, New York (NY), 1995. ACM.

[11] Silvie Spreeuwenberg, Tom van Engers, and Rik Gerrits. The Role of Verification in Improving the Quality of Legal Decisionmaking. In Bart Verheij, Arno Lodder, Ronald Loui, and Antoinette Muntjewerff, editors, *Legal Knowledge and Information Systems (JURIX-2001)*, pages 1–16, Amsterdam, 2001. IOS Press. ISBN 1.58603.201.1.

[12] J. Straach and K. Truemper. Learning to Ask Relevant Questions. *Artificial Intelligence*, 111:301–328, 1999.

[13] R.G.F. Winkels, D. Bosscher, A. Boer, and J.A. Breuker. Generating Exception Structures for Legal Information Serving. In Th.F. Gordon, editor, *Proceedings of the Seventh International Conference on Artificial Intelligence and Law (ICAIL-99)*, pages 182–195, New York (NY), 1999. ACM.

[14] R.G.F. Winkels, D. Bosscher, A. Boer, and R. Hoekstra. Extended Conceptual Retrieval. In J.A. Breuker, R. Leenes, and R.G.F. Winkels, editors, *Legal Knowledge and Information Systems (JURIX-2000)*, pages 85–98, Amsterdam, 2000. IOS Press. ISBN 1.58603.144.9.