A Method for the Development of Legal Knowledge Systems

Pepijn R.S. Visser[1], *Robert W. van Kralingen*[2], and *TrevorJ.M. Bench-Capon*[1]

[1] LIAL - Legal Informatics at Liverpool
Department of Computer Science, University of Liverpool
PO Box 147, Liverpool, L69 7ZF
United Kingdom
{pepijn, tbc}@csc.liv.ac.uk

[2] Center for Law, Public Administration and Informatization
Tilburg University, PO Box 90153
5000 LE Tilburg
The Netherlands
R.W.vKralingen@kub.nl

Abstract

In this article we present a four-phased method for the development of legal knowledge systems. We set out from the well-studied CommonKADS method for the development of knowledge systems and tailor this method to the legal domain. In particular, we propose a generic legal ontology, and describe the creation of statute-specific ontologies to make the method more suitable for our purposes. In the construction of these ontologies we start from a theoretical analysis of the legal domain. The well-known example of the Imperial College Library Regulations (ICLR) is used to illustrate the method.

1. Introduction

Several methods are available for the design of knowledge systems. The essence of these methods is the division of the system-development process into a number of comprehensible phases. The result of each phase is a model of specific aspects of the system. Examples of such models are organisational models, addressing the system in its organisational context, and, functional models, specifying the tasks of the system.

Despite the attention system-development methods have received in the field of computer science, they have not been widely applied in the field of artificial intelligence and law. Hardly any research has been reported on the process of designing legal knowledge systems as such. In this article, we address the design of legal knowledge systems from a methodological point of view. In particular, we present a method for a stepwise construction of legal-knowledge systems, showing four major design phases: analysis, conceptual modelling, formal modelling, and implementation. Our point of departure is the CommonKADS method for knowledge-system development (*e.g.*, Breuker and Van de Velde, 1994). We tailor the method to the legal domain by adding domain-specific elements. In this article, we focus primarily on the conceptual and formal models of the system. Also, we present heuristics for assembling these models. We illustrate our method by discussing the creation of a small knowledge system that operates on a fragment of the Imperial College Library Regulations (Jones and Sergot, 1992).

The outline of this article is as follows. We begin with a short description of the legal-theoretical background in section 2. Next, we provide an overview of the method in section 3 after which we elaborate on three of the four phases. In section 4 we address the analysis phase, in section 5 the conceptual modelling phase, and in section 6 the formal modelling phase (the implementation phase is not described in this article). Finally, we conclude by presenting our main findings in section 7.

2. Legal-theoretical background

The stepwise construction of knowledge systems facilitates bridging the gap between knowledge and knowledge system. The models consecutively developed during the development process (as stated, in this article the focus is on conceptual and a formal models) can be viewed as intermediate representations. Bench Capon *et al.* (1987) list three advantages of such intermediate models. First, they

ICAIL-97, Melbourne, Australia © 1997 ACM 0-89791-924-6/97/06.$3.50

impose structure on knowledge acquisition and knowledge modelling. Second, they make the interpretation of the knowledge to be contained in the knowledge system more accessible. Third, they allow for a representation of knowledge that does not have to commit to the quirks of the implementation language.

In order for an intermediate representation to fulfil its function, it must meet with certain requirements. In our opinion, the most important of these requirements is that the representation must comply with ideas domain experts have on the structure of the domain to be represented. Since we take the legal domain as our research domain, we have taken legal theory as the point of departure for the outline of our conceptual and formal models. We start from institutional theories of law, such as the ones proposed by MacCormick and Weinberger (1986) and Ruiter (1993). Both MacCormick and Weinberger, and Ruiter turn to the theory of speech acts (Searle, 1969) in their analysis of the legal domain. In speech-act theory, the concept 'institutional fact' plays an important role. An institutional fact can be seen as 'an abstract, socially-defined entity or event'. The legal domain comprises many of these entities. We name some instances: legal institutions, legal definitions, legal performatives, and legal norms all qualify as institutional facts. In this article, we focus on legal norms. Other entities are discussed in Van Kralingen (1995).

Traditionally, legal theory has spent much attention on the concept of a legal norm (see for instance Hart, 1961; Von Wright, 1963; Ross, 1968; and Kelsen, 1991). Since the norm is the most salient construct of the legal domain, we have selected it as the point of departure for our intermediate representations. With Von Wright (1963, 1983) we define a norm as 'a statement to the effect that something ought to, ought not to, may, or can be done'. From this definition, we can learn that norms come in different types. Some norms regulate conduct ('ought', 'ought not' and 'may' norms pose commands, prohibitions and permissions, respectively), others regulate competence (*e.g.*, 'can' norms establish competence). However, the distinction between norms of conduct and norms of competence is not the only important point of departure for a classification of norms. Classifications can also be made on the basis of, for instance, the regulated object, the norm subject, or the conditionality of a norm (for a complete classification of norms, we refer to Van Kralingen, 1995). If we consider the regulated object, the distinction between norms of the *tun-sollen* and norms of the *sein-sollen* type comes to mind. The former type of norm regulates what ought to or ought not to be *done*, the latter type of norm regulates what ought to or ought not to *be* (*cf.* Von Wright, 1963, p.13). The norm subject forms another basis for a distinction between norms; norms can be addressed to individuals, sets of individuals, or collectivities (Van Kralingen, 1995, pp.42-44). Hence, dependent on the norm subject, we can distinguish between individual and general norms. The last distinction we discuss, is the one between hypothetical and categorical norms. If a norm has any conditions of application, we coin it a hypothetical norm, if a norm is applicable unconditionally, it is a categorical norm.

When creating a comprehensive conceptual model and, consecutively a comprehensive formal model, it must be possible to represent all types of norms. Consequently, in our ontology, the basis of the conceptual and formal model, we have taken the above-mentioned classification into account. Before elaborating on the form of the ontology, we provide an overview of the method.

3. An overview of the method

Although there are several design methods for knowledge systems, the application of these methods to the legal domain is not yet widespread. Often, the design of a legal knowledge system (henceforth: LKS) is a rather ad-hoc and ill-documented process. One of our research aims has been to tailor an existing knowledge-system development technique to the legal domain thus creating a dedicated method for the development of LKS. Ideally, such a method would provide guidance for all steps in the design of an LKS, providing better support for the designers of LKSs than the more general methods.

The method presented here largely adopts the CommonKADS framework (Breuker and Van de Velde, 1994). An important feature of this method is the division of the design process into separate phases. In the spirit of this method we distinguish:

(1) an *analysis phase*, (2) a *conceptual modelling phase*, (3) a *formal modelling phase*, and (4) an *implementation phase*. As in CommonKADS we specify - in phases 2 and 3, respectively - an informal and a formal *expertise model*. In this model we separate *domain knowledge* (specifying the static knowledge in the domain), and *control knowledge* (specifying how the domain knowledge is applied to realise a goal). Control knowledge consists of specifications of *inferences* (primitive reasoning steps), and *tasks* (a control structure over tasks and inferences).

The CommonKADS method has been criticised for a lack of support to specify legal domain knowledge (*e.g.*, Gardner and Spelman, 1993; Visser, 1995). For this reason we propose to supplement CommonKADS with a *legal ontology*, as developed by Van Kralingen (1995) and Visser (1995). The most important feature of this (frame-based) ontology is the distinction between *norm frames*, *act frames*, and *concept-description frames* (we elaborate on this ontology in sections 5 and 6). In addition to the legal ontology, we adopt the domain-analysis method KANT (Bench-Capon, 1991, Bench-Capon and Coenen, 1992; Visser *et al.*, 1997). This method is used to create a *statute-specific ontology*, defining the vocabulary (*viz.* predicate names) with which to instantiate the frame structures. Below, we discuss the four design phases in more detail (this method is based on Van Kralingen (1995) and Visser (1995))[1].

1. ANALYSIS PHASE
   a. *Domain identification*: Identify the legal knowledge that is to be contained in the LKS in terms of references to legal sources (*e.g.*, set of legal cases, articles in statutes, heuristics).
   b. *Task identification*: Identify the task(s) that the LKS has to perform using the domain knowledge. In particular, this should result in a description of the input, the output, and the problem-solving goals of the LKS. Both steps, *1a* and *1b*, are meant to determine the competence of the LKS.

2. CONCEPTUAL MODELLING PHASE
   a. *Method description*[2]: Provide an informal description of *how* the system will perform the task. Otherwise stated, describe the method used to realise the problem-solving goals by transforming the input into the output (use, for instance, the CommonKADS library of tasks). The method specification provides guidance in the acquisition of the relevant domain knowledge (see step *2c*). The result of this step is a hierarchical decomposition of the main task in a series of sub tasks. Also, the various tasks are allocated either to the system or to the user.
   b. *Domain ontology selection and adaptation*: Select an appropriate legal ontology and tailor the ontology - if necessary - to support the tasks and methods described. As stated before, we select the frame-based ontology as described by Van Kralingen and Visser (in the remaining steps of the method we take this ontology as the standard).
   c. *Knowledge acquisition and modelling*: Identify the norms, acts, and concept descriptions in the domain knowledge, and gather the necessary information that is needed to instantiate the frame structures (*viz.* the domain ontology). The result of this step is a set of instantiated frame structures, each with their contents described in (structured) natural language (shortly: the conceptual domain specification).

3. FORMAL MODELLING PHASE
   a. *Determine boundaries of control and domain knowledge*: Identify procedural knowledge embedded in the conceptual frame structures (*viz.* meta-level procedural norms of competence, and conflict-resolution knowledge) and decide whether to model this knowledge in the expertise model as domain knowledge or as control knowledge. Conflict-resolution knowledge, for instance, can be modelled as control knowledge (*e.g.*, in case we want to conduct explicit meta-level

[1] Although the phases are largely executed in the order listed here, the design steps should not be considered to be independent. Several iterations through all phases are likely to be necessary in practice.

[2] The use of the word 'method' here concerns the method of the LKS, and should not be confused with the system-design method presented in this article.

reasoning about conflicts), or as domain knowledge (*e.g.*, in case we 'compile out' conflicts). More details about this step can be found in Visser (1995).

b. *Define control knowledge*: Create a formal description of the tasks, recognised in step *1b* and *2a*. This description should specify the hierarchical decomposition of tasks, and the information that is passed between tasks in a formal language. Following CommonKADS, we refer to the tasks at the lowest level of the hierarchy as inferences. How they apply knowledge contained in the frame structures will be specified in step *3e*.

c. *Create statute-specific ontology*: This step aims at determining and defining the predicate relations that are used to express the domain knowledge in a formal language. It involves the application of the KANT method on the legal texts identified in step *1*. In particular, this step involves: (c1) the creation of a *TOO (Test-On-Objects) structure* (identifying entities and the test applied to them), (c2) the creation of a *EAV (Entity-Attribute-Value) structure* (identifying entities, their attributes, and the values these attributes can take), (c3) the creation of a class hierarchy (grouping the entities in a class hierarchy), and (c4) the selection of predicate names to model the class hierarchy.

d. *Formalise domain knowledge*: Model the knowledge described in the informal conceptual domain model by bringing together the formal ontology and the statute-specific ontology. This step results in the formal domain specification.

e. *Define inferences*: Define the inferences (primitive tasks) that link the control knowledge and the domain specification.

4. IMPLEMENTATION PHASE

a. *Select language and platform*: Select an appropriate language and platform to implement the formal descriptions of the tasks and inferences, and the domain specification.

b. *Implementation*: Implement the formal model in the chosen language (and platform).

In the remainder of this article we elaborate on phases 1, 2 and 3. The scope of this article does not allow us to extensively discuss all aspects of the process. Our primary focus is the modelling of domain knowledge in phases 2 and 3 (for guidelines on how to model task knowledge we refer the reader to Visser (1995)). In section 4 through 6 we illustrate the method by applying it to the Imperial College Library Regulations (henceforth: ICLR) example.

4 Analysis phase

The analysis phase is meant to outline the competence of the LKS. In our example, the domain-identification step (step *1a*) yields the articles and allowances of the ICLR (Jones and Sergot, 1992):

art. 1.	A separate form must be completed by the borrower for each volume borrowed.
art. 2.	Books should be returned by the date due.
art. 3.	Borrowers must not exceed their allowances of books on loan at any one time.
art. 4.	No book will be issued for borrowers who have books overdue for return to the library.

Book allowances: undergraduates: 6,post graduates: 10, academic staff: 20

In the ICLR example the execution of the task-identification step is merely a matter of choosing a task since many different tasks can be performed on the ICLR. We here choose to do an assessment task, and more in particular, the task of assessing whether in a given case description any norms of the ICLR are breached (and by whom). The case description, being the input of the task, is assumed to be expressed in terms of the following phrases: 'university status S of person P', 'person P has borrowed book/volume B', 'person P has book/volume B overdue', 'person P has completed a form for book/volume B', and 'the librarian lends a book/volume to P' (later on, these phrases will have to be stated more formally - see section 6). Note that we do not distinguish between books and volumes. The output of the task is a list of tuples of norms and agents, specifying the norms that are breached and by which agents.

5. Conceptual modelling phase

In the first step of the conceptual modelling phase we provide an informal description of the method with which the system will perform its task (step *2a*). Our task 'assessment of breach' evaluates a case in retrospect. Space limitations prevent us from discussing this form of knowledge extensively. In essence, the control of the task is an iteration of the following three steps (1) determine applicable concepts, (2) determine which acts have been performed, and (3) determine whether norms are breached. The task returns a list of breached norms. More details on the description of the method can be found in Visser (1995).

The second step in the conceptual modelling phase involves the selection (and adaptation) of a domain ontology (step *2b*). With the legal-theoretical background described in section 2 in mind, we have developed three structures for the representation of legal knowledge. We have named these structures norm frames, act frames and concept-description frames. The structures form the backbone of our ontology. Here, we discuss only the norm frame. The act frame is briefly touched upon in the description of the conceptual model and in the description of the formal model. The concept-description frame is not presented in this article (see Van Kralingen, 1995).

	Element	*Typification*
1	Norm identifier	The norm identifier (used as a point of reference for the norm).
2	Norm type	The norm type (norm of conduct or norm of competence).
3	Promulgation	The promulgation (the source of the norm).
4	Scope	The scope (the range of application of the norm).
5	Conditions of application	The conditions of application (the circumstances under which a norm is applicable).
6	Subject	The norm subject (the person or persons to whom the norm is addressed).
7	Legal modality	The legal modality (ought, ought not, may, or can).
8	Act identifier	The act identifier (used as a reference to a separate act description).

Table 1. A norm frame.

A norm frame is constituted by four primary and three auxiliary elements. The primary elements are the norm subject, the legal modality (distributed over two slots; norm type and legal modality), the act identifier, and the conditions of application. The auxiliary elements are the norm identifier, the promulgation, and the scope. In table 1, the elements are typified.

A norm frame adheres to the general conception of what a norm is. We can paraphrase (the four primary elements of) the structure as: 'under certain conditions, the norm subject is obligated, forbidden, permitted, or empowered (legal modality) to do something'.

The third step in the conceptual modelling phase is the acquisition and modelling of domain knowledge. (step *2c*). In essence, this step involves the creation of the (conceptual) frame-based model by filling in the frame structures. The language that is used to fill the structures can be characterised as 'structured English'. It contains means to represent textual constructions (*e.g.*, references, rule-exception structures and application provisions), means to represent the norm promulgation, means to typify the legal modality, etc. In this article, we do not elaborate on the conceptual language (see Van Kralingen, 1995).

We have developed a number of heuristics to guide the process of assembling a frame-based model. The scope of this article allows us only to briefly discuss the two core heuristics (for more details, see Van Kralingen, 1995). The first core heuristic reads: start at the core of a norm, act, or concept description. This heuristic aims at finding an appropriate starting point for the modelling process. The second heuristic governs the extension of the model. It reads: a new provision should be added to an existing frame if and only if adding the provision does not result in changes to more than one slot of the frame to which the provision is added (for the application of this heuristic the norm-identifier slot and the norm-promulgation slot are not taken into account since they are merely used as a means of referring to a norm frame and a means of representing the norm's promulgation, respectively). The rationale behind the heuristic is a representation in a minimal number of frames while preserving the original meaning of the regulation represented.

Applying the first heuristic to the ICLR yields the following (conceptual) norm frame:

(1)	
norm identifier:	'norm-1'
norm type:	Norm of conduct
promulgation:	ICLR article 1
scope:	ICLR
conditions of ap.:	Subject wants to borrow a book.
subject:	Borrower
legal modality:	Ought to
act identifier:	'complete-form'

In this representation, the act identifier 'complete-form' refers to a separate act description. For the elements of such a description, we have resorted to the work of Rescher (1967, 1970). An act description comprises elements such as an agent, an act type, a modality, a setting, and a rationale. The latter three elements are subdivided into sub-elements. For instance, the modality has been divided into a modality of means and a modality of manner. Due to space limitations, we do not provide the act descriptions corresponding to the act identifiers (see section 6 for a formal act description and Van Kralingen, 1995; Visser, 1995).

If we consider the second article of the ICLR we find that the second core heuristic prevents the article from being added to norm frame (1) since adding the article would result in changes to more than one slot, namely the conditions-of-application slot and the act-identifier slot. Consequently, a second frame is created:

(2)	
norm identifier:	'norm-2'
norm type:	Norm of conduct
promulgation:	ICLR article 2
scope:	ICLR
conditions of ap.:	Subject has borrowed a book.
subject:	Borrower
legal modality:	Ought to
act identifier:	'return-book-by-date-due'

The third article presents us with an interesting interpretation issue. It can be argued that two norms can be read from the article: one norm forbidding a borrower to exceed his allowance, and one norm forbidding the librarian to issue a book if a borrower has reached his allowance. We can represent both interpretations in separate norm frames:

(3a)		
	norm identifier:	'norm-3a'
	norm type:	Norm of conduct
	promulgation:	ICLR article 3
	scope:	ICLR
	subject:	Borrower
	legal modality:	Ought not
	act identifier:	'exceed-allowance'

(3b)		
	norm identifier:	'norm-3b'
	norm type:	Norm of conduct
	promulgation:	ICLR article 3
	scope:	ICLR
	conditions of ap.:	Borrower has reached allowance.
	subject:	Librarian
	legal modality:	Ought not
	act identifier:	'issue-book'

In fact, the phenomenon that one article comprises more than one norm is not uncommon (*e.g.*, Hart, 1961; Kelsen, 1991). For instance, in penal law, we often find provisions stating that a person will be punished if he performs a certain action. Such a provision can be interpreted as both a norm of conduct (a prohibition to perform a certain action) and a norm of competence (conferring a power onto an official to administer a certain sanction). Note that, while norm (3a) does not have any conditions of application, norm (3b) does.

6. Formal modelling phase

The first step in the formal modelling phase concerns defining more precisely the boundaries of control knowledge and domain knowledge, the two major types of knowledge in the expertise model (step *3a*). This step is necessary because legal sources, intuitively modelled as domain knowledge, often contain procedural aspects (which suggests to model them as control knowledge). Visser (1995) distinguishes two forms of procedural knowledge in statutes: meta-level procedural norms of competence, and conflict-resolution knowledge.

In the ICLR there are no meta-level procedural norms of competence. That is, there are no procedural norms of competence that express how other norms should be applied. Consequently, we do not have to decide how to model this form of control knowledge for our example system. Also, because there are no two norms or concept descriptions that can have conflicting conclusions, there is no conflict-resolution knowledge required (a librarian who issues a book where this is not allowed is considered to breach a norm rather than cause a conflict). Hence, we do not have to decide upon how we deal with conflicts. Hence, in the ICLR domain, all knowledge from the legal sources can be modelled as domain knowledge in the expertise model.

The second step in the formal modelling phase (step *3b*) concerns the definition of the control knowledge. As stated before, we do not address the specification of this form of knowledge in this article. A detailed description of the assessment task control knowledge can be found in Visser (1995).

The third step in the formal modelling phase (step *3c*) is the creation of the statute-specific ontology. This is done by applying the KANT method, the first step of which is the creation of a TOO structure. For the ICLR, the TOO structure reads:

borrower	completes *form* (for volume)
borrower	borrows *volume*
book	has a *date due*
borrower	has *allowance* (of books on loan)
borrower	has *book overdue* (for return to the library)
borrower	has *status*
librarian	issues *book*
undergraduate	has *allowance*
post-graduate	has *allowance*
academic-staff	has *allowance*

In our domain ontology all acts are assumed to be performed by an actor we have introduced the notion of a librarian to be able to represent that a book can be issued to a borrower.

Next step in the KANT method is the creation of the EAV structure, in which the entities are given attributes and the potential values of these attributes are identified. The EAV structure reads (values marked with an asterisk may have multiple instantiations):

entity	*attribute*	*value(s)*
book	has-id	book-id
borrower	completed-form	book-id*
borrower	borrowed	book-id*
book	date-due	date
borrower	allowance	integer
borrower	has-book-overdue	book-id*
borrower	has-status	{undergraduate, post-graduate, academic-staff}
librarian	issues	(book-id, borrower)*
undergraduate	has-allowance	integer
post-graduate	has-allowance	integer
academic-staff	has-allowance	integer

In the creation of the EAV structure, we have changed the tense of some of the attributes to obtain a more uniform terminology (*e.g.*, complet*ed*-form). Note that we interpret the ICLR such that books and volumes are the same.

For the creation of a class hierarchy we regroup the entities in the EAV structure and introduce some abstract entities. An abbreviated version of the class hierarchy for the ICLR reads (between brackets we list the attributes, potential values are left out here):

```
Thing
  Book (id, date-due)*
  Person (name, address)
      Librarian (name, address, issued)*
      Borrower (name, address, completed-form*, borrowed*,
              allowance, book-overdue*, status)
          Undergraduate (name, address, completed-form*, borrowed*,
              allowance, book-overdue*, status)
          Post-Graduate (name, address, completed-form*, borrowed*,
              allowance, book-overdue*, status)
          Academic-Staff (name, address, completed-form*, borrowed*,
              allowance, book-overdue*, status)
```

Classes lower in the hierarchy inherit the attributes of their parents. Note that we introduced the top level class *Thing* and the class *Person* (which is given a name and address as its attributes).

The class hierarchy is assumed to distinguish all relevant entities in the domain. For this reason, we use it as the basis for choosing predicate names. This is a process guided by heuristics. Briefly stated, a predicate *attribute-id(Class-id, Attribute-value)* corresponds to the entry *Class-id(attribute-id)* in the class hierarchy. For instance, the predicate *name(Person, Name)* corresponds to the entry *Person(name)* in the class hierarchy (note that in the class hierarchy presented above, *Person(name, address)* is an abbreviation of *Person(name)* and *Person(address)*).

```
id(Book, Id)
date_due(Book, Date)
name(Person, Name)
address(Person, Address)
issued(Librarian, Book, Borrower)
completed_form(Borrower, Book)
borrowed(Borrower, Book)
allowance(Borrower, Allowance)
status(Borrower, Status)
allowance(Status, Allowance)  **
undergraduate(Person)
post-graduate(Person)
academic_staff(Person)
book_overdue(Borrower, Book)
```

Predicates such as *borrowed(Academic-Staff, Book)* are left out since such predicates are effectively subsumed under the predicate *borrowed(Borrower, Book)*. For the same reason, we have left out predicates such as *name(Post-Graduate, Name)*. The predicate marked with ** is not found as a direct consequence of applying the heuristic mentioned. However, we added the predicate to avoid having to specify for each borrower what his or her allowance is.

The next step in the formal modelling phase (step *3d*) is the formalisation of the domain knowledge. For this step we need to formalise the frame structures (in this article we assume that we have formalised versions of the frame structures available - not of their contents). For a more detailed description of the formalisation process, including a discussion of the differences between the conceptual and the formal frame structures, we refer to Visser (1995) and (Visser and Bench-Capon, 1996). The main objective in this step is to express the knowledge in the conceptual frame structures in terms (*a*) of the general legal ontology (*viz.* the formal frame structures) and (*b*) the statute-specific ontology (*viz.* the predicate relations).

One of the differences between the conceptual and the formal frame structures is that in the formal norm frame, the conditions of applications slot has been split up in an object(-level) conditions slot (used to state conditions about the outside world) and a meta-level conditions slot (used to state (meta-level) conditions about other frame structures). For the condition slots a special set of reserved predicates is defined, the most important of which are: *breached(Person, Norm)* to state that a norm has been breached, *arithmetic(Expression)* to express necessary calculations, *true_from(T, Clause)* to state that a clause is truė at and after a certain point in time *realised(Agent, Event)* to state that an agent has realised an event, *function(FunctionCall)* to refer to an externally defined function, *occurs(Agent, Process, T-begin, T-end)* to state that an agent is involved in realising a process between two points of time, *capable(Agent, Act)* to state that an agent is capable of performing an act, *effectuate(Person, Modality, Norm, Act)* to state that a person ought (not) to do an act according to a particular norm, and the predicates *always_false* and *always_true* which effectively are a contradiction, and a tautology. Also, we use special predicates to refer to act frames (these predicates are not discussed here).

Below, we list the formal version of norm 2 and norm 3b. The time references are used to link predicates - and thus conditions - onto states (it is assumed that the case description consists of a chain of states and acts).

norm identifier:	norm_2
norm type:	conduct
promulgation:	{iclr_art_2}
scope:	{iclr}
time reference:	Today
object conditions:	
	true_from(Today, borrowed(Borrower, Book)) and true_from(Today, date_due(Book, Date_due)) and arithmetic(Today >= Date_Due)
meta conditions:	always_true
subject:	Borrower
legal modality:	ought_to
act reference:	return_book(Borrower, Book, Date_due)

norm identifier:	norm_3b
norm type:	conduct
promulgation:	{iclr_art_3}
scope:	{iclr}
time reference:	Today
object conditions:	
	true_from(Today, completed_form(Borrow, Book)) and true_from(Today, status(Borrower, Status)) and true_from(Today, allowance(Status, Allowance)) and function(number_of_books_borrowed(Borrower, Today, Number)) and arithmetic(Number < Allowance)
meta conditions:	always_true
subject:	Librarian
legal modality:	ought_not
act reference:	issue_book(Librarian, Borrower, Book)

In contrast to the conceptual model, the formal model has separate frames for *events* (acts that occur instantaneously) and *processes* (acts that have a duration). Also, a distinction is made between acts that occur in the world (*e.g., a* kills *b*), referred to as *physical acts,* and acts that are legal interpretations of acts that occur in the world (*e.g., a* murders *b*, or *a* manslaughters *b*), referred to as *institutional acts*. Below, we present the physical event of issuing a book.

event identifier:	issue_book
act:	issue_book(Librarian, Borrower, Book)
promulgation:	{iclr_art_4}
scope:	{iclr}
agent:	Librarian
act type:	physical
temporal setting:	always_true
spatial setting:	always_true
circumstant. setting:	
	true_from(Before, not(borrowed(Borrower, Book)))
time reference:	Before, After
Initial state:	{completed_form(Borrower, Book) not(borrowed(Borrower, Book))}
Final state:	{not(completed_form(Borrower, Book) borrowed(Borrower, Book))}

Note, that part of the event specification is an initial state and a final state. The set of clauses in the initial state are true in the state before the event takes place (tagged *Before*) and the set of clauses in the final state are true in the state after the event (tagged *After*). This idea is comparable to so-called add and delete lists in STRIPS-style planning systems (Fikes and Nilsson, 1971).

The last step in the formal modelling phase is the definition of inferences (not done here). Inferences link the tasks knowledge onto the domain knowledge (*viz.* the filled-in frame structures). They define how, for instance, the object conditions and the meta-level conditions slots are evaluated (in case of norm frames) and how initial state is transformed into the final state (in case of the act frames).

The formal model can be used as the basis for an implementation. In this article we have chosen to describe the formal model in a PROLOG-style language (which eases the implementation of the formal model in PROLOG), but other languages could have been chosen as well.

7. Conclusion

In this article, we have illustrated the applicability of the method with the help of a small benchmark problem. The article is necessarily a very brief description and several important issues have been left unaddressed. In Van Kralingen (1995) and Visser (1995) several steps from the method presented here have been applied to a substantial fragment of the Dutch Unemployment Benefits Act. This has resulted in a prototype system called FRAMER (which has been implemented in PROLOG). In several smaller research projects, the conceptual ontology has been used in diverse domains, such as penal law, administrative law and civil law. Its applicability has also been shown by Voermans (1995). We summarise our main findings:

- Legal knowledge systems often have an implicit conceptualisation. The use of ontologies to make conceptualisations allows us to compare and analyse - and thus to assess the merits - of different conceptualisations;
- Ontologies are a useful instrument during the construction of a legal knowledge system, in particular, for knowledge acquisition;
- The CommonKADS method as such provides little support for the specification of *legal* domain knowledge;
- Extending CommonKADS with ontologies of the legal domain makes the method more suitable for building legal knowledge systems;
- The distinction between a statute-specific ontology and a generic legal ontology proves useful;
- The method presented here provides a guided way of bridging the gap between domain knowledge and an operational prototypes;
- The method presented here is useful to create libraries of reusable problem-solving methods, domain ontologies, and domain models.

Acknowledgements

The research presented in this article has partly been carried out at the Department of Law and Computer Science of the University of Leiden (The Netherlands). The study at the University of Leiden was supported by a grant from the Foundation for Law and Public Administration (REOB) which is part of the Netherlands Organization for Scientific Research (NWO).

References

Bench-Capon, T.J.M., G.O. Robinson, T.W. Routen, and M.J. Sergot (1987). Logic Programming for Large Scale Applications in Law: A Formalisation of Supplementary Benefit Legislation, *Proceedings of the First International Conference on Artificial Intelligence and Law*, pp.190-198, Boston, Massachusetts, United States.

Bench-Capon, T.J.M. (1991). *Knowledge-Based Systems and Legal Applications*, APIC series, No. 36, Academic Press, London, United Kingdom.

Bench-Capon, T.J.M., and F.P. Coenen (1992). Isomorphism and Legal Knowledge Based Systems, *Artificial Intelligence and Law*, Vol. 1, No. 1, pp.65-86.

Breuker, J.A., and W. van de Velde (1994). *CommonKADS Library for Expertise Modelling, Reusable Problem Solving Components*, J.A. Breuker, and W. van de Velde (*eds.*), IOS Press, Amsterdam, the Netherlands.

Fikes, R.E., and Nilsson, N.J. (1971). *STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving*, Artificial Intelligence, Vol. 2, pp.198-208.

Gardner, K.M., and K.C. Spelman (1993). *The use of KADS to model an intellectual property legal practice: a case study*, Proceedings Kennistechnologie '93, pp.439-443, Amsterdam, the Netherlands.

Hart, H.L.A. (1961). *The concept of law*, Clarendon Law Series, Oxford University Press, Oxford, England.

Jones, A.J.I., and M.J. Sergot (1992). Deontic Logic in the Representation of Law: Towards a Methodology, *Artificial Intelligence and Law*, Vol. 1, No. 1, pp.45-64.

Kelsen, H. (1991). General theory of norms, *Translation of 'Allgemeine Theorie der Normen'*, Michael Hartney, Clarendon Press, Oxford, England.

Kralingen, R.W. van (1995). *Frame-based conceptual models of statute law*, Computer/Law Series, No.16, Kluwer Law International, The Hague, the Netherlands.

MacCormick, N., and O. Weinberger (1986). *An institutional theory of law: new approaches to legal positivism*, D. Reidel Publishing Company, Dordrecht, the Netherlands.

Rescher, N. (1967). *Aspects of action*, The logic of decision and action, appendix II, pp.215-219, University of Pittsburgh Press, Pittsburgh, United States.

Rescher, N. (1970). On the characterization of actions, *The nature of human action*, pp.247-254, Miles Brand (*ed.*). University of Pittsburgh, Scott Foresman and company, Pittsburgh, United States.

Ross, A. (1968). *Directives and norms*, Humanities Press, New York, United States.

Ruiter, D.W.P. (1993). *Institutional legal facts: legal powers and their effects*, Kluwer Academic Publishers, Dordrecht, the Netherlands.

Searle, J.R. (1969). *Speech Acts*. Cambridge University Press (Dutch translation by F.H. van Eemeren, 1977, Het Spectrum, Utrecht, the Netherlands).

Visser, P.R.S. (1995) *Knowledge Specification for Multiple Legal Tasks; A Case Study of the Interaction Problem in the Legal Domain*, Computer/Law Series, No.17, Kluwer Law International, The Hague, the Netherlands.

Visser, P.R.S., and T.J.M. Bench-Capon (1996). *The Formal Specification of a Legal Ontology*, Proceedings of the Ninth International Conference on Legal Knowledge-Based Systems (JURIX'96), Van Kralingen *et al.* (*eds.*), Tilburg, the Netherlands, pp.15-24.

Visser, P.R.S., T.J.M. Bench-Capon, and H.J. van den Herik (1997). A Method for Conceptualising Legal Domains: An Example from the Dutch Unemployment Benefits Act, *Artificial Intelligence and Law* (to appear).

Voermans, W. (1995). *Sturen in de mist ... maar dan met radar; de mogelijkheid van de toegepaste informatica bij het ontwerpen van regelgeving*. Doctoral thesis, Catholic University of Brabant, W.E.J. Tjeenk Willing, Zwolle, the Netherlands (in Dutch).

Wright, G.H. von (1963). *Norm and action; a logical enquiry*. International Library of Philosophy and Scientific Method. Routledge & Kegan Paul, London, England.

Wright, G.H. von (1983). *Practical reason*. Philosophical papers, Vol.I. Basil Blackwell, Oxford.