The Role of Verification in Improving the Quality of Legal Decision-Making

Silvie Spreeuwenberg
LibRT B.V.
Amsterdam
The Netherlands
silvie@librt.com

Tom van Engers
Belastingdienst
Utrecht
The Netherlands
t.m.van.engers@acm.org

Rik Gerrits
LibRT B.V.
Amsterdam
The Netherlands
rik@librt.com

Abstract. Many Public Administrations (PA's) use knowledge-based systems to support their legal decision making processes. The quality of the decisions made depends on the quality of the underlying knowledge. In the POWER research program (Program for an Ontology-based Working Environment for Rules and regulations) conducted by the Dutch Tax and Customs Administration (DTCA in Dutch: Belastingdienst) verification techniques are used to improve legal decision making. This paper describes a state-of-the art verification and validation technique for legal knowledge systems: VALENS. Within the POWER-program VALENS is used to verify legal knowledge. We will furthermore describe the experiences with VALENS in a recently finished project. The POWER-program in which VALENS has been used aims at improving law enforcement. In POWER methods and tools have been developed that help legislation drafters to improve the quality of (new) legislation. Legislation is transformed to formal legal specifications that can be used in knowledge-based systems. These formal legal specifications can also be subjected to further inquiries aimed especially at discovering anomalies. One of the aims of POWER is to improve legislation quality by communicating detected defects to the legal sources. We will give an example to show that communicating anomalies to a domain expert in an intuitive way is a non-trivial task. VALENS makes use of the fact that POWER-models support tractability (i.e. the formal specifications contain references to the original knowledge-sources they are based upon). This allows VALENS to communicate the verification results to domain experts, e.g. legislation drafters. They may consequently adjust the legal sources and solve the defects. Currently we are further improving the representations of anomalies to make them easier to understand for legislation drafters and domain experts.

Keywords. Verification, Validation, Legal knowledge systems, legal knowledge representation, Methodologies for the development of legal systems.

1 Introduction

Drafting and implementing new legislation is a rather time, energy and money consuming process. The many inter-connected processes and the large number of people involved make

Silvie Spreeuwenberg, Tom van Engers and Rik Gerrits, 'The Role of Verification in Improving the Quality of Legal Decision-Making' in Bart Verheij, Arno R. Lodder, Ronald P. Loui and Antoinette J. Muntjewerff (eds.), *Legal Knowledge and Information Systems. Jurix 2001: The Fourteenth Annual Conference*. Amsterdam: IOS Press, 2001, pp. 1-15.

it very vulnerable to errors. Varying interests have to be aligned and communication difficulties due to differences in technical jargon have to be overcome in both drafting and implementing changes to legislation or even completely new legislation. The knowledge and experience needed to create new laws, specify, design and implement procedures and systems in legislative domains is very scarce. However, getting the right knowledge at the right place at the right time is a critical success factor for the ability to effectuate the legislative power to regulate and control.

The Dutch Tax and Customs Administration (DTCA) has started the research program POWER (Program for an Ontology-based Working Environment for Rules and regulations), that aims to develop a method and supporting tools for the whole chain of processes from legislation drafting to executing the law by government employees. The goal of the POWER-program is to improve legislation quality by the use of formal methods and verification techniques. Integration of the VALENS tool in the POWER tool-suite supports the verification of legal models.

In this article we will describe the role and implementation of verification techniques in the POWER-program. The final goal is to improve the quality of law-enforcement and optimally support legal experts when drafting legislation. Therefore we are developing a knowledge representation in which anomalies found in the formal models of legislation can be easily understood and resolved by legal experts.

2 Verification of Legal Knowledge Systems

Public administrations like the Dutch Tax and Customs Administration (DTCA or Belastingdienst in Dutch) have the institutional task to enforce the law. To be able to fulfill this task a fine-tuned series of processes have to be executed. These processes have to be connected to the series of forth bringing processes of new legislation. These processes of drafting and implementing new legislation are rather time, energy and money consuming. In these processes a large number of people are involved and the inter-connectedness of these processes makes the overall process vulnerable to errors. To make the situation even more complex; varying interests have to be aligned and communication difficulties due to differences in technical jargon have to be overcome in both drafting and implementing changes to legislation or even completely new legislation. The knowledge and experience needed to create new laws, specify, design and implement procedures and systems in legislative domains is very scarce. The problem public administrations have to cope with is how to make available the right knowledge at the right place at the right time. For public administration knowledge is a critical success factor in the effectuation of the legislative power to regulate and control society.

To support this task the DTCA uses many supporting information systems including knowledge-based systems. These knowledge-based systems are used in diagnosing and selecting cases for specific handling processes and in the handling processes themselves. From a legal view point it may be argued that knowledge based systems are not suited for the scrutineous judgement of cases (including the context of the case) as could be done by people. However the need to achieve equality before the law and the massiveness of the administrative processes make the use of knowledge-based systems in this kind of processes quite acceptable if precautions to prevent injustice are taken.

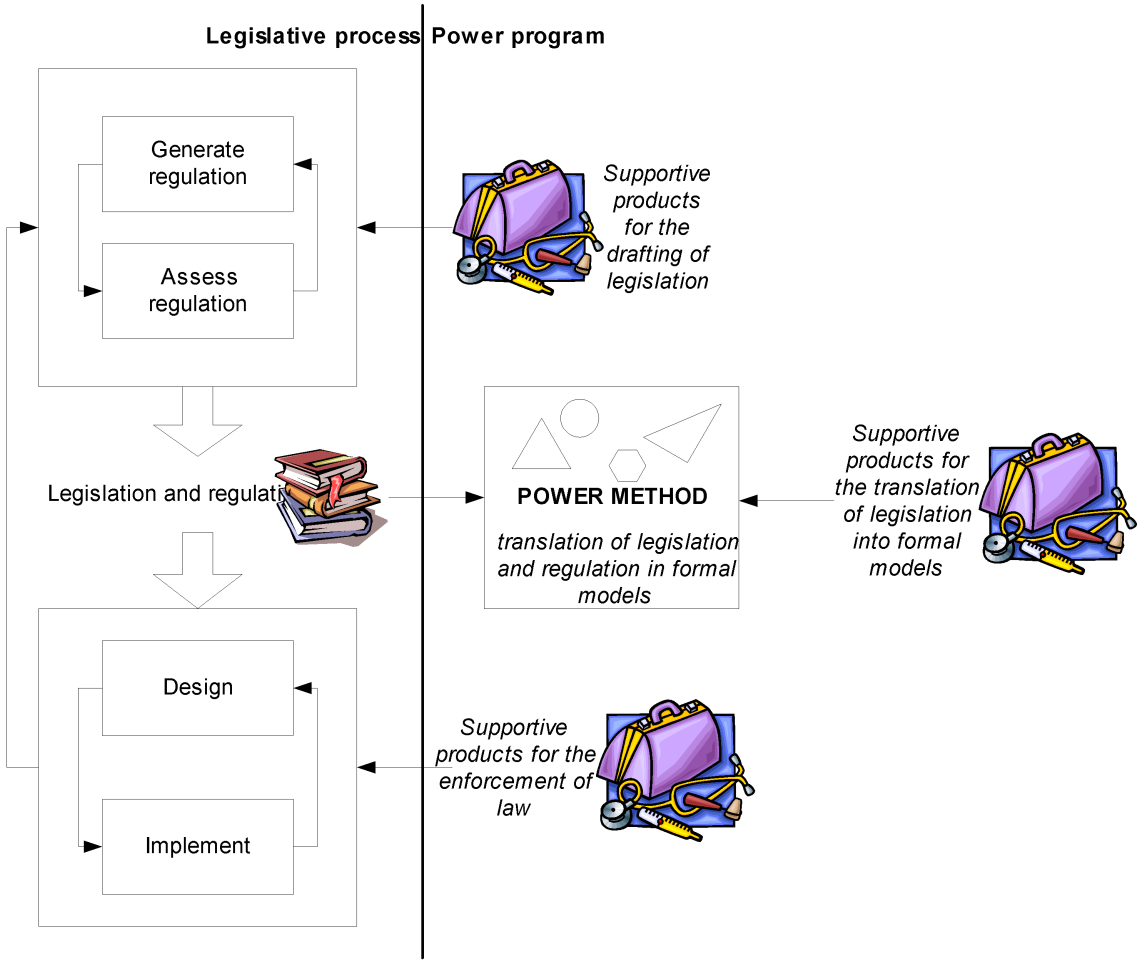

Figure 1: The POWER Program

The quality of the law enforcement depends on the quality of the legislation itself and on the quality of the knowledge-based systems that are actually used in the client handling processes as well. Many approaches have been described that aim at improving the quality of legislation (see e.g. Voermans [1]) or improving the quality of knowledge bases used in knowledge-based systems (see e.g. Preece [2] and Vanthienen[3]). The verification process as applied in the POWER-approach serves both perspectives.

The POWER-program (Van Engers et al [4], Van Engers, Kordelaar [5], Van Engers et al. [6]) has its background in the DTCA's earlier experiences with knowledge-based systems. The DTCA has built knowledge-based systems for many purposes, especially supporting compliance risk assessment, tax-payer case diagnosis and selection, since 1989. These knowledge-based systems proved to be useful systems, but the side effects caused when creating them were perhaps even more important. We found that the specification process in which experts from different disciplines and backgrounds were involved, helped to make knowledge explicit that would otherwise have remained implicit. The knowledge could furthermore be specified in a way that made it easier to establish its validity.

Based on the insight mentioned above the DTCA started the POWER-program that aims at achieving the following main goals:

- Improve legislation quality[1]
- Streamline legislative drafting effort
- Improve quality of law enforcement
- Reduce law enforcement implementation time
- Reduce law enforcement implementation effort

In POWER we focus on the knowledge specification but the program aims to support the whole chain of processes from legislation drafting to implementing law enforcement. In figure 1 the relations between the processes that are part of the POWER-program are visualized. The POWER-method is a central process in this program. This method has been developed to translate legal knowledge sources into formal models. These formal models are used for several purposes. Two of these purposes are:

1 Improving the quality of the legislation
2 Providing a basis for (knowledge-based) systems design.

Both purposes can benefit from a verification process that helps to detect anomalies. In the next section a description of the POWER-method developed thus far is given. This description focuses on the elements of the method that are important for the implementation of a verification process in the POWER-program.

3 The POWER Method

The POWER-method is based on conceptual modeling of legislation and regulations into formal legal specifications. The product of applying the POWER modeling process is, however, not only a conceptual model. We want to produce representations of legislation that computers can reason with. A set of coherent specifications can be delivered as a *knowledge component* [2]. The conceptual model is therefore used in a verification process and, together with a task model, used to generate a knowledge component. In the next paragraphs this process is described in more detail.

3.1 The POWER Modeling Process

Overviewing the POWER process it consists of a number of iterative sub-processes (see figure 2):

1 Translation of legislation and regulations into conceptual models, including completion of the models by expert knowledge elicitation
2 Re-factoring of conceptual models into coherent conceptual models

[1] Legislation quality is defined as absence of anomalies, law enforceability, and effectiveness in obtaining intended effects.

[2] A component is defined as a coherent package of software artifacts that can be independently developed and delivered as a unit and that defines interfaces by which it can be composed with other components to provide and use services (D'Souza [7]).

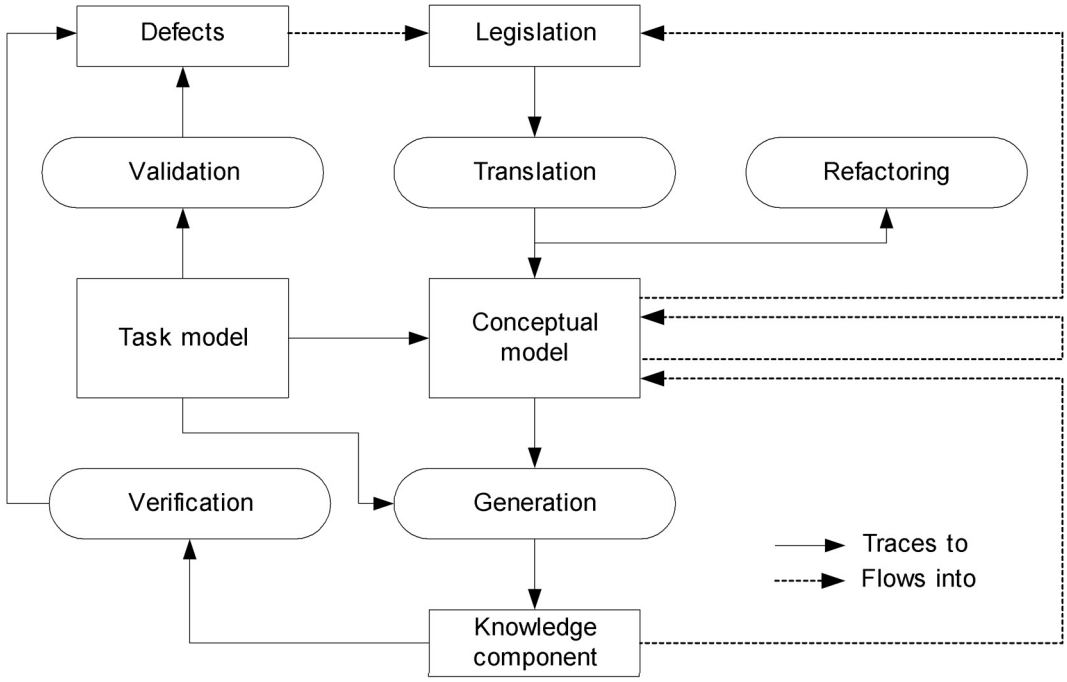

Figure 2: The POWER Modelling Process

3 Verification of conceptual models, including the detection of incompleteness and identification of missing legislation and regulations
4 Generating knowledge-based components for application frameworks, creating knowledge-based systems that can be delivered for implementing law enforcement
5 Testing and validating knowledge components, including the involvement of experts for certification of the knowledge components

The order of the sub-processes may depend on the route chosen by specific legal drafting or law enforcement implementation projects. Eventually, whatever route is chosen, the legislation, conceptual models and knowledge components will establish traceable refinement relationships.

The modeling language used is a proper extension of the ICT-industry standard Unified Modeling Language (UML), largely based on version 1.3, but including some proposed features of the upcoming version 2.0 that are particular to Catalysis (D'Souza [7]). Part of UML 1.3 is the Object Constraint Language (OCL), which is used to express constraints that apply to the model.

3.2 Translation of Legislation into a Conceptual Model

In POWER, we use a translation process that is a somewhat adapted and more rigidly described form of Catalysis business modeling. The difference is due to:

- the deliberate absence of business process information in legislation, which is more often than not left for the law enforcement agencies to set up as effectively as possible, albeit compliant with the legislation;
- the enhanced uniformity of conceptual models, delivered and required by the automated tool support of the POWER method;
- the particular features of the legislative domain, mainly the explicit references whereby a structure block in legislation (e.g. a chapter, article) refers to related structure blocks, combined with the need to create an independent model of each structure block for tractability purposes.

The structuring concept of the UML is the package. Packages are the container concepts that delimit a part of a model and have a hierarchical containment structure. In the translation step, each structure block of the legislation[3] (e.g. law, chapter, section, article, sub, part, sentence) is translated into a package. The hierarchical relationship of legislation structure blocks is reflected in the hierarchical nesting of packages.

Typically modelers have no legal background. Hence, they must often check with legal experts, who are familiar with the legislative culture, to validate the formal interpretation of legal structure blocks in the model. During this in-translation validation, the modelers attempt to detect standard interpretations in the legislative culture, identifiable by a typical legal phrasing. These standard interpretations and typical legal phrasings, are then described with the corresponding formal models, and are called translation patterns.

The in-translation validation with legal experts will also uncover semantic anomalies in the legislation (Kordelaar [8]). Semantic anomalies can occur if the rules used in the legislation or regulations lack clarity. In the POWER-translation processes detected anomalies are reported to legislators. They can repair the anomalies in the legislation before the law is enforced. Thus not only the quality of the legislation is improved but also its enforcement is also facilitated.

Tractability is an important feature in the POWER-method because in legal domains we need to be able to point at the sources of our knowledge on execution time (to be able to motivate our decisions). If we preserve the refinement between the original legal sources (e.g. a specific article in a specific law or a part of case law) we are also able to make impact analyses when we have to make changes in existing legislation. We use the same references in the communication process with legislation drafters who are likely to understand legal texts better than formal representations (e.g. when validating models or reporting anomalies).

3.3 Generation of Knowledge Components

The conceptual model containing the formal specifications of a specific legal domain can be generated into a knowledge component for a modeled task, that is executable by computers. In the context of the POWER project, a knowledge-based component is defined as *a rule based system that supports interfaces to an application framework to perform automated*

[3]The names and kinds of structure blocks depend on legislative culture: they are related to the authoring legislative body or country in a certain time period. Since conceptual models and the translation process have to be applied regardless of legislative culture, both the conceptual model formalism and the translation process description have to remain culture-free.

knowledge intensive tasks. These knowledge components form the basis for a test against syntactic requirements, i.e. the verification process (see e.g. Den Haan [9] and Kordelaar [8]) and the knowledge components can also be used to simulate the effects of legislation and regulations for a certain population (see e.g. Svensson [10]).

To execute the integrated models we need an executing environment that supports:

- Object Orientation (OO).
- An inference engine.
- Component based development (CBD).

An inference engine prevents us from having to restructure the knowledge into a procedural model that has to be evaluated in a particular (task dependent) order. The OCL invariants must be mapped on a one-to-one basis on concepts of the target language. These concepts are usually called (business) rules.

If we want to deliver executable components that can be plugged into an application framework, the component has to be 'framework-aware', which means that the component has to be designed in a way that it is aware of other components that use it. In this paper we will not elaborate on CBD principles.

A tool called FORCE (Factory for Object-oriented, Rule-based and Component-based engineering) generates these components. This tool takes an integrated model (UML/OCL and Process model) as input and generates a knowledge base by conducting the following steps:

1 Transition to a general programming system
Since the POWER-method wants to be (programming) language independent, we have to come up with an approach that helps us to set up a (automated) software factory that can operate in any programming environment that supports the three requirements mentioned above.
To preserve maximum independency the POWER-team has developed a conceptual system that supports the three basic requirements with no specific (industry) language features. This general programming system (GPS) consists of all concepts needed to fulfill the requirements of the knowledgebase generation. The idea is to map all elements of the UML/OCL and Process models on general concepts, to be supported by the target language. An extract of this mapping is shown below.

Paradigm	**Model Concept**	**GPS Concept**	**Concept in C++**
OO	Type	Class	Class
OO	Object	Instance	Object
RB	Invariant	Rule	Method
RB	Task	Inference task	Method
CBD	Interface	Interface	Abstract class
Process	Task	Task	Method

2 Parse and translate invariants
The OCL invariants need to be parsed into a parse tree. If the OCL statement is valid we use this tree to generate the target language. We generate the language of a commercial development environment, but the parse tree will also be used to generate Java.

3 Generate code
Based on the mapping between the concepts in the GPS and the target language (C++, Java or any other language or programming environment that supports the three basic requirements), all the constructs are generated in top-down order.
When all parts of the source are generated, the component is ready to be tested and deployed. It is up to the user of the FORCE-tool how the source code is compiled, based on the deployment models that are supported by the target-programming environment. These components can be subjected to a verification process (Preece et al. [2], den Hartog [6]).

4 Implementation of Verification in POWER

In the beginning of the '90s, universities devoted much attention to verification and validation of rule based systems. Some tools were developed to verify rule bases of which Preece [11] has given an overview and comparison. An even more extensive overview comes from Plant [12] who lists 35 verification tools built in the period 1985 – 1995. The verification[4] tools can be compared using a number of criteria. We have compared some of the widely known systems using the following criteria:

1 Anomalies detected by the tool
2 Language supported by the tool
3 Focus and behavior of the tool in the analysis or development phase of a system

The first criterion is formed by the anomalies that are detected by the tool. Some tools do not detect anomalies in a chain of logic, for example the Rule Checker Program (RCP) [13] and CHECK [14]. Others like RCP, CHECK and EVA [15] do not detect missing rules and unused literals. VALENS is complete with respect to the anomalies defined by Preece [2].

Another criterion is the language that is supported by the tool. Most verification and validation systems, which verify a knowledge base, cope with a restricted language, for example first order predicate logic [17][18] or formal specification language [16] as opposed to the rich language of a (fourth generation) programming environment. There are also tools which have their own internal language defined and which, manually or automatically, translate diverse languages into the internal language. EVA is an example of a system with its own internal language and provides a set of translation programs that translate the rule languages of some expert system tools (for example, ART, OPS5 and LES) to an internal canonical form, based on predicate calculus. PROLOGA works the other way around, it allows a user to create and verify decision trees and then generates code in diverse programming languages (for example, Aion, Delphi and C++). COVER and VALENS work in the programming language they where developed with, which is respectively Prolog and Aion[5]. VALENS is a verification component with a clearly defined interface description. At this moment the interface is implemented for knowledge bases developed with Aion.

The last criterion for comparison of verification tools is their respective behavior in the analysis and development phase of a system. The work of Nouira and Fouet [17] concentrates on the analysis phase of a system but results in a valid and executable knowledge base. The

[4] Tools that focus on validation or testing, like KVAT [22], are excluded from this definition.

[5] Aion is a widely used commercial development environment for rule based systems and a product and trademark from Computer Associates. Aion is a new version of AionDS.

work of van Harmelen [16] also concentrates on the analysis phase and validates formal specification language. The idea is that the formal specification has to be translated to a programming language to obtain an executable program. VALENS can be used by a developer after or during construction of a KB or can be integrated in a tool that allows users to write their own business rules. The output of the tool is a document in which all invalid rules (combinations) detected are reported.

The flexible nature of the VALENS verification component, the completeness and accuracy of the verification algorithms, and the possibilities for integration of VALENS in a modeling workbench have resulted in the decision to integrate VALENS in the POWER-program.

In the next section a description of the VALENS developed so far is given. This description focuses on the aspects of the tool that are important for the integration in the in the POWER-program.

4.1 Verification Algorithm

The verification algorithm that VALENS uses performs three main steps:

1 Construction of meta model
In this step all rule constructs, necessary to reason with the rules in the KB are instantiated. This step is performed on a "when needed" basis to reduce performance overhead.

2 Select potential anomalies
Potential anomalies are selected with the use of heuristics. These heuristics have been designed as meta rules but are implemented as procedures due to performance considerations.

3 Proof anomalies
The theses (potentially invalid rules) are proved by running the rules to be tested in a forward chaining mode, while providing them with the right truth-values (input). We call this process proof-by-processing. The proof-by-processing algorithm resembles the "saturation" process used by Nouira and Fouet [17]. Nouira and Fouet generate the largest possible number of properties for an object that would certainly appear during a real execution and then start forward chaining during which it tries to fire constraints. In VALENS the heuristics defined in the meta rules search for the smallest number of properties for an object of which the potential anomaly can be proved. VALENS does not use explicit constraints. The knowledge contained in the constraints of Nouira and Fouet are integrated in the meta rules of VALENS. In contrast to the meta rules of VALENS, the constraints of Nouira and Fouet contain semantic knowledge. Semantic meta rules are foreseen but not yet implemented in VALENS.

For a more detailed description of the proof-by-processing algorithm used in VALENS to detect anomalies the user is referred to Gerrits and Spreeuwenberg [18].

4.2 Benefits of Proof-by-processing

The proof-by-processing algorithm has some benefits in comparison with the algorithms based on formal logic. The most important benefit is that proof-by-processing allows us to deal with predicate logic (i.e. functions are allowed to be used in rules). The functions that

are used in the rules may contain procedural algorithms.

A second important benefit of the proof-by-processing algorithm is that we can work in an object-oriented (OO) environment. The rules work on attributes of the current instance. These attributes can be inherited.

A third and very important benefit of proof-by-processing is that there can be no discrepancy between the run time logic and the logic used in the validation process if the inference engine used in the verification process is the same as the inference engine used to evaluate and fire the knowledge rules in the application.

4.3 Integration of VALENS in the POWER Methodology

VALENS verifies knowledge rules in an executable environment because the proof by processing algorithm is based on the fact that the knowledge can be executed and the inference engine that executes the knowledge can be traced (i.e. the inference engine is able to supply information about the events that have taken place during the inferencing process).

Therefore VALENS is integrated in the FORCE tool that takes an integrated model as input, generates a knowledge base and verifies that knowledge base.

The user selects the tasks within the knowledge base that need to be verified. Invalid rules are reported in a HTML document. An anomaly is defined in [19] as a minimal set of rules, eventually associated to an input fact set that is a sufficient condition to prove an ‘anomaly’. In the HTML document we show, beside this information, the rule chain that caused the anomaly to be detected.

Groups of knowledge rules and the status of faults found during verification can be stored in a database to allow regression testing

4.4 Experience with VALENS in the POWER Methodology

The methods and infrastructure that have been designed in the POWER-program support the continuous improvement of knowledge productivity. In POWER we present an architecture that facilitates the processes mentioned and we have created a platform (and communication vehicle) for different stakeholders in the implementation processes.

We had the opportunity to model the (concept version of the) new Dutch Income Tax Law (an ambitious piece of legislation and the first major law revision in years). Since almost nothing of the old law on Income Tax was to remain intact, we were asked to look for anomalies (and to create a basis for further implementation and supporting infrastructure). We modeled the Income Tax working closely together with members of the knowledge groups that have to deal with income tax problems.

Figure 3 shows an example taken from the POWER-model of the new Dutch Income Tax Law. The two articles specify the income tax deduction.

The first article specifies the deduction type based on the tariff group, a tax payer is assigned to. The second article specifies how a tariff group is assigned to a tax payer. The translation of the bold phrases: if the tax payer is assigned to tariff group 1, he may deduct an amount identified by “bovenbasisaftrek”. If a tax payer may not deduct an amount identified

LB Art. 20
Lid 2. De belastingvrije som bedraagt:
a. ten aanzien van de in tariefgroep 0 ingedeelde werknemer: nihil;
b. ten aanzien van de in tariefgroep 1 ingedeelde werknemer: de bovenbasisaftrek;
c. ten aanzien van de in tariefgroep 2 ingedeelde werknemer: eenmaal de basisaftrek, vermeerderd met de bovenbasisaftrek;
d. ten aanzien van de in tariefgroep 3 ingedeelde werknemer: tweemaal de basisaftrek, vermeerderd met de bovenbasisaftrek;
e. ten aanzien van de in tariefgroep 4 ingedeelde werknemer: de basisaftrek, vermeerderd met de bovenbasisaftrek en de alleenstaande-ouderaftrek;
f. ten aanzien van de in tariefgroep 5 ingedeelde werknemer: de basisaftrek, vermeerderd met de bovenbasisaftrek, de alleenstaande-ouderaftrek en de aanvullende alleenstaande-ouderaftrek;
De belastingvrije som, bedoeld in de vorige volzin wordt verhoogd met de ouderenaftrek, dan wel met de ouderenaftrek en de aanvullende ouderenaftrek ten aanzien van de in tariefgroep 1, 2, 3, 4 of 5 ingedeelde werknemer die de ouderenaftrek, onderscheidenlijk de ouderenaftrek en de aanvullende ouderenaftrek geniet.

LB Art. 21
Lid 1. Ingedeeld wordt:
a. in tariefgroep 0: de werknemer die geen basisaftrek en geen bovenbasisaftrek geniet;
b. in tariefgroep 1: de werknemer die niet de basisaftrek maar wel de bovenbasisaftrek geniet;
c. in tariefgroep 2: de werknemer die de basisaftrek en maar niet de basisaftrek van een ander of de alleenstaande-ouderaftrek geniet;
d. in tariefgroep 3: de werknemer die mede de basisaftrek van een ander geniet;
e. in tariefgroep 4: de werknemer die alleenstaande-ouderaftrek maar niet de aanvullende aleenstaande-ouderaftrek geniet;
f. in tariefgroep 5: de werknemer die de aanvullende alleenstaande-ouderaftrek geniet.

Figure 3: New Dutch Income Tax Law

by “basisaftrek”, but may deduct an amount identified by “bovenbasisaftrek”, he is assigned to tariff group 1. These two phrases are modeled in UML/OCL as shown in figure 4.

They are generated as follows into a rule-based environment:

```
rule deductionType
ifrule current._tariffGroup.GroupNr = 1
then
    current._deduction.type = ”BovenBasisAftrek”
end
rule tariffGroup
ifrule current._deduction.type ¡¿ ”BasisAftrek”
and current._deduction.type = ”BovenBasisAftrek”
then
    current._tariffGroup.GroupNr = 1
end
```

When FORCE generates a knowledge base from this model VALENS will detect circularity and present this in an HTML report from which an extract is shown in Figure 5.

We found more then 150 anomalies that were not detected by knowledge groups before. The anomalies were reported to the drafters and repaired. The effectiveness of the feedback process depends of course on representation. Therefore we have conducted some research on law-representations that promote the communication between legislative- and IT-experts (by means of a cognitive ergonomic study).

4.5 Exploitation of the Results of Verification in POWER

The objective of the verification of the POWER models is not only to acquire a certificate of consistency and completeness for the knowledge components which are used to support decision making but mainly to improve the quality of the legal sources.

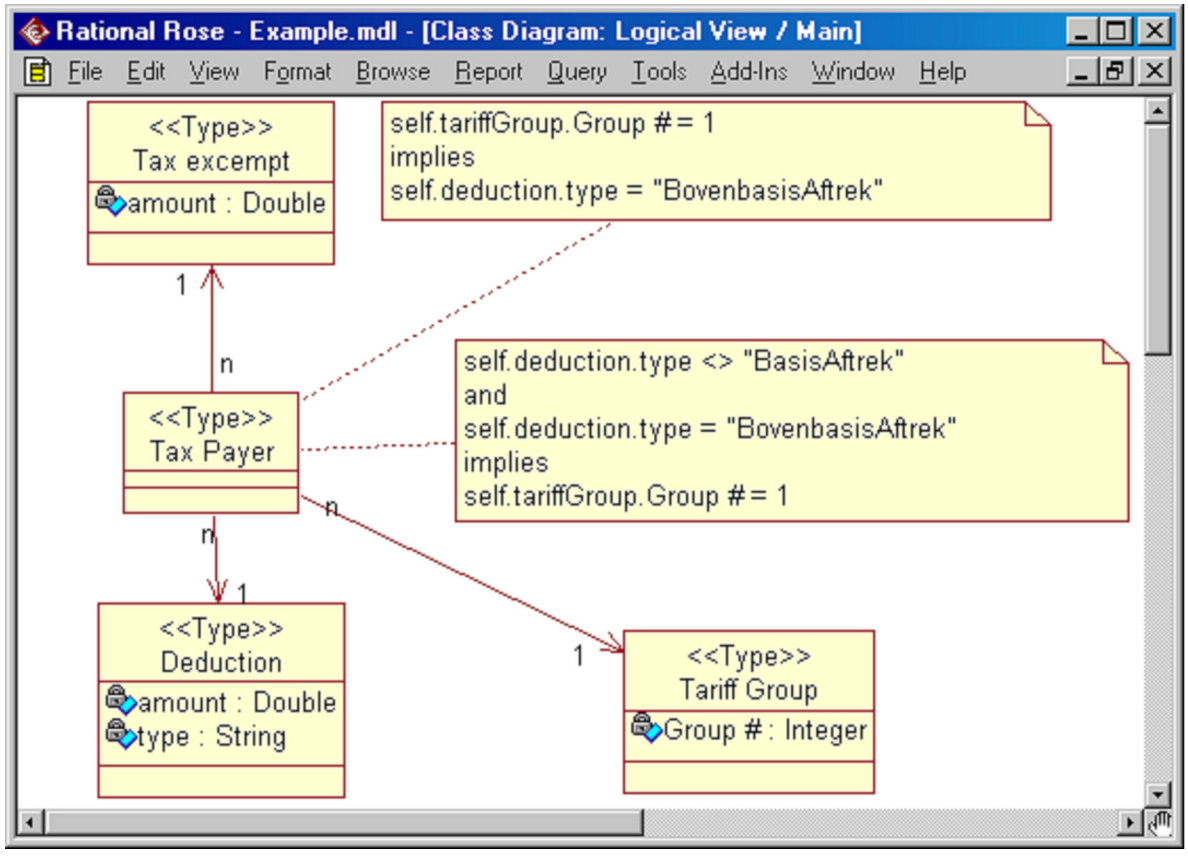

Figure 4: POWER Model

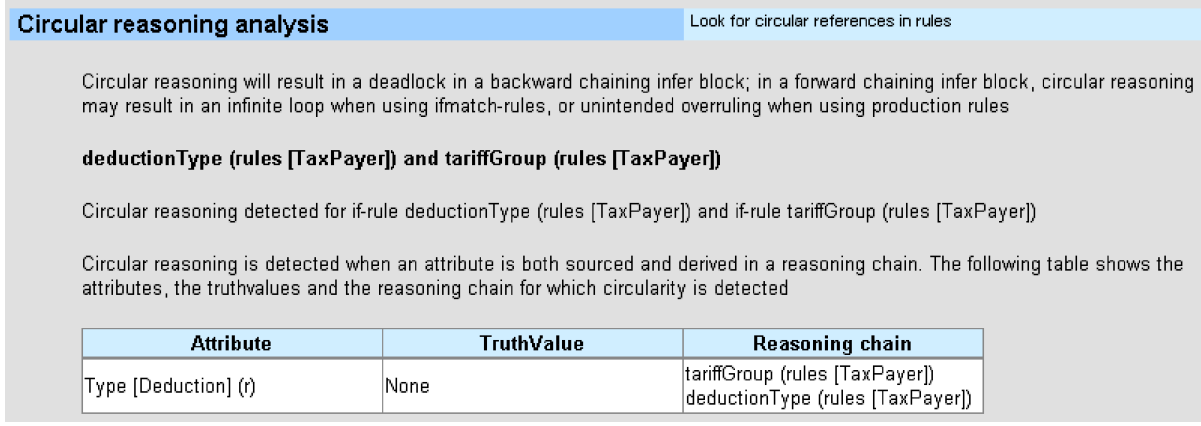

Figure 5: VALENS Report

Exploitation of the verification results therefore requires effective communication with the legislation experts, since they have to be able to adapt the legislation on the basis of the reported anomalies. Due to the fact that not the legal source, nor the conceptual model, but the knowledge component, is verified, the results of the verification process might be non-intuitive or, at least, difficult to understand by the legislation drafter.

5 The Challenge of Communication

Perhaps even more complex than the verification process itself is the communication of the verification results with domain experts. In the legal context of the POWER-program these experts are often people with a legal background and these people are not familiar with the (formal) representations used by knowledge engineers and information scientists. Furthermore people with a legal background are used to focus on individual cases while knowledge engineers and information scientists focus on the invariance between sets of cases.

The representations of the verification process' results need serious attention. Domain experts (including legislation drafters) have to decide what the impact of these results is in terms of redesign of legislation or chance of legal interpretation. It is obvious that the domain experts can only do this task properly if the representations used are fully understood.

In an attempt to get a better understanding of what representation formats are adequate in communicating anomalies in legal sources to domain experts we set up a small empirical study.

We tested the understandability of five different representation formalisms:

1 Horn clauses
2 Decision trees
3 Production rules
4 Grammar
5 Decision tables

In the experiment the subjects had to find the right representation (formal model) that reflected a legal text and they had to select a legal text that reflected a given representation.

In the experiment 15 legal experts had to answer 10 multiple-choice questions. Five (one for each representation format) considered the selection of the right representation for a given legal source and five related to finding the right legal source for a given representation. Both the results of these tests and the subjects' preferred representation were scored.
The legislation drafters scored higher on both tests:

Legislation drafters	
Find legal text with representation	97%
Find representation with legal text	100%
Other domain experts	
Find legal text with representation	80%
Find representation with legal text	89%

We also found that some representation formats were easier to understand than others:

Representation format	Average	Legislation drafters (n=6)	Other experts (n=9)
Horn clauses	94%	100%	89%
Decision trees	94%	100%	89%
Production rules	94%	100%	89%
Grammars	92%	100%	83%
Decision table	82%	92	72%

A surprising result was that subjects preferred decision tables as representation format although this was the worst understood representation.

The results of this study indicate the importance of designing adequate representation formalism if we want to communicate anomalies in legal sources. Since referring to the original legal sources (i.e. linking the formal representation formalism as used in the POWER-method to the legal source, e.g. a section of the law) is included in the POWER-method, presenting the relevant legal source text is relatively easy.

6 Discussion and Conclusion

The POWER-method (in its current state) aims at improving both the quality of the legal sources and legal decision making at the operational level in PA's. Verification of the formal representations of the legal sources supports these quality improvement efforts. Verification is not a new phenomenon, but structurally applying verification in a juridical context is. VALENS was originally designed to support knowledge engineers but the verification process implemented in VALENS proved very useful in a legal context as well. We experienced that the use of VALENS in the POWER-projects improves the quality of the legal knowledge bases and helps to communicate anomalies in the legal sources to domain experts (members of the knowledge groups and legislation drafters).

Further research is still needed. In the near future we aim at further improving the representation of the verification process results. Another extension will be the inclusion of validation support in the POWER-suite. At this moment we are researching the possibilities to adopt the validation framework from Rainer Knauf [20] whose ideas are attractive because of the very pragmatic approach to knowledge validation. We expect that the research on how to present anomalies (deficits) in an intuitive way to a legislation drafter can be re-used for the presentation of validation results. Validation or testing the results of the reasoning processes that are reflected in the knowledge bases against the aims of the legislation drafters and policy makers cannot be done fully automated. We will therefore incorporate testing facilities and simulation functionality (that is needed to check if the law has the intended effects).

Many people depend on the PA's legal decisions. Information science and knowledge engineering provide the means to improve the quality of legal decision-making and therefore need serious attention. The application of insights from these disciplines in the processes of legislation drafting and law enforcement will consequently lead to a more effective government.

References

[1] Voermans, Computer-assisted legislative drafting in the Netherlands: the LEDA-system, A National Conference on Legislative Drafting in the Global Village 2000

[2] Preece, Shingal, 1994, Foundation and Application of Knowledge Base Verification, International Journal of Intelligent Systems, 9, 683 – 701

[3] J. Vanthienen, C. Mues, G. Wets, 1997, Inter-Tabular Verification in an Interactive Environment, Proceedings Eurovav 97, 155 – 165

[4] Van Engers, T.M., Glassée, E., 2001, Facilitating the Legislation Process using a shared Conceptual Model. In: IEEE Intelligent Systems January 2001, p. 50- 57.

[5] Van Engers, T.M., Kordelaar, P., 1998, POWER: Programme for an Ontology based Working Environment for modeling and use of Regulations and legislation, Proceedings of the ISMICK'99.ISBN 2-913-923-02-X.

[6] Van Engers, T.M., Kordelaar, P.J.M., Den Hartog, J., Glassée,E., 2000, POWER: Programme for an Ontology based Working Environment for modeling and use of Regulations and legislation. In Proceeding of the 11th workshop on Databases and Expert Systems Applications (IEEE) eds. Tjoa, Wagner and Al-Zobaidie, Greenwich London, ISBN 0-7695-0680-1, pp. 327-334.

[7] Souza D.F. and Wills A.C., 1999, Objects, components and frameworks with UML; the Catalysis approach, Addison - Wesley, ISBN 0-201-31012-0

[8] Kordelaar, P.J.M., Betere wetten met kennissystemen, Universiteit Twente, Enschede 1996 (diss).

[9] den Haan, N, Automated Legal Reasoning, University of Amsterdam, Amsterdam, 1996 (diss)

[10] Svensson, J.S., Kennisgebaseerde micro simulatie, Universiteit Twente, Enschede, 1993 (diss.)

[11] A. Preece, 1991, Methods for Verifying Expert System Knowledge Bases.

[12] Robert T. Plant, 1995, Tools for Validation & Verification of Knowledge-Based Systems 1985 – 1995 References, *Internet Source*

[13] M. Suwa, A.C. Scott, E.H. Shortliffe, 1982, An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System, AI Magazine, Vol. 3, Nr. 4

[14] W.A. Perkins, T.J. Laffey, D. Pecora, T.Nguyen, 1989, Knowledge Base Verification, Topics in Expert System Design, 353 – 376

[15] C.L. Chang, J.B. Combs, R.A. Stacowits, 1990, A Report on the Expert Systems Validation Associate (EVA), Expert Systems with Applications, Vol. 1, Nr. 3, 217 – 230

[16] F.V.Harmelen, 1995, Structure Preserving Specification Languages for Knowledge Based Systems, International Journal of Human Computer Studies, Vol. 44, 187-212

[17] Rym Nouira, Jean-Marc Fouet, 1996, A Knowledge Based Tool for the Incremental Construction, Validation and Refinement of Large Knowledge Bases, Workshop Proceedings ECAI96

[18] R. Gerrits, S. Spreeuwenberg 1999, A Knowledge Based Tool to Validate and Verify an Aion Knowledge Base, Validation and Verification of Knowledge Based Systems, Theory, Tools and Practice, 67 – 78, ISBN 0-7923-8645-0

[19] F. Bouali, S. Loiseau, M.C. Rousset, 1997, Revision of Rule Bases, Proceedings Eurovav 97, 193 – 203

[20] Rainer Knauf, Generation of a minimal set of test cases that is functionally equivalent to an exhaustive set, for use in knowledge - based system verification, Proceedings of the Florida Artificial Intelligence Research Symposium (FLAIRS-96), pp. 280 - 284