UvA-DARE (Digital Academic Repository)

Facilitating the Legislation Process Using a Shared Conceptual Model

van Engers, T.M.; Glassée, E.

Publication date
2001

Published in
IEEE Intelligent Systems

[Link to publication](#)

Citation for published version (APA):

van Engers, T. M., & Glassée, E. (2001). Facilitating the Legislation Process Using a Shared Conceptual Model. *IEEE Intelligent Systems*, 16, 50-58.

General rights

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

Facilitating the Legislation Process Using a Shared Conceptual Model

Tom M. van Engers, *Ministry of Finance, Belastingdienst*
Erwin Glassée, *Application Engineers*

The Dutch Tax and Customs Administration (Belastingdienst in Dutch) conducts a research program called Power (program for an ontology-based working environment for rules and regulations), which develops methods that support the systematic translation of new legislation into the DTCA's processes. This program combines two

frequently separated approaches to knowledge management: the stock, or codification, approach and the flow, or organizational, approach.

Preliminary results indicate the Power program's methods—although still under construction—already have added value. These methods help improve new legislation's quality and codify the knowledge used in translation processes, in which Power transforms legislation and regulations into procedures, computer programs, and other designs. (We define legislation quality as law enforceability, an absence of anomalies, and effectiveness in obtaining intended effects.) The transformation processes developed thus far provide a focal point for the experts (in many cases working in knowledge groups) involved in these processes. Working together on the Power processes that apply Power methods, these experts develop a common mental model (and vocabulary). Thus, these experts' knowledge and assumptions are made explicit, creating a corporate knowledge corpus. This corpus's form makes its content traceable and certifiable. The Power method also helps knowledge workers involved in implementing new legislation to develop a shared mental model of the implementation process itself.

Overview

The Power research program^{1,2} aims to develop a method and supporting tools for the whole chain of processes from drafting the legislation to executing the law (see the "Improving the Legislation Process" sidebar). The Power process consists of five iterative subprocesses:

- translating legislation and regulations to conceptual models, including model completion by expert knowledge elicitation (see Figure 1);
- refactoring conceptual models for coherency;
- verifying conceptual models, including detecting incompleteness and identifying missing legislation and regulations;
- generating knowledge-based components for application frameworks, which creates knowledge-based systems for implementing law enforcement; and
- testing and validating knowledge components, including expert involvement to certify knowledge components.

The order of the subprocesses can depend on the route that specific legal drafting or law enforcement implementation projects take. Whatever route chosen, the legislation, conceptual models, and knowledge components will eventually establish traceable refinement relationships.

Power uses a proper extension of the information and communications technology (ICT) industry-standard Unified Modeling Language, largely based on version 1.3 but including some of version 2.0's proposed features particular to Catalysis, a development method for component-based systems that uses UML as its notation.³ The Object Constraint Language (OCL), which expresses constraints that apply to the model, is part of UML 1.3.

However, UML provides a very wide expression capability to accommodate several systems' development methods—for example, the Rational

The Power program combines a knowledge-capitalization–knowledge-codification approach with an organization dynamics approach. Initial results indicate this method will help improve the quality of law enforcement and decrease the time needed for implementing legislation and regulation changes.

Unified Process (RUP), Catalysis, and Select/Perspective. To obtain a well-defined translation process, we must describe both the Power method's modeling conventions, which delimit a UML subset and the UML extensions in the method and the translation process. Power uses a translation process that is a somewhat adapted and more rigidly described form of Catalysis business modeling. The differences arise from the

- deliberate absence of business process information in legislation,
- enhanced uniformity of conceptual models, and
- legislative domain's particular features—mainly the explicit references whereby a legislation structure block (for example, a chapter or article) refers to related structure blocks—combined with the need for an independent model of each structure block for traceability purposes.

The UML's structuring concept is *packages*, which are the container concepts that delimit a part of a model. Packages have a hierarchical containment structure, and each nested package can reference the model items (such as other packages) that it contains. Other model items can only be referenced if the model item's containing package is imported in the referring package; the nesting package is always implicitly imported.

Packages

Power translates each legislation structure block into a package. The names and kinds of structure blocks depend on legislative culture, which relates to the authoring legislative body or country in a certain time period. Because conceptual models and the translation process must be applied regardless of legislative culture, both the conceptual model formalism and the translation process description must remain culture-free. The packages' hierarchical nesting reflects the hierarchical relationship of legislation structure blocks, and Power chooses an appropriate name for each package. To preserve traceability, we don't import any related packages. So, Power must express each package with the package's and the nesting packages' model items. This translation of structure provides a clean, traceable way to transition from a semiformal legislation to a formal specification, while exploiting the legislative

Figure 1. The Power modeling process. Power translates legal sources into formal specifications (the conceptual model) and uses expert knowledge to extend, complete, and validate conceptual models. It can refactor these models before using them in knowledge-based systems, for example.

domain's specific features, as compared to general business modeling.

To preserve nonimport modeling, we extended UML with the concept of a *package reference*, which models a structure block's explicit reference to another structure block. Because we cannot import the package that corresponds to the referenced structure block, Power translates the reference to a package reference model item. This lets us include legislation structural defects in a first conceptual model if, for example, we cannot determine which structure blocks the reference references.

(The verification process can automatically detect this type of defect for many references. During refactoring, we will construct new packages that import both translated packages and other refactored packages. Refactoring aims to integrate a set of coherent specification packages into a set of complete, consistent specification packages that Power can generate as a knowledge component. The package references contribute to calculating coherence heuristics.)

Types and attributes

Next, we use Catalysis business modeling to describe the domain expressed in any given structure block. We select the structure blocks to be translated economically, based on expert judgment, and exploit the package structure to maintain a record of each structure block's modeling status. A versioning system can be integrated with the modeling tool to achieve this. We can automatically track project status, as well as the *version traceability* between legislation and a conceptual model. Version traceability lets us translate the model as soon as a legislation concept version becomes avail-

able, picking up changes as legislative drafting continues. This causes the modeling task to overlap legislative drafting and approval, letting legislators correct reported defects before the legislation is approved and reducing total project duration.

Catalysis models the business domain with types and attributes only, which are only a small UML subset. Thus, the domain model will describe a minimal set of specifications, thereby avoiding design issues. Specifications let us identify exactly whether a given design conforms, without excluding a priori other potential designs that might conform. To preserve minimal modeling, we specify only the information required to conduct the legislative reasoning, using object types and attributes. Associations are considered inverse pairs of attributes that relate object types and, hence, are a special case of attributes.

Types and attributes do not model stored data; they model information that must be derived from collected data. That is, any design of collected data must prove that this information can be calculated using the data. This proof is called refinement of the domain model's abstract information into concrete or realized data. We define design as the creative process of coming up with a well-structured model that optimizes technological constraints, given a specification. For example, legislation will often refer to a person's (type) age (attribute) in its reasoning even though collecting a birthdate and the applicable date of the legislative reasoning makes more technological sense. The age can be mapped to the difference of both, rounded down to whole years, proving that the design conforms.

Figure 2. Article 8.9 of the revised Dutch Income Tax Law: (a) the original Dutch text (references are italicized, terms are in bold, and relationships are underlined) and (b) an unofficial English translation that is as literal as possible to reflect the nuances of the Dutch text.

Figure 3. Article 8.9 comprises three paragraphs modeled by three packages. The first paragraph refers to article 8.8 with an absolute package reference.

Invariants

Power completes translation by capturing the legislative reasoning with static invariants, relating the package's types and attri-

butes with first-order predicate logic. Power expresses invariants using OCL. To work with UML extensions, we have extended OCL with logical operators for package ref-

erences. Static invariants preserve the model's independence from a task model. In practice, Power will develop the task model and the conceptual model simultaneously. The task model will determine the set of structure blocks that the model requires, but the conceptual models will also influence the tasks to be designed in executing the law in governmental agencies. Often, several tasks will deal with different aspects of the same conceptual model.

Patterns

Modelers typically have no legal background. So, they must check often with legal experts, who are familiar with the legislative culture, to validate the formal interpretation of the model's legal structure blocks. During this in-translation validation, the modelers attempt to detect standard interpretations in the legislative culture, identifiable by typical legal phrasing. They then define these standard interpretations and typical legal phrasings, called *translation patterns*, with the corresponding formal models.

Features

The in-translation validation with legal experts also uncovers semantic anomalies in the legislation.⁴ Semantic anomalies can occur if the rules used in the legislation or regulations lack clarity. The Power translation processes report detected anomalies to legislators.

Traceability is an important feature of the Power method because in legal domains we must be able to point to our knowledge sources at execution (to justify the systems' decisions). If we preserve the refinement between the original legal sources (for example, a specific article in a specific law or a part of case law), we can also make impact analyses when we change existing legislation. We refer to these sources when communicating with legislation drafters, who are likely to understand legal texts better than formal representations (for example, when validating models or reporting anomalies).

We can generate the conceptual model containing a specific legal domain's formal specifications into a knowledge component for a modeled task, which computers can execute. These knowledge components form the basis for a test against syntactic requirements—that is, we can also use the verification process^{4,5} and the knowledge components to simulate the effects of legislation and regulations for a certain population.⁶

Figure 4. Article 8.9 of the Dutch Income Tax Law: (a) The first paragraph. Power models all concepts as UML types, the relationships as associations, and each “rule” as a type exception invariant. The phrase *typeException* shown in the invariant is a stereotype that indicates how to treat the invariant during refactoring. (b) The second paragraph. (c) The third paragraph. This model has an example of an age attribute, demonstrating the modeling of concepts used in legal reasoning as opposed to designing technically optimal data models. This model also shows the applies operation, which is added to the package reference model item as an extension to be used in OCL.

Formal specification verification will expose missing specifications and inconsistencies. We can translate additional specifications from executive measures and resolve inconsistencies through executive decisions made by the authorized expert group. Refactoring the specifications with these newly added specifications results in a more consistent and complete conceptual model. Eventually, the verification process demonstrates the consistency and completeness for the task at hand. The design of the enterprise software systems for integrating the task into the executive business processes could be well under way by that time, to achieve the throughput time-to-execution required.

A simple example: Dutch Income Tax Law

We'll illustrate Power at work, using an example from the revised Dutch Income Tax Law, which will become effective for the fiscal year 2001. Figure 2 gives the Dutch original text and English translation of Artikel 8.9.

Packages

We'll use the Dutch version (see Figure 2a) to show how the translation into a conceptual model takes place. One of the features of the legislation is the use of structure in the text. Furthermore, legislation often contains absolute or relative references. In this example, we find two references: *artikel* 8.8, referencing an absolute part of the law, and a relative reference *dit artikel* (this article), referencing the article itself.

Four packages (one for the whole article and three nested packages for the paragraphs) model Artikel 8.9's structure. The first paragraph will contain an absolute package reference to Artikel 8.8 (see Figure 3).

Types, attributes, and invariants

After thus establishing the law's structure, we continue by detecting the terms (the con-

Figure 5. Refactoring two conceptual models representing a part of the domain by connecting the concepts (types) in the models to the concept definition. Modeling this way, we end up with a network of interconnected, but still traceable, models. This network's scope depends on the project's task.

cepts) and the relationships between these terms in the legal text subject to our analysis. This is typically done article by article, paragraph by paragraph. In the first paragraph of Artikel 8.9, the terms are in bold while the relationships are underlined (see Figure 2a).

Power models the terms in UML types and represents the relationships as UML associations. The “rules” that an article’s paragraphs describe are modeled as OCL invariants (see Figure 4a). The second and third paragraph are modeled as well (see Figures 4b and 4c).

According to Artikel 8.9, we must calculate the combined levy reduction’s amount before and after the first paragraph is applied. If we model this article, we lack the definition of many concepts such as *Algemene Heffingskorting* (general levy reduction), the *Arbeidskorting* (labor reduction), *Partner*, *Kalenderjaar* (calendar year), *in belangrijke mate onderhouden* (largely maintained), and *ouder* (parent). We also lack some attributes’ definitions, which relate these concepts. This

is due to the nonimport modeling; consequently, we will have incomplete models for most tasks. So, we detect this incompleteness and look for knowledge to add from other legal sources (other articles, specifically those that contain these concepts’ definitions or, failing that, an expert’s interpretation).

In the refactoring process, we produce an integrated model that comprises the individual partial models, by connecting concept references and concept definitions (see Figure 5).

Problems found in the model

The model unveils defects:

- In the first paragraph, nearly the same concepts are added twice. Is this sum what’s intended? Could a clearer phrasing be used—for example, “amounts minimum to,” the inverse of a translation pattern that the second paragraph uses, which contains the phrase “amounts maximum to” (result of in-translation validation)?

- To whom does the term “his” refer in the second paragraph—the partner or the taxpayer? Wouldn’t it be more understandable if this were explicit (a result of in-translation validation)?
- Does the third paragraph render itself inapplicable? Shouldn’t it read “The first and second paragraphs of this article shall not apply....”?

We can resolve some of these inconsistencies by selecting and translating further specifications, originating from juridical conventions, expert groups, or, in the most expensive case, court rulings. In the absence of court rulings, executive measures might be required to resolve the ambiguities. Detecting these potential problems early could cut government costs and enhance taxpayer compliance. In all cases, we can justify the decisions that the software system makes using the authority of the translated specifications that contributed to the refactorings that produced the decision.

Validation and legislation verification

Anomalies in legislation (for example, incompleteness and inconsistencies) impact law enforcement and enforcement organizations. Instruments to detect anomalies in legislation can thus help avoid legislation abuse and combat fraud. The Power method has an automated prototype tool for detecting anomalies in an early stage of the legislation process. By applying this instrument, we can assist legislators and consequently improve the quality of law.

Definitions of anomalies

We developed an anomaly framework (see Table 1) based on published research.⁷ The Power anomaly framework classifies the different anomaly types, including semantic and syntactic, but because pragmatic anomalies require further research, we did not include them in this discussion.

The approach

The current prototype can detect syntactical anomalies by reasoning about (knowledge) rules and inference frames. The anomalies found in the analyzed models are potential defects (the legislator might, for example, intentionally have redundant rules).

The inference frames, derived from the task model, contain knowledge about the inference process, such as goal attributes and

Table 1. The Power anomaly framework.

Syntactic anomalies	Operationalizations
1. Incompleteness	1. Dead-end rules, missing rules, and unreachable rules
2. Discrepancy	2. Conflicting rules
3. Circularity	3. Circular rules
4. Redundancy	4. Redundant rules, subsumed rules, and unnecessary IF conditions
Semantic anomalies	Operationalizations
1. Synonyms	1. Occurrence of semantic redundancy
2. Homonyms	2. Occurrence of ambiguities
3. Vagueness	3. Vague terms, evaluative-open terms, and open-texture terms
Pragmatic anomalies	
Requires further research	

start attributes. With this knowledge, the prototype can create a topology of possible decision paths. The prototype evaluates these paths to detect potential anomalies. A potential redundancy, for example, occurs when two paths share at least one start attribute and one goal attribute (see Figure 6). A legislator must determine if such a potential defect is really a defect in the legislation.

Experiences thus far

Our assessment of modeled parts of legislation indicates that the Power prototype can detect certain legislation anomalies. Additionally, empirical research proved that the prototype is also useful for verifying legislation models. The tool has some shortcomings, of which the most important is its inability to support validation (by testing a case against the rules in the knowledge components). We will therefore expand the prototype's functionality with simulation abilities.

Implementation: Assessment of retirement letters

Director-major stockholders of companies have a great deal of freedom in formulating their pension regulations. Sometimes they explore the boundaries of legislation because they can deduct the contributions made to their retirement funds from the company's profit taxes. Owing to new legislation, the DTCA expects that, within five years, numerous major stockholders will ask the DTCA for a preliminary judgment about whether their pension regulation falls within the legislation's framework. A great deal of knowledge about the wage tax law and regulations containing pension-related topics is needed when assessing pension letters. However, existing knowledge is insufficient, compared to the volume of judgments expected. So, the DTCA has decided to create the *pension assessment system* (PAS), a knowledge-based system that supports the dissemination of expert knowledge about pensions, developed using the Power method.

Through a user-friendly wizard (see Figure 7), a DTCA employee can insert a director-major stockholder's pension regulation into the system. The knowledge component that PAS produces from the pension legislation in the system then assesses the pension regulation's acceptability. On those points where the regulation crosses the boundaries laid down in the legislation, the system makes a remark. These remarks also contain references to the (parts of) legislation upon which it bases its

```

HeeftTegenWoordigeArbeid
If GetHeeftArbeidsKorting() <> Null and
    BDH3InkomenUitWerkEnWoning.GetWinstUitOnderneming(Current) <> Null and
    BDH3InkomenUitWerkEnWoning.GetLoon(Current) <> Null and
    BDH3InkomenUitWerkEnWoning.GetResultaatUitWerkzaamheden(Current) <> Null
Then
    SetHeeftArbeidsKorting(
        GetHeeftArbeidsKorting() <> Null and
        ((BDH3InkomenUitWerkEnWoning.GetWinstUitOnderneming(Current) +
          BDH3InkomenUitWerkEnWoning.GetLoon(Current) +
          BDH3InkomenUitWerkEnWoning.GetResultaatUitWerkzaamheden(Current)
          ) > 6950
        )
    )

```

Figure 6. An example of a rule (from the Dutch Income Tax Law) containing a circularity.

Figure 7. A screenshot of the pension assessment system.

remarks. The remarks help the employee write an adequate response to the taxpayer. Two DTCA departments tested PAS, and it has been consequently rolled out in all DTCA units. The first results show promise.

Embedding knowledge components in a process environment

In most law enforcement organizations, many departments and many people contribute to the legislation implementation processes (see Figure 8). To design these processes to guarantee both quality and implementation speed pose a challenge.

Figure 8. The government and its environment.

Improving the Legislation Process

Drafting, implementing, or changing legislation tends to consume time, energy, and money. The many interconnected processes and the numerous people involved make it very vulnerable to errors. Varying interests must be aligned and communication difficulties (due to differences in technical jargon) must be overcome. The knowledge and experience needed to create new laws and to specify, design, and implement procedures and systems in legislative domains is very scarce. However, getting the right knowledge at the right time and place is critical to create effective legislation.

The Power program has its background in the DTCA's earlier experiences with knowledge-based systems. Since 1989, the DTCA has built knowledge-based systems for many purposes, especially supporting compliance risk assessment and taxpayer case diagnosis and selection. These knowledge-based systems proved to be useful in themselves, but perhaps even more important were the side effects of creating them. We found that the specification process, which involved experts from different disciplines and backgrounds, helped make explicit knowledge that would otherwise have remained implicit. Furthermore, the experts could specify knowledge in a way that facilitated verification of its validity. In addition to improving the efficiency of constituency treatment in its operational units (of the DTCA), the knowledge-based systems served primarily as a dissemination vehicle, letting the DTCA better use its experts' knowledge, while improving the quality of law enforcement in its operational units. (We define quality of law enforcement as the constituency's satisfaction

with the adoption of the principles of equality before law, predictability of law enforcement, and proper use of authority by law enforcement agencies.)

The Power program elaborates on these earlier experiences. The Power method (see Figure A), based on conceptual modeling of legislation and regulations into formal legal specifications, plays a central role in the Power research program. We

Figure A. The Power program aims to support the whole chain of processes from legislation drafting to implementing law enforcement.

The Power system's methods and infrastructure support continuous improvement of knowledge productivity. Power provides an architecture that facilitates implementation of the processes mentioned, and we have created a platform (and communication vehicle) for different stakeholders in these processes.

We had the opportunity to model the concept version of the new Dutch Income Tax Law—an ambitious piece of legislation, the first major law revision since 1964. Because almost nothing of the old income tax law remains intact, we were asked by our steering group (in which the legislation drafting department is represented) to look for anomalies (and to create a basis for further implementation and supporting infrastructure). We modeled the income tax working closely with knowledge-group experts who deal with

income tax problems. We found more than 150 anomalies that these knowledge groups had not detected. We reported the anomalies to the drafters, who repaired them.

The feedback process's effectiveness depends on representation. So, we have conducted a cognitive ergonomic study—which aims to establish the measure of how far people can understand the representation—on law representations that promote communication between legislative and IT experts.

When we designed a support tool for assessing pension letters, we worked closely with knowledge-group experts and a legislation drafter, who was involved with drafting the new wage tax law (which contains pension aspects). These experts had different viewpoints on many topics before we started the project, but during the design process, we developed a common mental model. The

experts became enthusiastic about this new way of working. They are involved in and committed to the implementation process, and the retirement-assessment application has become a regular part of their knowledge-distribution instruments.

Furthermore, we have tested several modeling tools that support the developed methods. We are still trying to improve both the method and modeling speed. While modeling legislation, we continue to develop the Power legislation ontology. We also developed an architecture that lets us embed the knowledge components in future process models of knowledge-intensive processes and tasks.

Thus far, we have observed that legislation drafters, experts from the knowledge groups, and people from our automation department are very enthusiastic about our approach.

want to produce legislation representations with which computers can reason.

A set of coherent specifications can be delivered as a *knowledge component*, a coherent package of software artifacts that can be independently developed and delivered as a unit and that defines interfaces that let it be composed with other components to provide and use services.¹ Conceptual models, combined with task models, can be used in

- *Anomaly detection.* Conceptual models can be verified with a knowledge verification tool developed in the Power program. (We define verification as the detection of potential defects [anomalies] in formally represented knowledge ["building the system right"], as opposed to validation, defined as establishing the correctness of formally represented knowledge with respect to expert judgment ["building the right system"]). This way legislation and regulations can be checked automatically for potential defects such as incompleteness, redundancy, contradictions, and reasoning loops.
- *Simulation of legislation effects.* Using the knowledge components on appropriate data can simulate the consequences of legislation and regulations. The simulation can be micro-, meso-, or macro-level (that is, for individuals, groups [social, ethnic, and so on], or the entire constituency). When insufficient real data exists, we can automatically construct fictitious cases, assuming, for example, probability distributions of data or well-chosen experimental data. We can also apply this capability to test and validate the knowledge.
- *Data modeling.* We can derive the data necessary to apply these knowledge components (for example, income, household situation, and so on) from the conceptual models of legislation and regulations. This information inventory describes the minimal information that must be inquired, retrieved, or derived to enforce the law. The information inventory can support the development of information collection processes, as well as the evaluation of law enforceability with respect to reliability and collection of required

information.

- *Specification (of knowledge-based systems).* Knowledge-based systems can support the application of legislation and regulations to specific cases. The knowledge components contain the reasoning necessary in the enforcement organizations' operations. These knowledge components form the core of such knowledge-based systems. Other components are data storage, data access, the graphical user interface, security, workflow, and so on. Examples of this type of application are software that supports complex reasoning tasks such as calculating the income tax due and assessing pension letters to detect if any norms are violated.

Although the Power method translates legislation into formal specifications, we also elicit knowledge from experts. The knowledge elicitation occurs at a systematically chosen time and serves only to complete and validate the models. The Power approach is typical in the knowledge management arena. We capture the knowledge about the legislation in a model, which we use to improve the knowledge's quality and disseminate it for further use.

The DTCA strives to achieve following goals with the Power research program:

- improve legislation quality (we define legislation quality as absence of anomalies, law enforceability, and effectiveness in obtaining the intended results);
- streamline the legislative drafting effort;
- improve law enforcement quality ;
- reduce law enforcement implementation time; and
- reduce the law enforcement implementation effort.

Reference

1. D.F. D'Souza and A.C. Wills, *Objects, Components and Frameworks with UML: The Catalysis Approach*, Addison-Wesley, Reading, Mass., 1999.

Further developments

Our initial research with Power has delivered useful tools and several models that support new-legislation implementation. We had already observed that working with different experts from different knowledge groups using the Power methods increased the homogeneity of the conceptual models of the legislation in question.

Many problems, however, remain. Legislation consists of definitions, normative descriptions of situations, and, in some occasions, directives for processes (for example, the inspector should send a confirmation letter within two weeks). The last is an especially complicating factor when bridging the gap between legislation and design. Public administrations have a great amount of freedom for their organizational and process design.

The law-based Power models, therefore,

should fit in common process-organization and systems-design methods. Our current research activities take this process aspect into account. We are working to implement an architecture that consists of several layers:

- legislative model and process model,
- functional (a refactored conceptual model integrated with the task model),
- technical (knowledge components and application frameworks, and a connector architecture),
- realization (generated knowledge component code or developed framework code), and
- implementation (deployed knowledge components).

We have already applied the last four levels on a small scale in the DTCA's automation

department. We plan to broaden the architecture to a full-scale application. The DTCA developed the process layer in cooperation with the Telematics Institute (a research school guided by the University Twente) in their Testbed research program. Our challenge is to integrate Power and Testbed and connect the output of both to the functional layer that is input for our software factory (Belastingdienst Automatiserings Centrum).

We think that the Power method will work in many other governmental situations and perhaps even in nongovernmental organizations where regulations play an important role (for example, process industries, insurance companies, and so on).

IEEE Internet Computing reports emerging tools, technologies, and applications implemented through the Internet to support a worldwide computing environment.

In 2001, we'll look at

- Embedded systems
- Virtual markets
- Internet engineering for medical applications
- Distributed data storage
- Web server scaling
- Personalization

... and more!

IEEE Internet Computing

computer.org/internet/

IEEE
COMPUTER
SOCIETY

We noticed that legislation drafters (as well as knowledge-group representatives) often express themselves procedurally and, consequently, often create unintended restrictions on the design of procedures, organization, and systems. Information technologists and knowledge engineers therefore prefer declarative specifications, which allow a great amount of implementation freedom. Although we already succeeded in ending discussions if declarative specifications are more efficient, much research is needed, especially on representation formalisms suitable for both legal experts and information technologists.

Many knowledge management approaches focus on stock approaches (for example, the creation of digital libraries and Internet and intranet technologies, or knowledge-based systems) or flow approaches (for example, HRM measures aimed at increasing organizational learning and knowledge sharing). We've found that with the right combination of these approaches, we can achieve tremendous results, and knowledge management becomes more than just an interesting research field—it pays. ■

References

1. T.M. Van Engers and P. Kordelaar, "Power: Programme for an Ontology-Based Working Environment for Modeling and Use of Regulations and Legislation," *Proc. Sixth Int'l Symp. Management of Industrial and Corporate Knowledge (ISMICK '99)*, 1999.
2. T.M. Van Engers et al., "Power: Programme for an Ontology-Based Working Environment for Modeling and Use of Regulations and Legislation," *Proc. 11th Workshop Database and Expert Systems Applications*, IEEE Press, Piscataway, N.J., 2000, pp. 327–334.
3. D.F. D'Souza and A.C. Wills, *Objects, Components and Frameworks with UML: The Catalysis Approach*, Addison-Wesley, Reading, Mass., 1999.
4. P.J.M. Kordelaar, *Betere Wetten met Kennissystemen*, doctoral dissertation, Universiteit Twente, Enschede, Netherlands, 1996 (in Dutch).
5. N. Den Haan, *Automated Legal Reasoning*, doctoral dissertation, Univ. of Amsterdam, 1996.
6. J.S. Svensson, *Kennisgebaseerde micro Simulatie*, doctoral dissertation, Universiteit Twente, Enschede, Netherlands, 1993 (in Dutch).
7. D. Kracht, C.N.J. Vey Mestdag, and J.S. Svensson, *Legal Knowledge-Based Systems: An Overview of Criteria for Validation and Practical Use*, Koninklijke Vermande BV, Lelystad, Netherlands, 1990.

The Authors

Tom M. van Engers

works for the Ministry of Finance in the Belastingdienst Center for Process and Product Development. Until recently, he was head of a research department that coordinates development of knowledge-based systems and research on the application of innovative technology. He is responsible for several research projects and advises several other projects in software engineering and systems design, data mining, and knowledge management. He is also Power's program manager. He has a university degree (Drs.) in artificial intelligence from the University of Utrecht. He is a member of the ACM. Contact him at Herman Gorterstraat 5, Postbox 18280, 3501 Utrecht, Netherlands; t.m.van.engers@acm.org.

Erwin Glassée

is a consultant for Application Engineers, is responsible for the development of the Power knowledge-modeling method, and is a certified Catalysis practitioner. His research interests include system development methods, formal modeling, and component-based systems. He has a master's degree in software engineering from the University of Leuven. Contact him at Herman Gorterstraat 5, Postbox 18280, 3501 Utrecht, Netherlands; erwin.glassee@applicationengineers.com.