Human-Readable and Machine-Readable Knowledge Bases Using Specialized Word Processors

Martin Molina
Departamento de Inteligencia Artificial
Facultad de Informática
Universidad Politécnica de Madrid, Spain
mmolina@fi.upm.es

Gemma Blasco
EADS CASA European Aeronautic
Defence and Space Company -
Construcciones Aeronáuticas S.A, Spain
gemma.blasco@eads.com

Abstract

The maintenance of knowledge bases is one of the crucial activities in the life cycle of knowledge systems. This paper describes an innovative approach to write complex and large knowledge bases using specialized word processors. According to this, a knowledge model is represented as a conventional document that is written following the standard operations of word processors. Following this approach, domain experts that are not familiar with computer languages could easier read and write complex knowledge models. In addition to that, the processor is able of interpreting the content of the document to automatically perform tasks of the knowledge model. The paper describes the basic characteristics of the document and its specialized word processor and presents our experience following this approach for a knowledge system in the domain of hydrology.

1. Introduction

The creation and maintenance of knowledge bases can be carried out with the help of specialized software tools that guide developers in writing the models and keeping its consistency. However, when the domain presents large and complex knowledge bases as it happens in modern knowledge systems, the existing approaches for these types of tools present certain deficiencies that receive even more importance when the system maintenance is intended to be done by domain experts that are no familiar with computer languages and knowledge engineering techniques. This is a case where the ultimate goal is that domain experts themselves should be able of creating and maintaining knowledge bases using their own language instead of artificial and complex symbolic languages. This general need has been already underlined as a significant unsolved problem. For example, the need of knowledge development tools usable by non-experts in knowledge engineering was formulated by AI researchers within the semantic web context as one of the challenges for the twenty-first century AI research [8].

As an answer to this, we describe here an innovative solution based on specialized word processors and standard documents. Word processors are software tools with well-known procedures for document manipulation. Our approach takes the advantage of the extended and common use of this type of tool and proposes to use it as a solution to write knowledge bases in the same way persons write on a document knowledge about certain problem solving tasks, following certain syntactic and organizational conventions. The proposal considers a special type of word processor that helps users in writing part of a document that represents the whole knowledge model. The tool supervises and constraints the changes made by the user according to a limited freedom for writing. The resulting knowledge base formulated with the help of this type of word processor is operational, i.e., able to be interpreted by the corresponding inference procedures to automatically solve the problems related to the tasks described in the document.

The main aim of this paper is to summarize the results of our research work in specialized word processors for knowledge modeling. This corresponds to a line of work in our research group about knowledge modeling tools in parallel with the development of real-world knowledge systems in civil engineering domains (hydrology, road transport, etc.).

The paper shows the general approach of using specialized word processors as knowledge modeling tools. Then, we describe the characteristics of the document that corresponds to the knowledge model. Then, the paper illustrates the proposal with the case of the KATS tool that we developed in the domain of hydrology following the general approach. Finally, we present a discussion that summarizes the contribution of the proposal and compares the approach to related work.

2. Specialized word processors as knowledge modeling tools

In the field of knowledge engineering, different types of software tools have been proposed to help developers in building and maintaining a complete operational knowledge base. These types of tools include: (1) *general tools for basic knowledge representations*, that correspond to the traditional shells for the development of expert systems with one knowledge representation, (2) *method-based knowledge modeling tools*, such as MOLE [6] for diagnosis systems with the cover-and-differentiate method, or SALT [11] for design systems with the propose-and-revise method, (3) *domain-based tools* that include prefixed knowledge about the domain in which the knowledge base is developed (e.g., SIRAH [1] for prediction tasks in hydrology), (4) *general knowledge modeling tools* that assist developers in the application of a modeling methodology (for example, KREST [10] and KSM [4]), and (5) *ontology management tools* such as Protégé-2000 [13] and Ontolingua [7].

When the previous approaches are directly applied to complex problem solving tasks certain difficulties may arise. We have experienced these difficulties in our group during the development of complex knowledge systems. For example, the SAIDA system was developed to assist operators for emergency management in the context of floods in river basins. This system includes different types of tasks (assessment, prediction and scheduling) with several types of knowledge representation formalisms (rules, frames, uncertainty with bayesian networks, logic clauses, temporal and spatial representation, etc.). SAIDA was initially developed for a particular river basin but it was required to be easily portable to other river basins. In this context, a knowledge modeling tool was needed with a language and a level of abstraction close the way the domain experts describe their knowledge in their particular professional area. According to our experience in this domain, the main difficulties that we found using the existing approaches were:

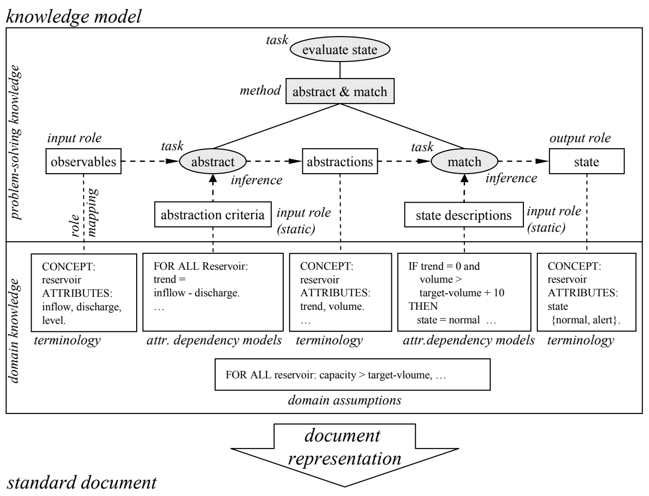
- *Excessive technicality*. Each particular knowledge base uses a specific symbolic language (rules, frames, functions, attributes with temporal references, bayesian networks, etc.). These languages follow a declarative approach that provides flexibility to accept changes but, still, domain experts perceive them as artificial programming languages.
- *Heterogeneous representation*. The diversity of knowledge bases with different representation languages offers a heterogeneous view of the model that makes more difficult to be learned by end users.
- *Limited perception*. The knowledge base was also viewed as a kind of complex and large data base that the user could change with the help of prefixed windows. However, since the existing approaches presented limited views about knowledge roles, it was difficult to anticipate the impact of changes in the knowledge base.
- *Non-standard operation procedures*. Most of the approaches for knowledge modeling tools follow certain operation procedures that are very specific and must be learned by domain experts with additional learning effort.

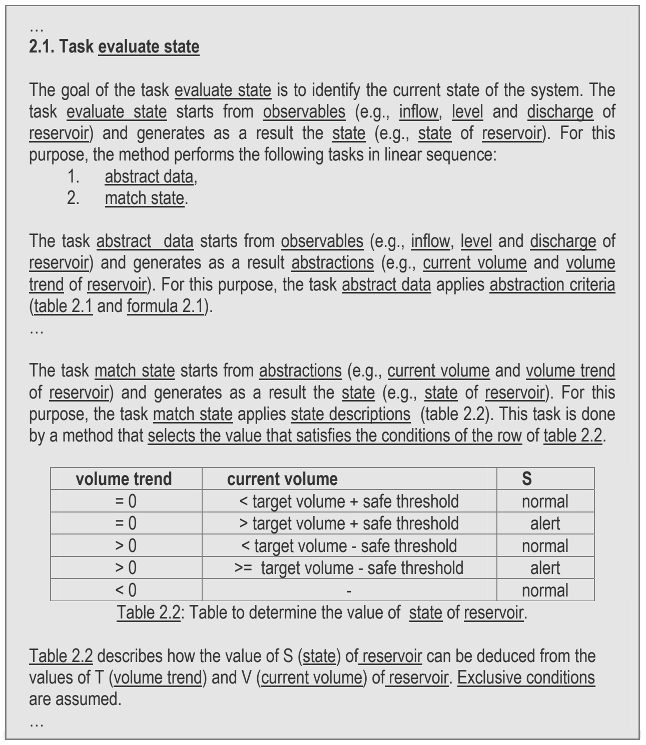
In order to cope with these problems we designed and developed an original solution based on specialized word processors for knowledge modeling. In general, word processors are well known tools with which a user can easily read and modify documents, changing margins, or adding, deleting, and relocating entire blocks of text, graphics and images. Currently, a word processor is one of the most widely used software tools with standard and well-known edition procedures.

However, the implementation of specialized word processors for knowledge modeling presents important technical difficulties (in the extreme case, it should require a complete solution for natural language interpretation, which it is still an unsolved problem). Thus, we have developed the first part of our research work in this area by focus on *method-specific word processors.* With this alternative, the word processor allows the user to read a complete knowledge model but only domain knowledge can be modified in the form of document-based representations (tables, formulas, text paragraphs, etc.). The word processor assumes prefixed problem-solving methods that are explicitly described in the document but cannot be modified by the user.

3. Standard documents for knowledge models

According to commonly extended knowledge engineering methodologies (e.g., [15]) a knowledge model can be formulated as a collection of tasks with a set of problem-solving methods and domain knowledge. For example (figure 1), a knowledge model in the domain of hydrology may include, among others, the task *evaluate state*, that interprets the measures of sensors in order to evaluate the severity of a flood problem. This task can be carried out with the method *abstract & match,* an adapted version of the heuristic classification method [3]. The figure also shows how this model could be described following a standard document representation (we call *standard document* a human-readable document as it is explained below). In the following, we describe the main components of a knowledge model and, then, how they could be represented following a document-based approach to be used by a specialized word processor.




…

2.1. Task evaluate state

The goal of the task evaluate state is to identify the current state of the system. The task evaluate state starts from observables (e.g., inflow, level and discharge of reservoir) and generates as a result the state (e.g., state of reservoir). For this purpose, the method performs the following tasks in linear sequence:

1. abstract data,
2. match state.

The task abstract data starts from observables (e.g., inflow, level and discharge of reservoir) and generates as a result abstractions (e.g., current volume and volume trend of reservoir). For this purpose, the task abstract data applies abstraction criteria (table 2.1 and formula 2.1).

…

The task match state starts from abstractions (e.g., current volume and volume trend of reservoir) and generates as a result the state (e.g., state of reservoir). For this purpose, the task match state applies state descriptions (table 2.2). This task is done by a method that selects the value that satisfies the conditions of the row of table 2.2.

volume trend	current volume	S
= 0	< target volume + safe threshold	normal
= 0	> target volume + safe threshold	alert
> 0	< target volume - safe threshold	normal
> 0	>= target volume - safe threshold	alert
< 0	-	normal

Table 2.2: Table to determine the value of state of reservoir.

Table 2.2 describes how the value of S (state) of reservoir can be deduced from the values of T (volume trend) and V (current volume) of reservoir. Exclusive conditions are assumed.

…

Figure 1: Partial example of a knowledge model and its representation as a standard document. The underlined words in the document are recognized by the word processor as specific elements of the model (tasks, concepts, attributes, etc.).

3.1. The components of a knowledge model

The knowledge model of figure 1 shows the main components of a knowledge model. In general, problem-solving knowledge is described by *task*s (*what to do*) and *methods (how to do it).* These components are used recursively developing a task-method hierarchy that shows a functional view of the model. Input and outputs of tasks are described with input and output *roles*. The problem solving knowledge also includes *method assumptions* to describe properties that must be satisfied by the domain knowledge for a correct operation of the method.

We consider that domain knowledge includes *terminologies* and *attribute dependency models*. A *terminology* expresses the basic terms of a domain in the form of concepts, attributes with values and relations between concepts (e.g., the concept reservoir with the attributes inflow, discharge, level, volume, etc.). Concepts can be organized into classes, subclasses and instances. We call an *attribute dependency model* to a model that relates values of certain attributes with the values of other attributes, based on a functional dependency (with a similar approach to [5]). For example, causal relations or rule bases are cases of attribute dependency models. As examples of this type of models, figure 1 shows part of the rules used to determine the value of the attribute state of reservoir or the arithmetic expression to compute the value of volume trend of reservoir. The domain model also includes *domain assumptions* that express certain properties that domain knowledge must accomplish (for example, physical laws or properties such as that the capacity of reservoir must be greater than its target-volume).

To relate the domain model to the problem solving model *role mappings* are used. This informs about the role that the domain knowledge plays in the reasoning process. For example, figure 1 shows that the *discharge* and the *level* of a *reservoir* play the role of *observables* in the inference *abstract.*

3.2. The document-based representation

In our approach, we have considered that the document that represents the knowledge model should satisfy the following two basic requirements:

- *Human-readable*, the content of the document must be comprehensible by readers who are not specialized in computer languages. Users should be able to manually perform the described tasks by reading the instructions of the document;
- *Machine-readable*, the content of the document must be able to be interpreted by a program that can perform automatically the tasks described in the document.

According to this, the knowledge model should be written as a conventional document with the following characteristics:

1) *Document-based representation*. The document is written with text paragraphs in natural language together with additional document-based representations (e.g., tables, figures, illustrations, etc.). This provides a well-known and uniform global

representation to describe problem-solving tasks. This characteristic assumes the absence of computer-oriented symbolic representations (rules, bayesian networks, logic clauses, etc.).

2) *Document-based organization*. The document is organized according to the usual document parts (chapters, sections, subsections, etc.). It is assumed that the document reading starts from the beginning of the text and develops a linear description of the content, following a top-down approach (from general to specific) with redundant complementary views such as summaries or glossaries. This follows a natural organization of the document with progressive understanding by introducing first general views that are later developed with details. The terminology used to organize the content of the document must be commonly understood by general users. This terminology should avoid abstract concepts of knowledge engineering or software engineering (agents, objects, processes, messages, etc.).

3) *Complete knowledge model*. The document describes the complete knowledge required to perform problem-solving tasks. This includes both domain and problem-solving knowledge describing inference procedures. This guarantees that the reader does not require additional knowledge sources to understand the content of the document. However, to be natural, the word processor may assume a minimum level of professional commonsense. For example, in the domain of hydrology, the document should not describe too obvious knowledge about temporal representation of physical magnitudes.

In order to represent domain knowledge, the document is written following document-based representations according to standard conventions. We have identified a collection of these representations that are familiar to general users that are not experts in computer languages. We have analyzed several problem-solving tasks in different domains to study the way experts use these representations to formulate domain knowledge. Based on the analysis, we have generalized a set of document-based representations (see figure 2) for each of type of knowledge that can automatically be translated into symbolic representations. These representations include text paragraphs, tables, formulas and illustrations such as schematic diagrams, graphs, 2D graphics, etc.

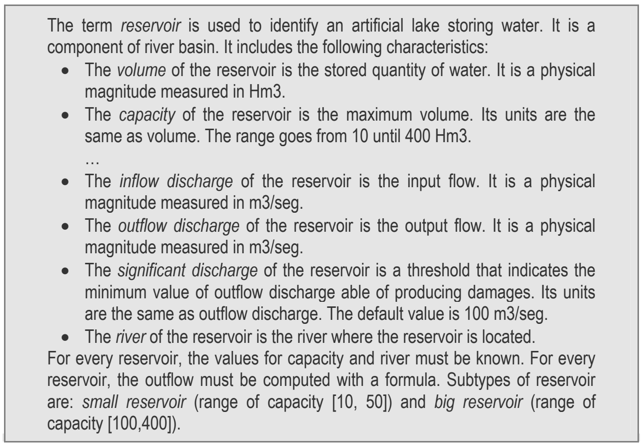
The combination of such representations provides appropriate expressiveness to represent knowledge in complex domains. We do not claim that this set of representations is complete (i.e., totally expressive for any kind of problem) but they cover a representative number of complex cases of real-world knowledge systems and it is general to be applied to different domains. In addition to that, it follows an open approach, i.e., this set of representations is open to include in the future new representations according to the specific needs of particular problems.

Type of knowledge	**Document-based represent-tation**	**Description**
Terminologies	*Structured paragraph for terminolo-gies*	Formatted text paragraph in natural language to describe explicitly or implicitly a concept, an attribute of a concept or a relation between concepts.
	Table of instances	Table to define particular instances of a class (e.g. the instances of reservoir in the domain of hydrology) with specific values for certain attributes (e.g., volume of each particular reservoir, etc.).
	Graph of relations	Graph with nodes that represent concepts and arcs that represent relations. This is useful to present a global image of the relations between a set of instances.
	Schematic diagram	An image with lines that associate components or parts of the picture to names of concepts (classes or instances) or attributes.
Attribute dependency models	*Decision table*	Table that includes conditional relations to be used in logical decisions. The decision table can adopt different shapes according to the amount of elements to be presented.
	Simple formula	Formula with arithmetic operators (+,-,/,*) and/or standard functions (sin(x), cos(x), etc.). The formula can be defined either for the attribute of a particular concept (specific) or for the attribute of a class (general).
	Iterative formula	Formula that is defined using an iteration (summatory Σ, productory Π, etc) on a set (or sets) of reference that is explicitly formulated in a table or implicitly defined in an attribute whose content is a list of values.
	Qualitative number line	A set of consecutive segments on a line with linguistic labels to describe a qualitative interpretation of a quantitative dimension.
	Table of causal relations	Table that shows a set of types of relations between variables that are causes and variables that are effects. This is especially useful to formulate bayesian causal models in the domain of hydrology.
	Others	Decision tree, graph of dynamic components (word bound graph, production chain, work-flow), 2D graphics, state transitions, etc.
Domain-specific assumptions	*Structured paragraph for assumptions*	Formatted text paragraph in natural language to describe specific domain assumptions such as the maximum number of instances, the type of dependency model associated to an attribute, etc.
Role mappings	*Structured paragraph for role mappings*	Formatted text paragraph in natural language to associate inference roles of problem-solving methods to domain knowledge.

Figure 2: Summary of document-based representations.

Figure 3 shows an example of text paragraph that describes a concept with attributes (this example is a direct translation from the original text in Spanish). It is formulated using specific natural language patterns with local editors that guide and constraint the way of writing these sentences. For this purpose, the word processor includes a grammar with a set of preformatted types of text paragraphs to explicitly define: (a) classes corresponding to categories of concepts, (b) attributes with allowed values and (c) relations between concepts.

In order to avoid repetitive non-natural sentences, different patterns can be used for the same type of component.


The term *reservoir* is used to identify an artificial lake storing water. It is a component of river basin. It includes the following characteristics:

- The *volume* of the reservoir is the stored quantity of water. It is a physical magnitude measured in Hm3.
- The *capacity* of the reservoir is the maximum volume. Its units are the same as volume. The range goes from 10 until 400 Hm3.

  …
- The *inflow discharge* of the reservoir is the input flow. It is a physical magnitude measured in m3/seg.
- The *outflow discharge* of the reservoir is the output flow. It is a physical magnitude measured in m3/seg.
- The *significant discharge* of the reservoir is a threshold that indicates the minimum value of outflow discharge able of producing damages. Its units are the same as outflow discharge. The default value is 100 m3/seg.
- The *river* of the reservoir is the river where the reservoir is located.

For every reservoir, the values for capacity and river must be known. For every reservoir, the outflow must be computed with a formula. Subtypes of reservoir are: *small reservoir* (range of capacity [10, 50]) and *big reservoir* (range of capacity [100,400]).

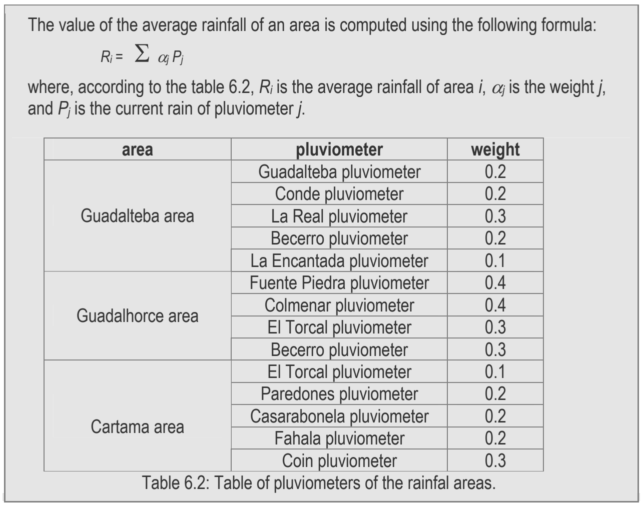
Figure 3: Example of text paragraph for basic terminology in the domain of hydrology.

The initial example of figure 1 also shows a case of a simple decision table in the hydrologic domain to determine the state of a reservoir using other values about volume trend and current volume. A decision table must be consistent with the terminology, i.e. only the defined attributes in the document can be used in columns and only allowed comparison operators can be used according to the allowed values for those attributes. Note that the reference of the table is described in the document with a text paragraph that shows details about its format and assumptions (e.g., exclusive conditions are assumed). In general, each particular component includes prefixed constraints formulated as natural language sentences that can be selected and adapted by the user.

Formulas can be used to relate the values of quantitative attributes. The user can write formulas with usual types of functions (arithmetic, trigonometric, statistics, etc.). A formula can be associated to a table. For example, figure 4 shows a case of formula related to a table. This is the case of formula with a summatory where the index covers the total number of elements of the table. In this case, each row corresponds to an element of the set and columns identify values that can be used in the formula.

Images as illustrations and other graphics can be also included in the text for domain knowledge: qualitative number lines (a graphical view of qualitative interpretation), graph of relations (graphical description of relations), 2D graphic (quantitative relation between two magnitudes), etc.

In the document, there are natural language paragraphs to describe tasks, input-output roles, methods and method assumptions. This is prefixed text in the document that cannot be modified by the user but its explicit presence is very important to provide a complete view of domain knowledge to understand (1) the role of domain knowledge in the global problem solving process and (2) method assumptions and general domain assumptions. The user can modify role mappings, i.e., the user can associate domain knowledge to prefixed input and output roles. For example, as it is shown by figure 1, this is represented by writing between parentheses the names of attributes next to the name of the role (note that these roles are defined for classes, i.e. every instance inherits the general role description).


The value of the average rainfall of an area is computed using the following formula:

$$R_i = \sum \alpha_j P_j$$

where, according to the table 6.2, R_i is the average rainfall of area i, α_j is the weight j, and P_j is the current rain of pluviometer j.

area	pluviometer	weight
Guadalteba area	Guadalteba pluviometer	0.2
	Conde pluviometer	0.2
	La Real pluviometer	0.3
	Becerro pluviometer	0.2
	La Encantada pluviometer	0.1
Guadalhorce area	Fuente Piedra pluviometer	0.4
	Colmenar pluviometer	0.4
	El Torcal pluviometer	0.3
	Becerro pluviometer	0.3
Cartama area	El Torcal pluviometer	0.1
	Paredones pluviometer	0.2
	Casarabonela pluviometer	0.2
	Fahala pluviometer	0.2
	Coin pluviometer	0.3

Table 6.2: Table of pluviometers of the rainfal areas.

Figure 4: Example of formula associated to a table.

4. A specialized word processor for hydrologic knowledge models

This section summarizes the KATS word processor, one of the software tools that we developed following the general approach described in this paper (a preliminary version of this tool was described in [12]). KATS was developed to help in building and maintaining the knowledge model of an emergency management system called SAIDA. SAIDA is a knowledge system that assists operators of river basin control centers during flash flood problems. SAIDA was initially developed for a particular river basin but it was required to be portable for other basins.

The SAIDA's knowledge model includes different types of tasks (assessment, prediction, scheduling) with 21 different types of knowledge bases that can be instantiated for 3 types of agents. For example, in the case of Júcar river in Spain, there are a total of 36 agents, each one with up to 6 types of knowledge representations (using rules, frames, uncertainty bayesian networks, logic clauses, temporal and spatial references, etc.). The SAIDA's knowledge model was represented as a conventional document as it is described in this paper.

The resulting document (initial version for the Júcar river basin) includes 70 pages with 4 chapters, 8 sections and 9 subsections (with 25 paragraphs for concepts, 55 paragraphs for attributes, 18 tables for instances, 1 graph of relations, 16 formulas, 43 number lines, 29 decision tables and 1 causal table).

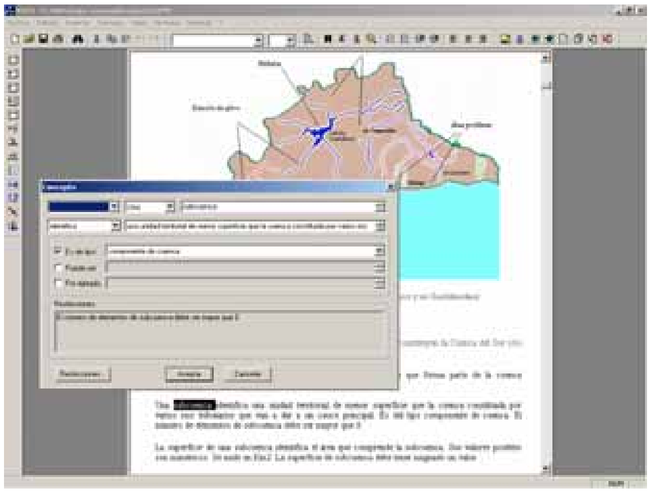


Figure 5: Example of screen presented by the KATS tool. The tool presents the knowledge model showing different pages of a document as it is shown by standard word processors. Specific local windows guide the user in writing the different parts of the document keeping consistency between different components (concepts, attributes, tables, formulas, etc.).

With the KATS tool, the user may read and write the document corresponding to the knowledge model. The user can read the entire document but, since the specialized word processor assumes a prefixed set of problem-solving methods, the user can only change the part of the document corresponding to the domain model. The word processor assumes a fixed structure of the document (chapters, sections, etc.) following a top-down description according to the task-method organization. The word processor also assumes the presence of prefixed text paragraphs corresponding to problem-solving knowledge that the user cannot change.

During the creation and maintenance of the document the user writes domain knowledge and the word processor makes the following automatic procedures for assistance: (1) *syntax checking*, to verify that each component of the document (text paragraph, table, formula, etc.) satisfies the corresponding valid grammar, (2) *consistency checking*, to verify whether the information written by the user verifies method assumptions and general domain assumptions, (3) *guidance*, the word processor uses domain specific assumptions written by the user to guide the user in the construction of the model (for example, the user can declare that the number of instances of reservoirs in the Júcar basin is 7 before writing any particular instance), and (4) *changes propagation*, when the developer writes a certain part of the document, the word processor writes automatically other parts of the document to create complementary views of the knowledge model that help to better understand the model with summaries and views at different levels of abstraction.

The user interface of KATS was designed according to the standards of the most extended word processors. The main window of the user interface shows an image of the document as it will be printed out on paper. When the user wants to modify a variable part of the document, she or he clicks directly on it (double click) and automatically a specialized editor window is presented. For example, if the user clicks on a text paragraph for attributes, an additional window corresponding to the editor for attributes is presented. This editor allows the user to describe an attribute by selecting menu options and completing prefixed natural language sentences. Similarly, if the user clicks on a decision table, a specialized editor for decision tables is presented. The window for each particular editor has a similar appearance to the way the content is presented in the document, together with certain buttons and resources for manipulation.

With these local editors, the word processor checks the consistency of the document and guides the user. Thus, for example, the editor of a decision table constraints the potential content of the table. The table only can include columns (or arrows) that have been defined as attributes of concepts and the content of the table is constrained with the allowed types of values for those attributes.

In order to be operational (to be interpreted by inference procedures) the content of the document is translated by KATS into a knowledge base with conventional symbolic structures (rules, frames, etc.). The complete document is formally written using a formal language. A specific compiler program designed for this purpose automatically translates this source language into symbolic languages (rules, frames, etc.) that can be interpreted by the inference procedures. Basically, this compiler translates the representation resources of the document in the following way: (1) paragraphs for term description, graphs for relations, tables of instances are translated into hierarchies of classes, subclasses with attributes and values and also sets of agents (2) formulas and iterative formulas are translated into functional expressions with temporal extensions, (3) decision tables are translated into rules and frame-like patterns, (4) number lines are translated into attributes and rules, (5) tables of causal relations are translated into models based on bayesian networks, and (6) graphs of processes are translated into a temporal causal network. For example, the symbolic version for the Júcar river basin includes the following components: 25 classes, 150 attributes, 204 instances, 173 nodes of a causal network, 1510 rules, 264 frame-like patterns, 346 nodes of Bayesian networks, 67 logic clauses and 124 functional expressions.

5. Discussion

The KATS tool was analyzed to evaluate its practical utility. Besides a subjective evaluation by domain experts that confirmed a satisfactory operation for different operation scenarios, the utility of the tool was evaluated using an objective method. This method was based on a set of metrics to compare the performance of the KATS tool to the approach followed by domain-based or method-based knowledge modeling tools (such as MOLE [6], SALT [11] or SIRAH [1]) (details of the metrics used for the evaluation method can be found in [2]).

The analysis of the evaluation process applied to KATS showed several advantages provided by the approach described in this paper. The results of the evaluation showed significant less modeling effort using KATS (for example, the description of domain knowledge in the document is significantly smaller compared to a direct description with symbolic languages). KATS also shows better results about model understanding mainly because end-users use a common document-based representation and do not have to learn symbolic languages (rules, bayesian networks, frames, etc.). This better model understanding is also supported by using explicit descriptions of inference steps and roles that allow to anticipate the impact of changes in domain knowledge. In addition to that, KATS increases the level of support to the end user compared to other solutions. This is mainly due to explicit representations about assumptions. The capacity of KATS of using user defined domain-specific assumptions is also a powerful technique to provide better assistance. For example, conventional tools do not provide flexible mechanisms to guarantee the integrity of rule bases. In KATS, part of a rule base can be written as a table with certain domain-specific assumptions defined by the user (exclusive conditions, exhaustive values for premises, max. number of instances, etc.) that help to check consistency during maintenance.

From the point of view of friendly knowledge formulation, other approaches have been developed. For example, within the OpenMind project [16], different specific tools have been developed to facilitate the way users write new knowledge [17] [18]. These approaches are restricted to one specific representation so they are less general than the approach followed by KATS and they are not based on document representations.

Other proposals also follow a documentation approach for the maintenance of knowledge bases. For example, the MODI system [9] presents partial text descriptions of certain components of the knowledge base, linked to the knowledge model offering automatic support during maintenance. Another proposal is related to the Halo Project [14] within the semantic web. The knowledge base is presented using a document approach and, internally, it is formulated using an operational representation language (based on RDF). Compared to the approach followed by KATS, Halo is oriented to ontology representations (emphasizing sharing and reuse of domain knowledge) and the approach of KATS also includes inference procedures for problem solving knowledge besides the representation of domain knowledge.

6. Conclusions

In summary, according to the approach that we present in this paper, a knowledge modeling tool is considered as a specialized word processor with which a domain expert can read and write a special type of document following prefixed conventions about its format and content. The document describes a knowledge model as a set of problem-solving tasks using standard document-based representation resources (such as text paragraphs, tables, formulas, illustrations, etc.) commonly understood by users that are not familiar with computer languages. The word processor helps the user in writing the document keeping the consistency of the model. The word processor translates the document into an operational knowledge base and, consequently, the document can be interpreted by the inference procedures to automatically solve the problems for which the model has been designed.

One direct advantage of this approach is that the user has a more natural perception of the knowledge base because it is viewed as a document with text, tables, graphics, etc. instead of computer oriented languages (rules, constraints, frames, logic clauses, etc.). In addition, the tool has a quick acceptance and assimilation by the user because the procedures for editing the knowledge base are familiar for people (not only for computer programmers) due to the extended use of word processors.

Following the general approach presented in this paper, we developed several knowledge modeling tools for different domains. For example, the KATS tool was developed in the domain of hydrology. Part of our current research work is now oriented to generalize this approach by developing domain-independent word processors with document-based representations able to be interpreted with general inference procedures.

Acknowledgements

The development of the SAIDA system was supported by the Ministry of Environment of Spain (*Dirección General de Obras Hidráulicas y Calidad de las Aguas*) with the participation of local public organizations from river basins (*Confederación Hidrográfica del Júcar and Confederación Hidrográfica del Sur de España*). The development of the KATS tool was supported by the Ministry of Science and Technology of Spain within the RIADA project (REN2000-1394-C02-0).

References

[1] M. Alonso, J. Cuena, M. Molina: "SIRAH: An Architecture for Professional Intelligence". Proc. 9th European Conference on Artificial Intelligence ECAI 90. Pp: 19-24. Stockholm, Sweden. August, 1990.

[2] G. Blasco: "Procesadores de textos como herramientas de soporte a la creación y mantenimiento de modelos de conocimiento", PhD thesis. Universidad Politécnica de Madrid, Spain, 2008

[3] W. Clancey: "Heuristic Classification". Artificial Intelligence 27, 1985

[4] J. Cuena, M. Molina: "The role of knowledge modelling techniques in software development: a general approach based on a knowledge management tool". International Journal of Human-Computer Studies. Academic Press. 2000.

[5] G. Dobbie, X. Y. Wu, T. W. Ling y M. L. Lee: "ORA-SS: An Object-Relationship-Attribute Model for SemiStructured Data". Technical Report TR21/00. School of Computing, National University of Singapore. 2000.

[6] L. J. Eshelman, D. Ehret, J. P. McDermott, M. Tan: "MOLE: A Tenacious Knowledge-Acquisition Tool". International Journal of Man-Machine Studies. Vol. 26, nº 1. Pp: 41-54. January, 1987.

[7] A. Farquhar, R. Fikes, J. Rice: "The Ontolingua Server: a Tool for Collaborative Ontology Construction", International Journal of Human-Computer Studies, 46. Pp: 707-727, 1997.

[8] J. Hendler, E. A. Feigenbaum: "Knowledge Is Power: The Semantic Web Vision". Proceedings of Web Intelligence. Pp: 18-29. Japan. July, 2001.

[9] E. Lutz: "The knowledge Base Maintenance Assistant". Proc. 8th Knowledge-Based Software Engineering Conference, Chicago, USA. 1993.

[10] A. Macintyre: "KREST User Manual 2.5". Wrije Universiteit Brussel, AI-lab. Brussels. 1993.

[11] S. Marcus y J. McDermott: "SALT: A knowledge acquisition language for propose and revise systems". Artificial Intelligence. Vol. 39, nº 1. Pp: 1-37. May, 1989.

[12] M. Molina, G. Blasco: "KATS: A Knowledge Acquisition Tool Based on Electronic Document Processing". 14th International Conference on Knowledge Engineering and Knowledge Management EKAW 04. In "Engineering Knowledge in the Age of the Semantic Web" Lecture Notes in Artificial Intelligence, nº 3257. Springer Verlag. Whittlebury Hay, UK, October, 2004.

[13] N. Noy, R. Fergerson, M. Musen: "The knowledge model of Protege-2000: Combining interoperability and flexibility". Proc. 2th International Conference on Knowledge Engineering and Knowledge Management. Engineering Knowledge in the Age of the Semantic Web (EKAW 2000). Juan-les-Pins, France. October, 2000.

[14] G. Paul: "Creating a Digital Aristotle: A Computerized Knowledge System for Scientists and Students". Bussiness Wire. February, 2004.

[15] G. Schreiber, H. Akkermans, A. Anjewierden, R. De Hoog, N. Shadbolt, W. Van de Velde, B. Wielinga: "Knowledge engineering and management. The CommonKADS methodology" MIT Press, 2000.

[16] P. Singh, T. Lin, E.T. Mueller, G. Lim, T. Perkins and W. Li Zhu (2002). Open Mind Common Sense: Knowledge acquisition from the general public. Proc. of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems. Irvine, CA.

[17] P. Singh, B. Barry y H. Liu: "Teaching machines about everyday life.". BT Technology Journal. Vol. 22, nº 4. Pp: 227-240. 2004.

[18] R. Williams, B. Barry y P. Singh: "ComicKit: acquiring story scripts using commonsense feedback." Proc. ACM International Conference on Intelligent User Interfaces (IUI 2005). San Diego, CA. 2005.