A framework of knowledge versioning management

Article *in* Expert Systems · June 2004

DOI: 10.1111/j.1468-0394.2004.00271.x

CITATIONS
16

READS
164

2 authors:

Michael Maliappis
Agricultural University of Athens
22 PUBLICATIONS **185** CITATIONS

SEE PROFILE

Alexander Sideridis
Agricultural University of Athens
120 PUBLICATIONS **1,379** CITATIONS

SEE PROFILE

Article

A framework of knowledge versioning management

M. T. Maliappis and A. B. Sideridis

Informatics Laboratory, Agricultural University of Athens, 75 Iera Odos, 11855 Athens, Greece
E-mail: michael@aua.gr

Abstract: *Knowledge is an inherently dynamic entity continuously changing and evolving. In many cases, the coexistence of different versions of the same core knowledge is a necessity. So is the availability of the proper environment and tools to deal with knowledge versioning. In this paper, a framework of knowledge versioning management is proposed and implemented dealing with hybrid knowledge representation models using frames and rules. This framework facilitates knowledge version handling and maintenance, improving, in parallel, knowledge sharing and reuse. Knowledge components are stored in a set of tables and handled as data under the auspices of a database management system. The proper structure of tables and their relationships allows the creation of independent knowledge modules. Several knowledge modules can be assembled to construct higher level modules, which finally form versions of knowledge. Corresponding knowledge base versions consist of several knowledge modules easy to handle and process in various application areas. The proposed framework has been implemented and thoroughly examined in an application area of great importance, such as pest management.*

Keywords: knowledge bases, knowledge versioning, knowledge reuse, knowledge modules

1. Introduction

Knowledge is an inherently dynamic entity that changes extremely rapid, especially nowadays. Scientific research digs always at deeper levels and brings to light more details about the world and the relations of the world with human beings. Evolution of the perception of the world leads to changes in the abstract descriptions of the world, which in turn alters their representations. These changes can be seen as modifications in the real world (domain changes) leading to analogous changes to the specifications of the corresponding objects in the knowledge representation, or changes in the conceptualization, caused by alterations of how the world is perceived (e.g. by better understanding of a specific domain) or by adaptation of the conceptual model for other tasks. Knowledge bases, which stand for one type of knowledge representation, very often need to be modified as a result of these changes.

Various efforts of coding and representation of knowledge will always be supposed to take into consideration the evolution of knowledge, as well as the need for simultaneous existence of various versions of the same knowledge. This is rendered necessary because of the evolution and enrichment of knowledge and because of ways of knowledge differentiation.

- *The differentiation of knowledge in level of plenitude and maturity*. In the process of knowledge evolution, some versions might have been exhaustively tested and they include knowledge that can be used productively, while other versions include knowledge in different levels of testing, maturity, plenitude and correctness. Versioning allows experimentation with different portions of knowledge or incomplete knowledge in the same environment until convergence to the desired results.
- *The representation of the same knowledge in different languages or dialects*. The translation of a knowledge base into another language or dialect, although it contains essentially the same knowledge, needs special effort. Versioning allows the coexistence of a knowledge base in several languages or dialects in the same environment. The coexistence of the same knowledge in different technical or experience levels, using relevant terms and languages, can be considered as a versioning problem, too.
- *The existence of knowledge that is valid at different time intervals and must always be available for the export of conclusions for facts that are reported in different time periods (valid-time knowledge)*. The differentiation of knowledge for a limited time interval (temporal alteration) that leads to the need for versions of knowledge with limited time scope can be handled similarly.
- *The existence of knowledge that is differentiated with regard to the place of application*. In this case the base core knowledge remains the same, while some portions of it, depending on environmental conditions, change according to geographical location.
- *The existence of knowledge versions that can be used with small differentiations in similar or relative cognitive fields or subfields*.

- *In cases of disagreement between experts on a particular topic there is a need to keep separate versions of knowledge*. The ability to construct different versions of a knowledge base differentiated only on the specific topics of disagreement gives the opportunity to validate and test the various versions of the knowledge for their effectiveness in similar problems.
- *Versions that are kept for purely historical reasons or to answer why a previous judgment has been held.*

The motivation for this work came from some practical situations that were dealt with in the Informatics Laboratory of the Agricultural University of Athens. Some preliminary results of this work appeared in Yialouris *et al.* (1997), Lorentzos *et al.* (1999b), Passam *et al.* (2001) and Maliappis *et al.* (2001) concerning versioning depending on time (valid-time knowledge), knowledge applied to similar domains (relative domain knowledge) and knowledge translation in a new language or locale.

There are application areas such as law (Villa & Yoshino, 1995; Aravantinos *et al.*, 1996), agricultural and others (Sideridis *et al.*, 1997a, 1997b) where the time when the knowledge is updated represents itself an inherent part of the knowledge. After each update, the content of the previous knowledge base is then integrated within the content of the current knowledge base (Lorentzos *et al.*, 1999a). Knowledge base maintenance, which in any case is a difficult and cumbersome job, in these situations becomes very complicated. This kind of versioning can be handled efficiently using the proposed knowledge versioning framework.

Many fields exist in agriculture, medicine and other disciplines having different characteristics that share a common core of knowledge. In all cases beyond the common core of knowledge, extra specific knowledge is needed to handle the peculiarities of individual situations. Such applications can be found in horticulture (Passam *et al.*, 2001) for some groups of vegetables, such as tomato, cucumber, melon, lettuce, pepper and aubergine, where a common core of knowledge can be identified in their cultivation process. The knowledge needed to deal with each specific crop is an adapted version of the knowledge needed for other crops.

In constructing knowledge bases, maintenance of knowledge evolution is a headache for the knowledge engineer or administrator of knowledge. To cope with evolving knowledge, knowledge versioning is needed to support multiple variants of knowledge. We can define *knowledge versioning* as the ability to manage knowledge changes and their effects by creating and maintaining different variants of a knowledge base. The methodology to support knowledge versioning should provide methods to distinguish and recognize versions and procedures for changes and updates.

In this paper, a versioning framework is presented dealing with several kinds of knowledge versioning, such as knowledge evolution or enrichment, valid-time knowledge, space variant knowledge, relative domain knowledge and knowledge translation. The proposed framework uses knowledge components or modules to implement versioning, which can be used as a vehicle to bring knowledge sharing and reuse to the forefront. The overall structure of the framework is thoroughly described in Sections 2, 3 and 4 with Section 4 containing an overview of the operations imposed on knowledge objects by the framework. Section 5 presents the implementation of an illustrative example of the versioning framework, in order to clarify the concepts and ideas presented. The paper ends with conclusions and implications of the proposed framework.

2. Versioning framework

Several studies related to the management of different versions of complex objects can be identified in various fields, such as software engineering, database management and ontology management. Some of these works contributed ideas and techniques in the development of the proposed versioning framework.

Software engineering is the first field that was involved in the versioning investigation. Several frameworks and models have been proposed for handling versioning strategies for large-scale software projects (Kitcharoensakkul & Wuwongse, 2001). Identification, control, recording and tracking of the evolution of software products, objects, structures and their interrelationships have been investigated from the software development point of view.

Two trends towards versioning implementation can be identified in the field of database management: first concerning database schema evolution and second concerning evolution of the contents of the database (Gancarski *et al.*, 1995; Franconi *et al.*, 2000). A survey of schema versioning issues for database systems can be found in Roddick (1996). In Cellary and Jomier (1990) versioning is treated in object-oriented database contents, intended mainly for maintenance of transaction consistency in multiversion databases.

Ontology management is a field in which, recently, versioning handling appears to be of strong interest (Das *et al.*, 2001; Heflin, 2001; Klein & Fensel, 2001; Kiryakov *et al.*, 2002; Maedche *et al.*, 2002). Klein and Fensel (2001) analyze ontology versioning, providing an overview of causes and consequences of the changes in the ontology, while Das *et al.* (2001) stress the importance of versioning control in a framework of ontology management. Kiryakov *et al.* (2002) describe a knowledge control system which contains a module for tracking ontology changes and deals with ontology versions. Heflin (2001) in his work emphasizes evolution of ontologies on the Web and presents a Web-based knowledge representation language that supports multiple versions of ontologies.

In database and ontology management we have rather passive objects, databases or ontologies, which can be used by active objects, such as programs or agents. Therefore, in considering versioning in these fields relationships between objects play an important role. In contrast, in knowledge base management, knowledge base objects are quite autonomous objects, so, although many of the versioning principles may be the same, issues derived from their application are different.

Taking into consideration all the reasoning mentioned in the introduction, as well as the various ways the same problem is coped with in related fields and some guidelines for an efficient knowledge framework (Das *et al.*, 2001), we concluded that an efficient knowledge versioning framework for management of different knowledge versions in the field of expert systems should

- provide an unambiguous reference to the intended definition of concepts (identification);
- make obvious the relation of one version of a concept or relation to other versions of that construct (follow-up of changes – change tracking);
- offer a language of knowledge representation and the suitable tools for its implementation;
- provide reliable and flexible tools for easy management (change, control, search) of knowledge and its components (versions, modules, rules and frames) – the knowledge base development and the maintenance process have to be simple and reliable, and the tools have to be usable by knowledge engineers as well as domain experts and simple users;
- provide reliable security management – any system that allows multi-user access needs to be secure to protect the integrity of the data, prevent unauthorized access and support multiple access levels;
- give an overall frame of creation of systems which inspires confidence in the users by providing the mechanism for evaluation, validation and verification of knowledge – automatic or semi-automatic processes are of great importance for the quality of knowledge, especially in cases of large knowledge bases;
- give the possibility of creation of bases and systems of knowledge that can be used easily and efficiently to solve real problems – the framework should aim at knowledge reuse and sharing of already existing knowledge to the largest possible degree in new knowledge bases and applications;
- offer the proper tools to create new versions of knowledge in other languages by translating the initial version, facilitating internationalization;
- give multiple levels of access – there is no one-way proper access to the knowledge; the framework should be able to provide several levels of access, such as interactive use through a local GUI environment or a World Wide Web browser and the incorporation in a wider system for decision-making, via API materialized in Java;
- provide proper interfaces, such as extensible mark-up language (XML), for knowledge exchange and distribution through the Internet. Since XML is becoming a widely used means for information sharing and distribution between applications, the framework solution needs to provide XML interfaces to enable interaction and inter-operability with the external world.

In this paper we concentrate on the underlying knowledge representation model, the adaptations needed to implement knowledge versioning and the design principles of the versioning framework. The full implementation of the versioning framework is beyond the scope of this paper and will be described elsewhere.

3. Knowledge representation model

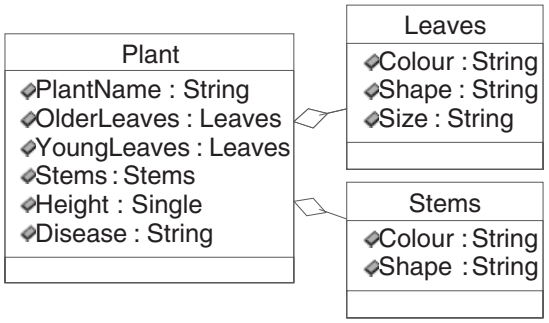
In the core of each knowledge base system design is the model used to represent knowledge. In this section we analyze framework features from a semantic point of view, we summarize the entities that constitute a knowledge base and we explore the relationships between these entities. The knowledge model used is a hybrid one in which knowledge is represented using rules and frames. The static knowledge concerning the entities is represented with frames and rules are used to express heuristics or the logic which joins the entities. If we consider a knowledge base as a kind of programming, frames represent data structures and variables and rules correspond to the programming instructions, which act on data.

A knowledge base is considered as a finite set $\{\mathrm{KM}_i | i = 1, 2, \ldots, k\}$, where each KM_i represents a knowledge module. A knowledge module is defined as a finite set $\{R_i | i = 1, 2, \ldots, m\}$, where each R_i represents a rule. Every rule R_i of this set is an expression of the form

$$\textbf{IF } C_1 \text{ AND } C_2 \text{ AND } \ldots \text{ AND } C_n \textbf{ THEN } S$$

where the C_j represent the conditions and S represents the conclusion of R. C_j and S are called the components of R. The number of conditions of a rule is finite. Distinct rules may have different conditions and conclusions. Each condition C_j is represented by an O–A–V triplet, where O corresponds to the object, which is a frame slot, A corresponds to the attribute, and V corresponds to the value of the slot.

Frames can be separated into two categories, frames representing concepts or classes and frames representing instances of these classes. Classes are represented as frames with slots and facets and are organized in a taxonomy or 'is-a' hierarchy. Classes can have local slots and slots inherited from their parent classes. Instances occur at the leaves of the hierarchy and concentrate all the slots of this hierarchy.



IF Plant.OlderLeaves.Colour have value Yellow_Margins
AND Plant.OlderLeaves.Colour have value Yellow_Between_Veins
THEN Plant.Disease have value Potassium_Deficiency

IF Plant.Stems.Size have value Thin
AND Plant.YoungLeaves.Size have value Small
AND Plant.YoungLeaves.Colour have value Pale_Green
AND Plant.OlderLeaves.Colour have value Yellow
THEN Plant.Disease have value Nitrogen_Deficiency

Figure 1: *An example of a hybrid knowledge module.*

Figure 1 shows an example of three frames and two rules that use these frames.

Frames and rules are assembled to form modules of knowledge. Modules are the main building blocks of a knowledge base. Modules are relatively independent and self-existent portions of knowledge, referred to some narrow cognitive field. They use the mechanism of inheritance to assemble bigger units of knowledge. Various modules of knowledge can be combined to compose other modules of knowledge and finally one or more knowledge bases. Thus, knowledge representation presents a hierarchical structure; each knowledge module can be a structural element for more than one knowledge base and each knowledge base is constituted by one or more modules.

Partitioning of knowledge into modules is a way to promote knowledge sharing and reuse. According to Philip (1993), a module, in knowledge represented by rules, is a group of rules. A modular system is one having modules with high cohesion and low coupling between the modules. Modularity allows a high-level design that decomposes a large system into smaller components and describes the internal interfaces between components. Modularity is a factor in the design of knowledge systems because a rule module is more likely to be reusable than an unstructured collection of rules. Reuse of knowledge apart from being beneficial is also essential in constructing knowledge-based systems. Knowledge gathered from previous elicitations and a number of knowledge bases could be reused and shared (Richards, 2000) resulting in reduced development time and improved knowledge base quality. Creation of a module is a task carried out mainly by the domain expert. The expert is the only one who is in a position to define what portions of knowledge can be separated to form an independent module.

The framework proposed here uses inheritance in two ways. The first is in the construction of frames, where each class can be created as a subclass of a more general one. The second use is in the construction of modules, where each module can inherit knowledge contained in another module defined as a derived module. Inheritance is used as a mechanism to specialize a module to be used for different purposes. In both cases, we only use single inheritance to keep matters simple and avoid the complications and conflicts derived from multiple inheritance. Note that there is no longer the notion of a single, universal knowledge base. Rather, each component can be thought of as specifying a 'mini knowledge base', which can be assembled by merging the knowledge contained in its predecessor components.

The above knowledge model corresponds to a monoversion knowledge base, i.e. a knowledge base consisting of a single variant of knowledge. In the next section we introduce versioning control, which differentiates the knowledge model and knowledge objects in several aspects.

4. Knowledge base version approach

4.1. Manipulation of versioning

A knowledge base can be seen as a representation of the real world. If a change is made in the perception or point of view of the real world, a new knowledge base can arise which may be completely different or a mere modification or expansion of the initial knowledge base. In the latter case of slight modification or expansion of the initial knowledge base, the adoption of knowledge base versioning offers several advantages over the independent treatment of the knowledge bases.

A knowledge base is composed of a set of knowledge objects. Each knowledge object is defined as a pair (object identifier, object value). Such a knowledge base defines a monoversion knowledge base, i.e. a knowledge base without versioning. In a multiversion knowledge base approach a *multiversion object* is defined as a pair (object identifier, set of object versions) and an *object version* is defined as a pair (version identifier, object value). A *knowledge base version* is defined as a pair composed of a version identifier and a set of objects contained in the multiversion knowledge base, one version per object. Thus, a multiversion knowledge base is defined as a set of logically independent and identifiable knowledge base versions.

Operations concerning multiversion knowledge bases can be divided into two broad categories, one containing the non-versioning operations and the other containing operations affecting versioning. Non-versioning operations act on a single version of a knowledge object, effecting changes to its contents independently of other versions.

Versioning operations create new versions of knowledge objects. They are addressed to a version of a knowledge object, the parent knowledge object version, and they create a child knowledge object version, which is a logical copy of the parent. Thus, the set of versions of knowledge objects is organized as a tree, called a *derivation tree*. Once created, the new knowledge object can be evolved autonomously, according to non-versioning operations addressed to it.

A user operates on a multiversion knowledge base choosing a specific knowledge base version. One way to do this is to specify a knowledge base version identifier used by the system. However, it is more convenient to use other identifiers, which reflect the semantics of the knowledge base and which are translated into the system identifiers. For instance, in a valid-time knowledge base this can be achieved using the time of interest, in a spatial knowledge base using spatial coordinates or the identification of a place and in a domain knowledge base using the name of the domain. Once the knowledge base version is chosen, the user may perform non-versioning operations, as if he/she works on a monoversion knowledge base. The system automatically identifies the version of objects belonging to the knowledge base version chosen and acts appropriately.

Except for functions concerning an individual knowledge base and its objects, users can work on the whole multiversion knowledge base using operations such as moving objects between knowledge base versions, comparing object versions, listing knowledge bases in which a specified object participates and so on.

The framework structure facilitates investigation of knowledge objects and extraction of useful information for knowledge base structure and contents. These functions use SQL statements to formulate queries whose answers assist the expert or the casual user in the study of the knowledge base contents and the evolution of knowledge. Among the questions that the system is able to answer are the following.

- Which are the knowledge versions, knowledge modules, frames and rules of the system?
- Which objects constitute each knowledge base version?
- Which are the knowledge base versions that contain a specific object?
- Which are the common objects for two specific knowledge base versions?
- Which are the orphan objects of the system, i.e. objects that do not belong to any knowledge base version?
- Which are the shared/unshared objects belonging to specific knowledge base versions?

4.2. *Version identification*

A major aspect of the versioning problem is identification of objects and referring. Usually, the same name represents the same concept. Any knowledge specification has identity and any conceptualization has identity, too. Thus, any time a new version of an object is created the system produces a new identifier and links the object with its predecessors and the version to which it belongs. The identifier is usually an incremental number and the linkage is accomplished by the insertion of suitable records in the proper tables of the knowledge base schema.

The basic objects that need to be identified are frames, rules and modules. Frames and rules are the building objects of modules and modules are the building objects of versions. All the other objects (conditions, conclusions) are unaware of the versioning issue. They constitute a repository of knowledge objects from where they are withdrawn each time they are needed to form frames, rules and modules, which, in their turn, are used to form knowledge modules.

The creation of new versions of an object is more or less an expert's responsibility. The expert decides which changes lead to a new version of an already existing object and which lead to a new object. Changes resulting in different character representations constitute revisions and are referred to a different specification of the same object, and changes in a conceptualization lead to a new object identity. The system offers tools to inspect the knowledge base and try to identify its contents in various ways to help the expert come to a more rational decision.

5. Implementation example

The proposed formalism integrates various individual cases into a general framework. Using the same basic methodology we are able to approach versioning issues coming from several different sources. In all these sources of knowledge differentiation there is a core of immutable knowledge that can be used as an initial building block to construct various versions of the knowledge. We identified four major sources of knowledge differentiation: differentiation of time of application, differentiation because of place of application, differentiation due to domain of application and differentiation due to translation. Each has its own intricacies and special problems in knowledge representation and versioning maintenance. We choose, among them, the category of knowledge change due to domain of application, considering it as the most typical one, and we present its implementation in the remainder of this section.

In horticulture there are groups of vegetables, such as tomato, cucumber, melon, lettuce, pepper and aubergine, where a common core of knowledge can be identified in their cultivation process (Passam *et al.*, 2001; Mahaman *et al.*, 2003). In the perspective of the proposed framework, autonomous pieces of knowledge can be used to construct knowledge modules and each knowledge base can be assembled from several knowledge modules. For each

vegetable a new version of knowledge is constructed containing the proper knowledge modules in their original form or in a new version with the appropriate changes.

As an illustrative example consider a knowledge base concerning pest management for two crops, tomatoes and peppers. It is well known, at least to the experts in the field, that the knowledge needed for pest management has large portions shared between these two crops. Therefore, the knowledge needed for handling the two crops concerns relative cognitive domains and seems to be a suitable field to apply the knowledge versioning framework.

Following the principles of the proposed framework, we started developing first the tomato knowledge base having in mind to identify and modularize knowledge that seems to be independent. The knowledge acquisition process identified, at the top level, two knowledge modules. The first contains knowledge related to identification of insects attacking tomatoes and the second contains knowledge needed to determine disease according to several symptoms occurring on specific parts of the tomato plant, e.g. bloom or fruit.

Each of the top-level modules can be partitioned into several lower level modules concerning more specific characteristics of the situation examined. As is shown in Figure 2, the insect module is partitioned into four lower level modules. The knowledge versioning framework allows the construction of the tomato knowledge base by assembling the previous referred knowledge modules in their initial versions.

For the creation of the pepper knowledge base, since a great portion of the knowledge needed is the same, we use the tomato knowledge base as the parent version and create an identical copy of it as a derived child knowledge base version. Afterwards, we modify the derived knowledge base and adapt it to the peculiarities of the pepper. During this process we have to revise a few of the initial modules, creating new versions of them to be used in the pepper knowledge base.

In the insects top-level module there are a few lower level modules that have to be changed. Therefore, working on the contents of the pepper knowledge base first we create a new version of the insects top-level module, creating an identical copy of it, and afterwards we create new versions of the lower level modules that have to be changed, such as the leaf and beet modules. The final step is to change the knowledge objects, usually rules or frames, contained in the modules at the lowest level.

The whole system is presented in Maliappis *et al.* (2003) where its implementation is fully described. The full implementation concerns three crops, tomato, pepper and aubergine, and covers pest management as well as nutritional deficiencies. The system can easily be extended to include more similar crops, such as melon, lettuce or cucumber.

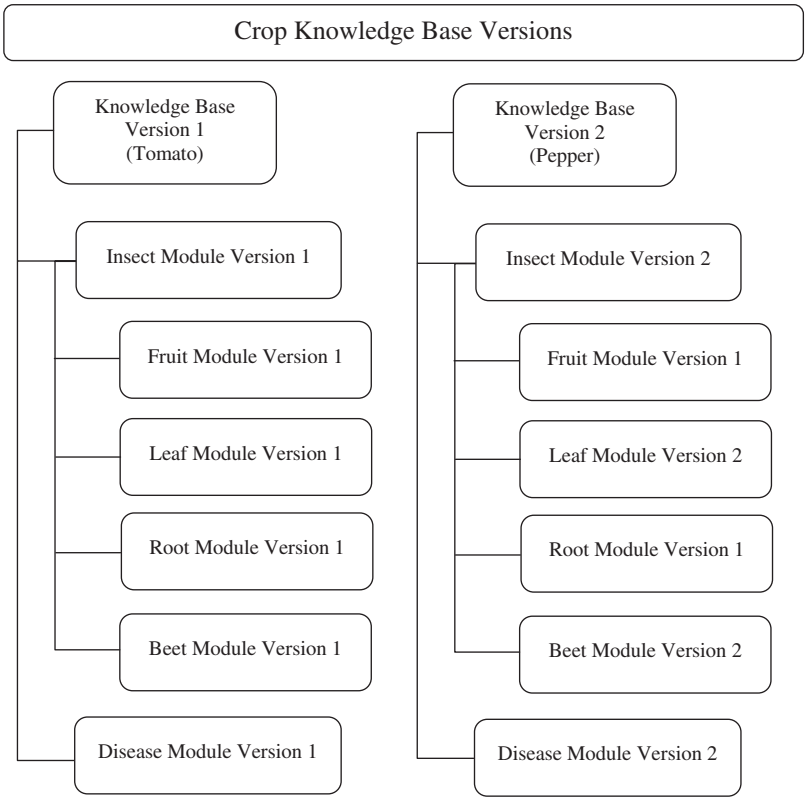


Figure 2: *A simplified portion of the crop knowledge base version structure.*

6. Discussion

The aim of this work was to present a framework for knowledge base versioning management. The presented framework employs a hybrid knowledge model using frames and rules for knowledge representation, exploiting thus the compact structure of a frame system in the representation of facts and the pure declarative nature of rules in reasoning. The proposed framework introduces knowledge base versioning and covers, under the same theoretical umbrella, various forms of knowledge differentiation such as

- knowledge differentiated according to time of application (valid-time knowledge),
- knowledge differentiated according to space of application (space variant knowledge),
- knowledge reported in relevant cognitive fields (relative domain knowledge) and
- knowledge differentiated due to translation.

The knowledge versioning framework offers a powerful tool for managing multiversion knowledge bases. It allows object version identification and provides the platform to construct software tools for version and object manipulation. Using these tools, expert and ordinary users are able to follow the evolution history of the various objects and easily select the desired knowledge base version to work with. At the same time, the underlying model promotes knowledge sharing and reuse between several kinds of knowledge bases.

The versioning framework offers the convenience to produce new knowledge bases by expanding the knowledge already contained in the system. Consequently, the time and cost needed to implement a system are reduced significantly, the process becomes cost effective and information and communication technology dissemination increases.

The implementation of the versioning framework for knowledge representation in pest management showed its advantages over the independent treatment of knowledge bases and made clear some implementation and structural issues. There are advantages at the implementation level, such as the reduction in storage and the easier implementation of checking and evaluation mechanisms. Furthermore, during knowledge base construction, coping with the whole knowledge concerning the crops of interest offers the expert the opportunity to clarify similarities or differences in knowledge which are not apparent at first glance.

On the other hand it is obvious that the framework should offer the proper tools to deal with knowledge modules and versions. Without these tools the effective manipulation of knowledge is impossible.

Since our major aim was to describe knowledge versioning and highlight its characteristics and effects on knowledge maintenance, we omitted details of framework implementation, although sometimes design and implementation are not easily distinguishable. What is worth mentioning is the underlying storage structure on which the framework is based. We chose to store knowledge data using a set of tables and base our framework on the relational database model. This gives us several advantages on storage level and knowledge maintenance as well as the interrogation of knowledge. The proper construction of the database schema and table indexing reduces data redundancy to a minimum and the ease of formulation of queries using SQL allows flexible retrieval and manipulation of knowledge. Exploitation of the maintenance tools offered by database management systems for knowledge manipulation provides methods, structure techniques and tools that improve knowledge engineers' productivity and increase users' control over knowledge.

Further work will be focused on implementation of tools facilitating automatic or semi-automatic construction and manipulation of knowledge versioning. Since each knowledge base system is as useful and reliable as the knowledge it contains, the provision of methods and tools for exhaustive knowledge verification and validation is an absolute necessity. Furthermore, something that needs more investigation is the use of XML and the evolving standards, such as RuleML, in rule representation, sharing and distribution through the Internet. The use of XML, combined with agents' technology, seems to be a promising vehicle for the exploitation of knowledge distributed over different geographical locations or systems using diverse knowledge models.

References

ARAVANTINOS, V., J. STRAGGAS and C.P. YIALOURIS (1996) Temporal knowledge systems in law, in *Proceedings of HERMIS 96, the 3rd Hellenic–European Conference of Mathematics and Informatics*, Athens, Greece, 26–28 September.

CELLARY, W. and G. JOMIER (1990) Consistency of versions in object-oriented databases, in *Proceedings of the 16th International Conference on Very Large DataBases (VLDB '90)*, San Mateo, CA: Morgan Kaufmann.

DAS, A., W. WU and D.L. MCGUINNESS (2001) Industrial strength ontology management, in *Proceedings of the International Semantic Web Working Symposium*, Stanford, CA: Stanford University Press.

FRANCONI, E., F. GRANDI and F. MANDREOLI (2000) A semantic approach for schema evolution and versioning in object-oriented databases, in *Proceedings of the International Conference on Deductive and Object-oriented Databases (DOOD 2000)*, London: Springer, 1048–1062.

GANCARSKI, S., G. JOMIER and M. ZAMFIROIU (1995) A framework for the manipulation of a multiversion database, in *Workshop Proceedings of Database and Expert Systems Applications Conference*, London: Springer, 247–256.

HEFLIN, J.D. (2001) *Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment*, College Park, MD: University of Maryland Press.

KIRYAKOV, A., K. SIMOV and D. OGNYANOV (2002) Ontology middleware: analysis and design, *Deliverable 38*, On-To-Knowledge EU Project, Onto Text Laboratory, Sofia, Bulgaria.

KITCHAROENSAKKUL, S. and V. WUWONGSE (2001) Towards a unified version model using the resource description framework (RDF), *International Journal of Software Engineering and Knowledge Engineering*, **11** (6).

KLEIN, M. and D. FENSEL (2001) Ontology versioning on the semantic Web, in *Proceedings of the International Semantic Web Working Symposium (SWWS)*, Stanford, CA: Stanford University Press, 75–91.

LORENTZOS, N.A., C.P. YIALOURIS and A.B. SIDERIDIS (1999a) Time-evolving rule-based knowledge bases, *Data and Knowledge Engineering*, **29**, 313–335.

LORENTZOS, N.A., M.T. MALIAPPIS, A.B. SIDERIDIS and C.P. YIALOURIS (1999b) Time dependent rule knowledge bases in decision support systems in agriculture, in *Proceedings of the Second European Conference of the European Federation for Information Technology in Agriculture, Food and the Environment*, Bonn, Germany: Universität Bonn-ILB.

MAEDCHE, A., B. MOTIC, L. STOJANOVIC, R. STUDER and R. VOLZ (2002) Managing multiple ontologies and ontology evolution in ontologging, in *Proceedings of the Conference on Intelligent Information Processing, World Computer Congress 2002*, Boston, MA: Kluwer Academic.

MAHAMAN, B.D., H.C. PASSAM, A.B. SIDERIDIS and C.P. YIALOURIS (2003) DIARES-IPM: a diagnostic advisory rule-based expert system for integrated pest management in *Solanaceous* crop systems, *Agricultural Systems*, **76**, 1119–1135.

MALIAPPIS, M.T., A.B. SIDERIDIS and B.D. MAHAMAN (2001) NEST: a new expert system tool and its application to pest management, in *Proceedings of the Third European Conference of the European Federation for Information Technology in Agriculture, Food and the Environment*, Montpellier, France.

MALIAPPIS, M.T., B.D. MAHAMAN and A.B. SIDERIDIS (2003) Knowledge versioning management in agriculture: application to pest management. *Technical Report TR-170*, Informatics Laboratory, Agricultural University of Athens.

PASSAM, H.C., A.B. SIDERIDIS, C.P. YIALOURIS and M.T. MALIAPPIS (2001) Improvement of vegetable quality and water and fertilizer utilization in low-tech greenhouses through a decision support management system, *Journal of Vegetable Crop Production*, **7** (1), 69–82.

PHILIP, G.C. (1993) Guidelines on improving the maintainability and consultation of rule-based expert systems, *Expert Systems with Applications*, **6**, 169–179.

RICHARDS, D. (2000) The reuse of knowledge: a user-centred approach, *International Journal of Human–Computer Studies*, **52**, 553–579.

RODDICK, J.F. (1996) A survey of schema versioning issues for database systems, *Information and Software Technology*, **37** (7), 383–393.

SIDERIDIS, A.B., C.P. YIALOURIS, K. THEOPHILIDOU and V. ARAVANTINOS (1997a) A temporal ES in land reclamation, *Technical Report TR-132,* Informatics Laboratory, Agricultural University of Athens.

SIDERIDIS, A.B., P. HARIZANIS and C.P. YIALOURIS (1997b) Spatiotemporal knowledge in apiculture, *Technical Report TR-133,* Informatics Laboratory, Agricultural University of Athens.

VILLA, L. and H. YOSHINO (1995) Temporal representation for legal reasoning, in *Proceedings of the 3rd International Workshop on Legal Expert Systems for the CISG*, K. D. Ashley and H. Yoshino (eds), College Park, MD.

YIALOURIS, C.P., H.C. PASSAM, A.B. SIDERIDIS and C. METIN (1997) VEGES: a multilingual expert system for diagnosis of pests, diseases and nutritional disorders of six greenhouse vegetables, *Computers and Electronics in Agriculture*, **19**, 55–67.

The authors

Michael Maliappis

Michael Maliappis received a primary degree in mathematics from the Aristotle University of Thessalonica and a postgraduate degree in informatics and operation research from the University of Athens. He is a PhD candidate at the Informatics Laboratory of the Agricultural University of Athens working in the area of knowledge-based systems.

Alexander Sideridis

Alexander Sideridis received a primary degree in mathematics from the University of Athens and an MSc and PhD from Brunel University. He is Professor and Head of the Informatics Laboratory of the Agricultural University of Athens. He has been involved in many major projects for the development of information systems in large public organizations. His current research interests concern the design and implementation of decision support and knowledge-based systems.