Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Int. J. Human-Computer Studies 62 (2005) 619–638

International Journal of Human-Computer Studies

www.elsevier.com/locate/ijhcs

Knowledge editing and maintenance tools for a semantic portal in oncology

Mathieu d'Aquin*, Christophe Bouthier, Sébastien Brachais, Jean Lieber, Amedeo Napoli

LORIA CNRS, INRIA, Nancy Universities, BP 239, 54506 Vandœuvre-lès-Nancy, France

Available online 20 April 2005

Abstract

The research work presented in this paper is about the design of a knowledge system architecture applied to oncology and relying on the semantic Web principles. The core of this architecture is a working knowledge system, called KASIMIR, using an object-based representation formalism and classification-based reasoning. The ontology editor PROTÉGÉ is connected with the KASIMIR system, and is adapted to the particular requirements of KASIMIR. The PROTÉGÉ system enables the integration of several editing and visualization modules. A first knowledge editing module relies on classification-based reasoning for detecting mismatches and redundancies in the edited knowledge hierarchy. A second knowledge editing module also uses classification-based reasoning for comparing two versions of the knowledge base for maintenance purposes. This last module is particularly useful for extracting and analysing the changes occurred during an editing session. Three modules are combined to visualize hierarchies, based on three different techniques having complementary advantages. All these modules—including KASIMIR and PROTÉGÉ—are integrated in a semantic portal architecture based on semantic Web principles. The proposed architecture takes advantage of the semantic Web technologies for integrating the different modules, and for providing a reusable environment for distributed knowledge management in oncology.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Knowledge management; Editing; Representation; Reasoning; Maintenance; Visualization; Semantic Web; Semantic portal

*Corresponding author. Tel.: +333 83 59 20 45; fax: +333 83 41 30 79.
E-mail addresses: daquin@loria.fr (M. d'Aquin), bouthier@loria.fr (C. Bouthier), brachais@loria.fr (S. Brachais), lieber@loria.fr (J. Lieber), napoli@loria.fr (A. Napoli).

1071-5819/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ijhcs.2005.02.003

1. Introduction

The idea of a semantic Web relies on the extension of the present Web in order to make its contents fully available and understandable by machines (Fensel et al., 2003). One of the key issues is the need for ontologies on a particular domain, and for reasoning mechanisms relying on these ontologies. More generally, the design of knowledge system in the framework of the semantic Web relies on a number of (classical) operations: acquiring, maintaining, accessing and exchanging the knowledge elements in a domain or an organization. These operations require the development of tools and new architectures based on adequate languages and terminologies, to provide support for applications (see for example the SESAME architecture (Broekstra et al., 2003) and SEAL (Maedche et al., 2003)).

The KASIMIR system follows the same idea: it aims at managing knowledge in oncology with respect to the semantic Web principles (Lieber et al., 2002; Brachais et al., 2003). In this way, the KASIMIR architecture has to be open, reusable and adaptable, for extensions to other contexts, according to the principles of declarative knowledge system design (see for example Stefik, 1995), where the knowledge components are separated from the manipulation mechanisms. Following this idea, a modular architecture underlying the KASIMIR system is proposed hereafter, with a number of components that can be plugged—or unplugged as well—among which a knowledge management module, a Web portal with a set of associated services depending on the domain. The main tasks that are considered for managing knowledge are the follows:

- Acquiring and modelling knowledge through domain expert interactions.
- Editing knowledge with a knowledge editor, representing knowledge within a knowledge representation formalism, e.g. object-based representation systems or description logics, and controlling the evolution of the knowledge base. The knowledge editing can be guided and controlled by the reasoning module associated with the knowledge representation formalism.
- Visualizing the elements of the knowledge base and the results of the reasoning processes with visualization modules providing different viewpoints on the knowledge base.

Hereafter, we detail the modular architecture of the KASIMIR system, and describe the functionalities of each module (see Fig. 1). A knowledge editor based on the PROTÉGÉ system has been connected with the KASIMIR system, for taking into account the knowledge model and the reasoning process in KASIMIR. The extensibility and the flexibility of the PROTÉGÉ environment allow the integration of companion visualization modules offering visualization with respect to different viewpoints, for helping manipulation and validation of the knowledge units. In addition, a module for comparing two versions of a knowledge base is integrated to the editing environment, having in charge the visualization of the modifications between two consecutive versions of a knowledge base. A discussion on the design

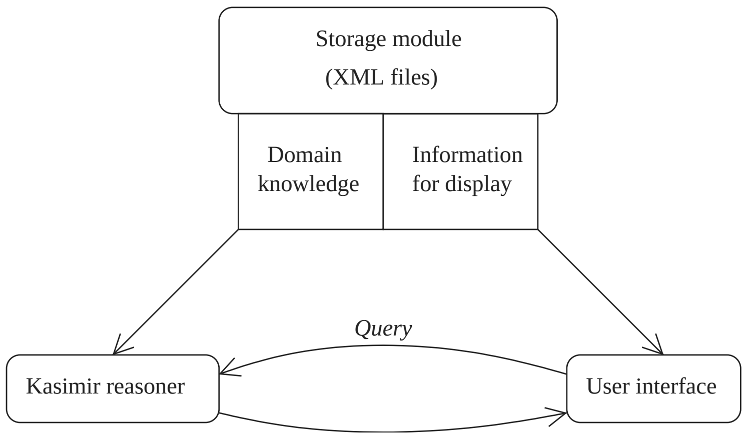


Fig. 1. The current architecture of the KASIMIR system. A KASIMIR knowledge base is stored as a set of XML files, loaded into the KASIMIR reasoner. A configurable user interface is then used for querying the reasoner and displaying the results.

problems of such an architecture within the context of the semantic Web is also proposed.

The outline of the paper is as follows. The next section introduces the KASIMIR system, the medical domain, the knowledge representation and the reasoning framework. In Sections 3 and 4, the modules for knowledge editing, knowledge base maintenance and knowledge visualization are detailed. Finally, the Section 5 includes a discussion on the place of the KASIMIR system with respect to the semantic Web principles, enlightening the role of such a knowledge system within a semantic portal in a given application domain.

2. The representation of knowledge in the KASIMIR system

2.1. The medical context of the KASIMIR system

The KASIMIR system is aimed at managing knowledge and decision support in the domain of oncology, providing a support for the medical treatment of people ill with cancer. This system is currently under development in a multidisciplinary context, where physicians, ergonomists and computer scientists work in collaboration (Lieber et al., 2002). Moreover, oncology is a medical domain of main importance nowadays, where knowledge is constantly evolving. The main problem considered here is to make the best medical decision for a given patient, with respect to the particular data concerning this patient, and to the current state of the knowledge in oncology. The knowledge units considered in the KASIMIR system are relative to decision protocols, simply called *protocols* in the following, that can be likened to decision trees. For example, the protocol concerning breast cancer treatment takes into account characteristics of a patient such as the age, the size of the tumor, the localization of the tumor, etc. Then, some decision elements correspond to these data in the breast cancer protocol, and can be considered by the physician for taking a decision treatment.

A protocol is relative to one type of cancer. At present, several protocols are implemented within the Kasimir system, and real-world experiments are currently carried on. In most of the cases, a protocol can be straightforwardly considered and understood by a physician, i.e. a set of particular characteristics corresponds to a particular decision treatment. However, and this is in concern with about 30% of the patients, the protocol cannot be always readily used, and must be *adapted* for decision making, for example, whenever the patient shows a contraindication for a particular treatment (Lieber et al., 2003). These adaptations are expected to lead to an evolution of the protocol, and to a revision of the associated knowledge units (Sauvagnac, 2000).

The protocols on which relies the Kasimir system are built with respect to the principles of *evidence-based medicine* (Evidence-based medicine working-group, 1992), i.e. a protocol provides recommendations that are in accordance with the current state of the knowledge in oncology. Accordingly, this implies that the knowledge units in the protocols have to be updated on a regular time basis, actually once or twice a year. In parallel, new knowledge bases are designed for taking into account particular decision problems, e.g. post-therapy surveillance, pain or treatments for other cancer localizations (prostate, colon, etc.). The constant evolution and increase of knowledge have to be guided and controlled by the system, and they need appropriate modules for being undertaken. Thus, it is often necessary to edit large set of knowledge units, in a specialized context, showing that knowledge editing must be guided and controlled with appropriate modules, in parallel with the evolution of knowledge. These complex and central tasks of knowledge editing and maintenance are detailed hereafter (see Section 3).

2.2. *Knowledge representation and reasoning in Kasimir*

Knowledge representation is at the heart of the knowledge management process in the Kasimir system. It relies on an object-based representation system (Napoli et al., 1994), that can be likened to a description logic system (Baader et al., 2003). The basic representation unit is the *concept*, that represents a set of objects, or individuals, sharing a number of *properties*, or *attributes*. The set of objects is called the *extension* of the concept, while the corresponding set of properties is called the *intension* of the concept. An individual being a member of the extension of a concept is also called an *instance* of the concept. An attribute has a *domain*, i.e. the concept to which it is attached, and a *range*, determining the type of the admissible values of the attribute. The range of an attribute may be a primitive type (number, string, etc.), another concept, or—a specificity of Kasimir—an interval of numbers. In the latter case, the attribute defines a *relation* between its domain concept and its range concept.

Two kinds of concepts can be distinguished. *Primitive concepts* are considered as atoms of the representation system. They are used as building blocks for the *defined concepts*. Moreover, the intension of a primitive concept is empty, i.e. it has no attribute, while the intension of a defined concept is composed of attributes acting as a set of necessary and sufficient conditions for recognizing an individual as an instance of the corresponding defined concept. Indeed, this quality of the attributes of a defined concept, i.e. being necessary and sufficient conditions, is the basis for the concept classification process that is made precise below.

A *subsumption* relation ($\sqsubseteq$) is defined on the set of concepts in the following way: a concept $\mathtt{C}_1$ is *subsumed* by a concept $\mathtt{C}_2$, denoted by $\mathtt{C}_1 \sqsubseteq \mathtt{C}_2$, whenever the extension of $\mathtt{C}_1$ is necessarily included in the extension of $\mathtt{C}_2$, i.e. the concept $\mathtt{C}_1$ is more specific than the concept $\mathtt{C}_2$, or in a dual way, $\mathtt{C}_2$ is more general than $\mathtt{C}_1$. The subsumption relation is a partial ordering (based on inclusion of extensions) that organizes concepts within a *hierarchy*, i.e. an acyclic directed graph denoted by $\mathscr{H}_{\mathscr{C}}$, where the subsumption relation is declared for primitive concepts, while it is calculated as follows for defined concepts. Given two defined concepts $\mathtt{C}_1$ and $\mathtt{C}_2$, the relation $\mathtt{C}_1 \sqsubseteq \mathtt{C}_2$ holds if and only if, for all attribute $\mathtt{a}_i$ in the subsuming concept $\mathtt{C}_2$, there exists a corresponding attribute $\mathtt{a}_j$ in the subsumed concept $\mathtt{C}_1$ that has the same name and whose characteristics verify the constraints associated with $\mathtt{a}_i$ in $\mathtt{C}_2$. These constraints are relative to the range of the attribute and are verified in the following way:

- If the range of $\mathtt{a}_i$ in $\mathtt{C}_2$ is a primitive type, say $\mathtt{T}_2$, then the range of $\mathtt{a}_j$ in $\mathtt{C}_1$ must be a primitive type, say $\mathtt{T}_1$, equal to $\mathtt{T}_2$ or a subtype of $\mathtt{T}_2$.
- If the range of $\mathtt{a}_i$ in $\mathtt{C}_2$ is a concept, say $\mathtt{D}_2$, then the range of $\mathtt{a}_j$ in $\mathtt{C}_1$, say $\mathtt{D}_1$, must be subsumed by $\mathtt{D}_2$: $\mathtt{D}_1 \sqsubseteq \mathtt{D}_2$.
- If the range of $\mathtt{a}_i$ in $\mathtt{C}_2$ is an interval of numbers, say $[\mathtt{q}_1, \mathtt{q}_2]$, then the range of $\mathtt{a}_j$ in $\mathtt{C}_1$ must be an interval of numbers, say $[\mathtt{p}_1, \mathtt{p}_2]$, included in $[\mathtt{q}_1, \mathtt{q}_2]$.

Given the subsumption relation between concepts, the *classification process* applies to *concept classification* and *instance classification*:

- Concept classification is used for comparing defined concepts, and placing a new concept $\mathtt{C}$ in the concept hierarchy, under its most specific subsumers and over its most general subsumes (Nebel, 1990; Baader et al., 2003).
- Instance classification is used for recognizing that an individual is an instance of a concept.

Defined concepts are used for representing classes of patients sharing common characteristics: these classes are considered as "problems" to which "solutions" may be attached, in accordance with a given protocol. Actually, a solution corresponds to a specific cancer treatment, that can be applied to every individual in the class. Following this idea, the problem of finding a "solution" for a given "problem", i.e. finding the right treatment for a given patient, is considered as a decision-support task, and relies on the classification process, as explained hereafter.

A protocol can be seen as a set of rules $\{\mathtt{pb}_1 \longrightarrow \mathtt{Sol}(\mathtt{pb}_1), \ldots, \mathtt{pb}_n \longrightarrow \mathtt{Sol}(\mathtt{pb}_n)\}$, where $\mathtt{pb}_i$ denotes a problem and $\mathtt{Sol}(\mathtt{pb}_i)$ a solution of $\mathtt{pb}_i$. The decision-support process relies on an inference rule, that can be interpreted as follows: whenever a problem $\mathtt{pb}_2$ is more general than a problem $\mathtt{pb}_1$ (according to the subsumption relation in the concept hierarchy $\mathscr{H}_{\mathscr{C}}$ considered as a problem hierarchy), then every solution of $\mathtt{pb}_2$ is also a solution of $\mathtt{pb}_1$ (Lieber and Napoli, 1998). Following this inference rule, the classification of a target problem `tgt` in $\mathscr{H}_{\mathscr{C}}$ returns the set of the most specific problems subsuming `tgt`. As soon as a subsuming problem has an associated solution, this solution can be reused in the context of `tgt`. The reuse of a solution may require an adaptation of the solution, following the case-based reasoning principles. The details of this extension of classification-based reasoning to case-based reasoning are not discussed here, but details can be found in Lieber et al. (2003).

3. Knowledge editing tools

The KASIMIR system is a knowledge-based system relying on a concept hierarchy. The question was raised, for the KASIMIR system, whether to develop a new specific knowledge editor or to reuse an existing one. This has led to the testing of several tools for this purpose. Among them, are the followings: OILED (Bechhofer et al., 2001), ONTOEDIT (Sure et al., 2002) and PROTÉGÉ (Noy et al., 2000). The main advantage of the knowledge editor OILED is its ability to communicate with a description logic reasoner, like FACT (Horrocks, 1998) or RACER (Haarslev and Möller, 2001). One of the strengths of ONTOEDIT lies in its exporting facilities in some of the knowledge representation languages of the semantic Web. Finally, PROTÉGÉ was chosen, in particular, for its frame-based knowledge representation model described thanks to metaclasses that can be specialized for a specific application. Moreover, the PROTÉGÉ architecture enables to add components, called plugins, for visualization or other purposes, thanks to a Java API, thus making PROTÉGÉ easily extensible and customizable.

This section describes how PROTÉGÉ is customized to be a knowledge editor for KASIMIR. Section 3.1 describes how the PROTÉGÉ metamodel has been adapted. Section 3.2 deals with the connection between PROTÉGÉ and the KASIMIR reasoner and shows some benefits of this connection. Finally, Section 3.3 describes a tool integrated into PROTÉGÉ for comparing two versions of a KASIMIR knowledge base for knowledge maintenance.

3.1. *Adaptation of the PROTÉGÉ metamodel*

The first step of the elaboration of a knowledge editor suited to our needs is to describe the KASIMIR knowledge representation model by specializing the PROTÉGÉ - metaclasses (see Fig. 2). This model is composed of primitive and defined concepts and attributes (see Section 2.2). The concepts are translated in the PROTÉGÉ model by classes, subclassing the PROTÉGÉ :STANDARD-CLASS, the attributes by slots, subclassing the PROTÉGÉ :STANDARD-SLOT and the constraints on attribute values by facets. These classes are themselves described with accuracy. The second step of this elaboration is to adapt the design of the forms associated with this set of new metaclasses in the "Forms" tab-widget to obtain forms especially built for the

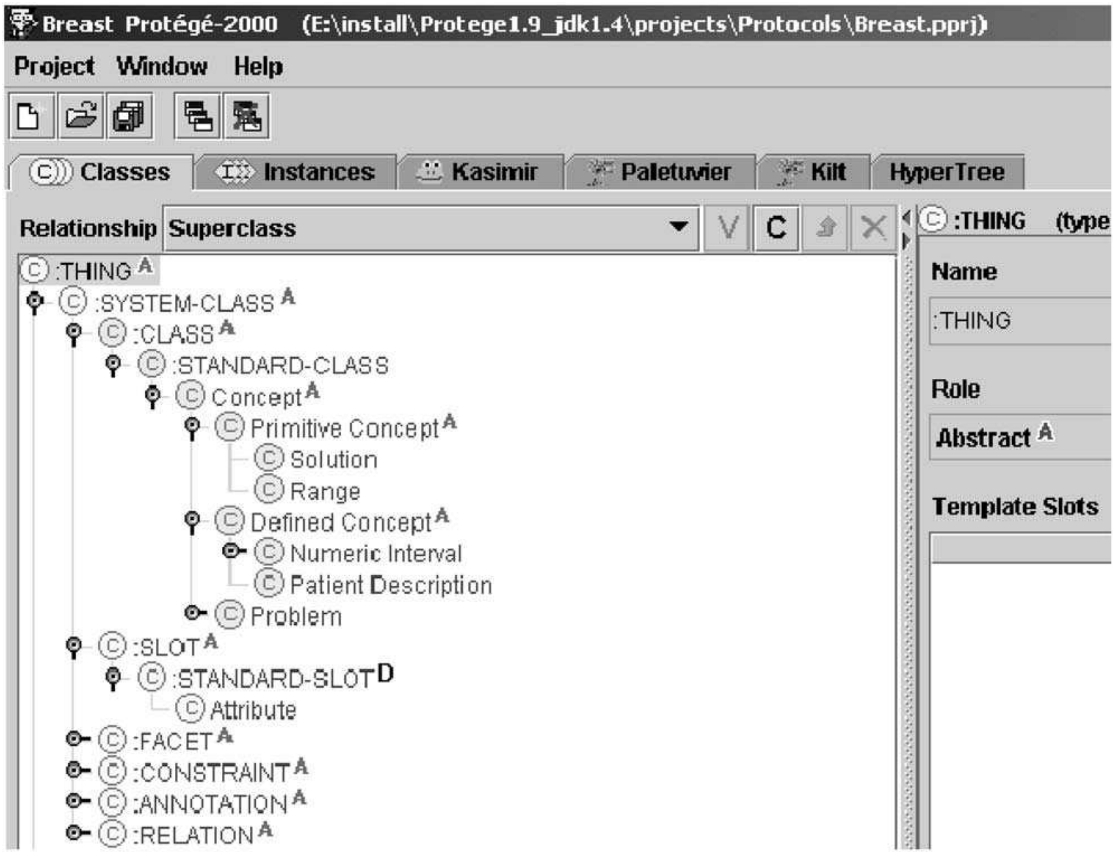


Fig. 2. Specialization of PROTÉGÉ metaclasses. The KASIMIR knowledge representation model is described in the metaclasses architecture by specializing :STANDARD-CLASS and :STANDARD-SLOT. Then corresponding forms can be designed in the "Forms" tab-widget to ease the editing process.

PROTÉGÉ	class	slot	facet	ontology
KASIMIR	concept	attribute	constraint on values	knowledge base

Fig. 3. Correspondence between KASIMIR and PROTÉGÉ notions.

editing process of KASIMIR concepts and attributes. Fig. 3 summarizes this correspondence between KASIMIR and PROTÉGÉ notions, that will be used in the rest of the paper.

3.2. Connection with the KASIMIR reasoner

At this point, the knowledge editing can be carried on in the following way. A knowledge base relative to a given protocol can be edited thanks to PROTÉGÉ and then exported for the KASIMIR decision support system. The connection to this tool of the inference engine has been made for avoiding errors during the editing process.

During a session, the knowledge base is edited in the tabs "Classes" and "Instances" of Protégé. In particular, this leads to the construction of a specialization hierarchy of classes, denoted by $\mathscr{H}_{\text{Protégé}}$, by *asserting* the subclass links. For example, $\mathtt{W}_{\geqslant 16}$ ="Woman of 16y or more" and $\mathtt{W}_{40-80}$ ="Woman between 40y and 80y" are direct specializations of the class W ="Woman" (see Fig. 4(a)). Then, after clicking on the Kasimir tab, the underlying plugin translates all the Protégé classes into Kasimir concepts, all the Protégé slots into Kasimir attributes, and all the facets associated with these slots into values of the corresponding attributes. Concepts, attributes and values are communicated to the Kasimir reasoner which builds a hierarchy of concepts, $\mathscr{H}_{\text{Kasimir}}$, according to the *calculated* subsumption relation based on the attribute values (see Section 2.2). Finally, this hierarchy is visualized in Protégé (see Fig. 4(b)). It may occur, as Fig. 4 shows, that the two hierarchies—the declared one $\mathscr{H}_{\text{Protégé}}$ and the calculated one $\mathscr{H}_{\text{Kasimir}}$—are not isomorphic, i.e. not in a one to one correspondence. This means that there are edges of $\mathscr{H}_{\text{Protégé}}$ that do not match edges of $\mathscr{H}_{\text{Kasimir}}$ (e.g., $\mathtt{W}_{40-80} \longrightarrow \mathtt{W}$) and/or, conversely, edges of $\mathscr{H}_{\text{Kasimir}}$ that do not match edges in $\mathscr{H}_{\text{Protégé}}$ (e.g., $\mathtt{W}_{40-80} \longrightarrow \mathtt{W}_{\geqslant 16}$). The mismatches are indicated by warnings (seeFig. 4(b)). Indeed, our hypothesis is that each specialization link of Protégé must correspond to a subsumption link in Kasimir and vice versa. In practice, these warnings have proven to be useful during an editing of a knowledge base for Kasimir. It must be noticed that the connection between Protégé and Kasimir can be likened to the currently developed OWL plugin for Protégé (Knublauch, 2003). This OWL plugin is still under development at the time of writing; in the prospects of a semantic portal for Kasimir (see Section 5) the joint use of Kasimir and this plugin is planned.

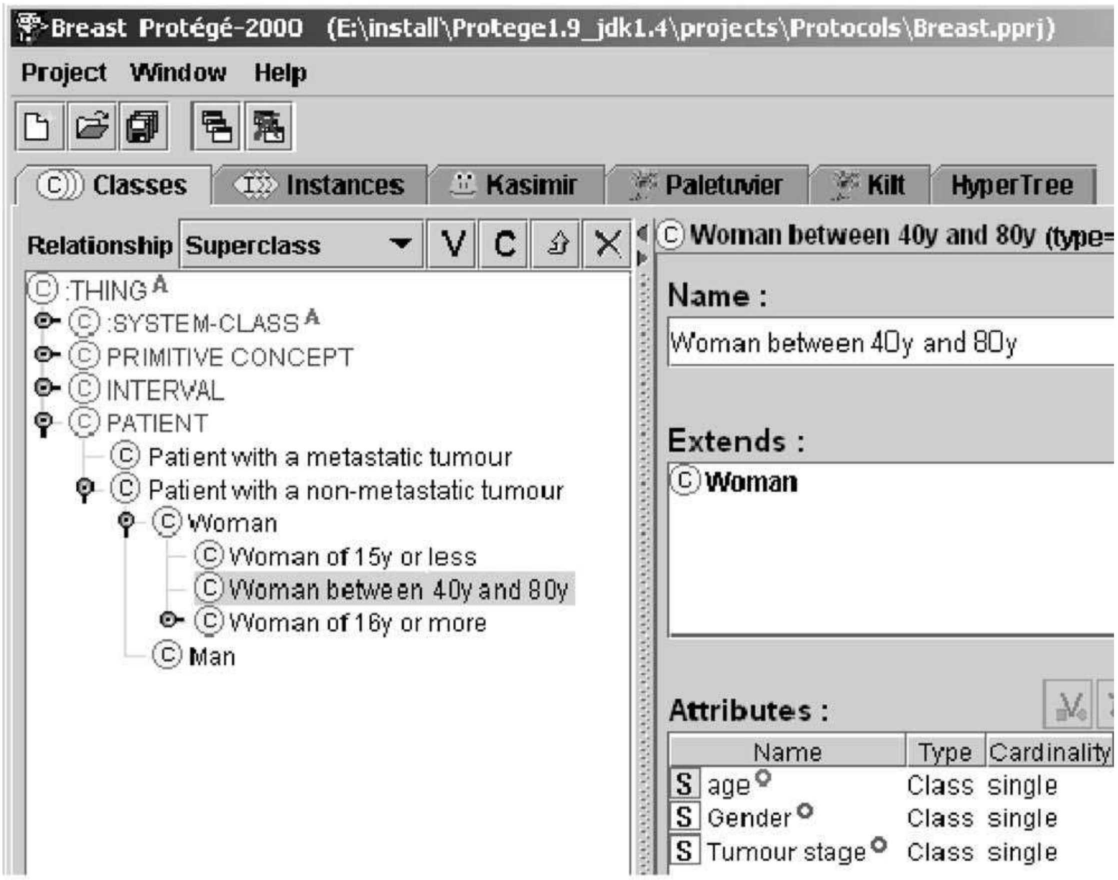
Another help to class editing that has been made possible thanks to the connection with the Kasimir reasoner is the detection of *redundant* classes, i.e. classes that are equivalent but with different names. For instance, let us assume that the two classes $\mathtt{W}_{\geqslant 16}$ and $\mathtt{W}_{>15}$ are edited with the following definitions:

$$\mathtt{W}_{\geqslant 16} = \text{Woman with } \mathtt{age} \geqslant 16$$
$$\mathtt{W}_{>15} = \text{Woman with } \mathtt{age} > 15$$

Since the age is given by an integer, the Kasimir reasoner finds that these two classes are translated into equivalent concepts of Kasimir in the sense that they denote the same set of individuals: $\mathtt{W}_{\geqslant 16}$ subsumes $\mathtt{W}_{>15}$ and $\mathtt{W}_{>15}$ subsumes $\mathtt{W}_{\geqslant 16}$ (for the sake of simplicity, we denote in the same way a Protégé class and the corresponding Kasimir concept). Thus, this is a class redundancy and this is notified by a warning.

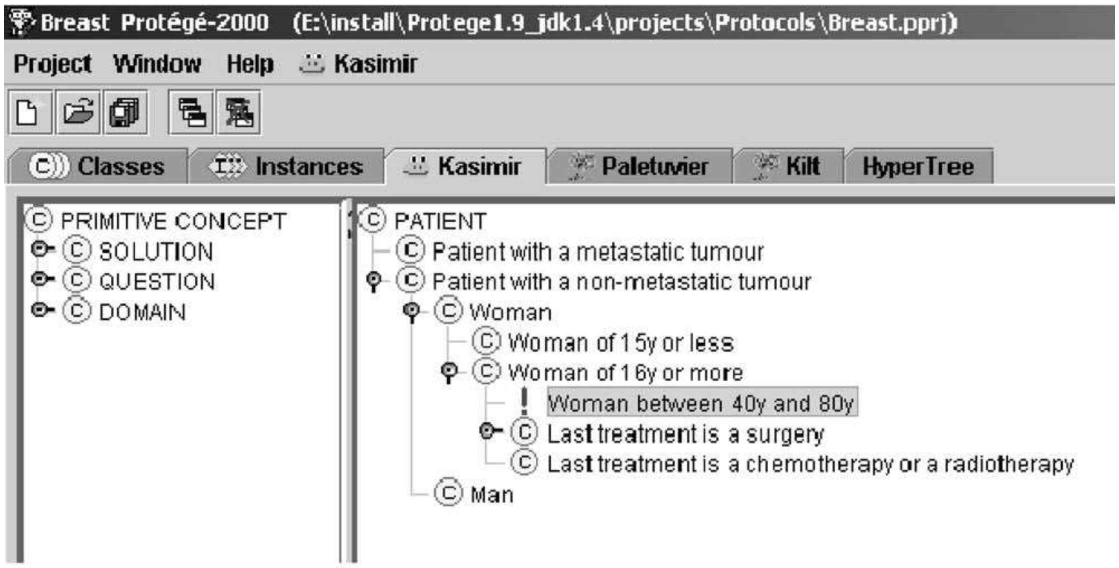
3.3. *Kilt: a maintenance tool for comparing knowledge base versions*

During an update (or a revision) of a Kasimir knowledge base, the need for automatically comparing the old base $\mathtt{KB}_{\mathtt{before}}$ (before the update) and the new base $\mathtt{KB}_{\mathtt{after}}$ (after the update) has appeared. A module comparing versions has to highlight what has been actually updated, to check whether the modifications are in accordance with the intents of the knowledge engineer.


Breast Protégé-2000 (E:\install\Protege1.9_jdk1.4\projects\Protocols\Breast.pprj)
Project Window Help
Classes Instances Kasimir Paletuvier Kilt HyperTree
Relationship Superclass
V C
:THING
:SYSTEM-CLASS
PRIMITIVE CONCEPT
INTERVAL
PATIENT
Patient with a metastatic tumour
Patient with a non-metastatic tumour
Woman
Woman of 15y or less
Woman between 40y and 80y
Woman of 16y or more
Man
Woman between 40y and 80y (type=
Name :
Woman between 40y and 80y
Extends :
Woman
Attributes :

Name	Type	Cardinality
age	Class	single
Gender	Class	single
Tumour stage	Class	single

(a)


Breast Protégé-2000 (E:\install\Protege1.9_jdk1.4\projects\Protocols\Breast.pprj)
Project Window Help Kasimir
Classes Instances Kasimir Paletuvier Kilt HyperTree
PRIMITIVE CONCEPT
SOLUTION
QUESTION
DOMAIN
PATIENT
Patient with a metastatic tumour
Patient with a non-metastatic tumour
Woman
Woman of 15y or less
Woman of 16y or more
!
Woman between 40y and 80y
Last treatment is a surgery
Last treatment is a chemotherapy or a radiotherapy
Man

(b)

Fig. 4. The KASIMIR reasoner highlights the mismatches between the hierarchy edited in PROTÉGÉ and the hierarchy calculated in KASIMIR. (a) Specialization hierarchy of PROTÉGÉ; (b) Subsumption hierarchy of KASIMIR (with a warning denoted by a "!").

This module, called Kilt, has been implemented and integrated into Protégé. Kilt enables to make a partitioning of the problems represented in $\texttt{KB}_{\texttt{before}}$ and/or $\texttt{KB}_{\texttt{after}}$ in four parts:

(1) The problems that appear in the two bases, with the same solutions;
(2) The problems that appear in the two bases, with different solutions;
(3) The obsolete problems, appearing in $\texttt{KB}_{\texttt{before}}$ but not in $\texttt{KB}_{\texttt{after}}$;
(4) The new problems, appearing in $\texttt{KB}_{\texttt{after}}$ but not in $\texttt{KB}_{\texttt{before}}$.

Recall that a particular problem $\texttt{pb}$ of a Kasimir knowledge base is described by a concept denoting a set of patients, and is possibly associated with a solution $\texttt{Sol}(\texttt{pb})$, i.e. a treatment, thanks to a rule $(\texttt{pb} \longrightarrow \texttt{Sol}(\texttt{pb}))$ (see Section 2.2).

From an algorithmic point of view, it is easy to make a partitioning of the different problems in this way, thanks to the use of the Kasimir reasoner. For example, the new problems in category (4) can be found in the following way. Each problem $\texttt{pb}_{\texttt{after}}$ of $\texttt{KB}_{\texttt{after}}$ is classified in the hierarchy of $\texttt{KB}_{\texttt{before}}$, which enables to check whether there is a problem $\texttt{pb}_{\texttt{before}}$ of $\texttt{KB}_{\texttt{before}}$ that is equivalent to $\texttt{pb}_{\texttt{after}}$, i.e. $\texttt{pb}_{\texttt{after}} \sqsubseteq \texttt{pb}_{\texttt{before}}$ and $\texttt{pb}_{\texttt{before}} \sqsubseteq \texttt{pb}_{\texttt{after}}$. If this is not the case, then $\texttt{pb}_{\texttt{after}}$ is a new problem. The three other categories of problems—(1)–(3)—can be found in a similar way. This shows that the implementation of Kilt is simple, once the connection with the Kasimir reasoner is done.

The result of this partitioning can be visualized using the hierarchy visualization module Palétuvier described further—in Section 4—, with a different color for each type of problem (see Fig. 5).

Kilt is used in Protégé in the following way. During a session, $\texttt{KB}_{\texttt{before}}$ corresponds to the state of the knowledge base at the beginning of the session, and $\texttt{KB}_{\texttt{after}}$ to its current state. Therefore, the Kilt module enables to visualize the editing modifications, i.e. addition or removal of a problem, and association of another solution to an already known problem, at any time of the session.

Kilt can be compared to PromptDiff, an algorithm for comparing ontology versions, based on a set of matching algorithms (called *matchers*) (Noy and Musen, 2002). Both tools enable to differentiate what has changed from what has not in the two versions of a knowledge base. The main difference between PromptDiff and Kilt is that the former is based on a purely syntactic approach, whereas the latter is based on semantics. More precisely, all the PromptDiff matchers described in Noy and Musen's (2002) work at a syntactic level: they are based either on the tree structure of the two ontology versions or on the names of the slots and classes. Conversely, Kilt makes comparisons at a semantic level: two concepts match when they have equivalent definitions, based on their attribute values and on the subsumption relation between classes. The main drawback of Kilt is that it assumes that the attributes—and their names—do not change from one knowledge base version to another, whereas PromptDiff can match two different slots. On the other hand, if two concepts are matched by Kilt, whatever their names or their positions in their respective hierarchies are, then they are proven to be equivalent, whereas the PromptDiff matchers are based on heuristics.

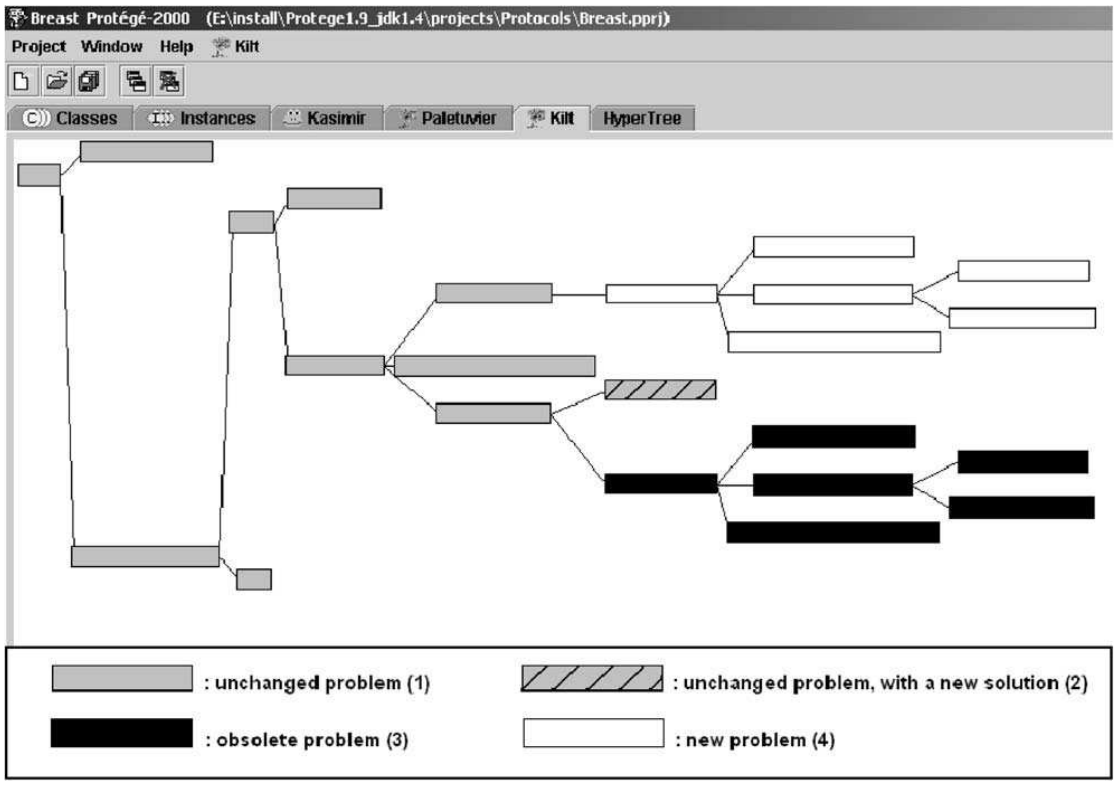


Fig. 5. Visualization of a coloured hierarchy of problems (each problem is coloured according to its status wrt the old knowledge base and the new one).

An interesting study would be to combine Kilt and PromptDiff, to have profit of the strengths of both. One way to do this could be to use Kilt as one of the PromptDiff matchers. Another way of combining them could be first to apply PromptDiff between $\texttt{KB}_{\texttt{before}}$ and $\texttt{KB}_{\texttt{after}}$, second, to use these differences to build a knowledge base $\texttt{KB}'_{\texttt{before}}$ (obtained by substitution of the attribute names in $\texttt{KB}_{\texttt{before}}$, according to PromptDiff matchings) and third, to apply Kilt on $\texttt{KB}'_{\texttt{before}}$ and $\texttt{KB}_{\texttt{after}}$.

4. Knowledge visualization

This section is about knowledge visualization, which is an important part in the conception of a knowledge editing environment. Our requirements for a visualization module are presented in Section 4.1 and the three visualization techniques meeting, when used together, these requirements, are explained in Section 4.2.

4.1. Roles of knowledge visualization

The main task of a visualization module integrated into a knowledge editing environment like Protégé is to help knowledge engineers to evaluate and to validate the result of the editing process. The knowledge visualization module in the Kasimir system should fulfill the following requirements (based on recommendations given in Shneiderman, 1997 and Card et al., 1999):

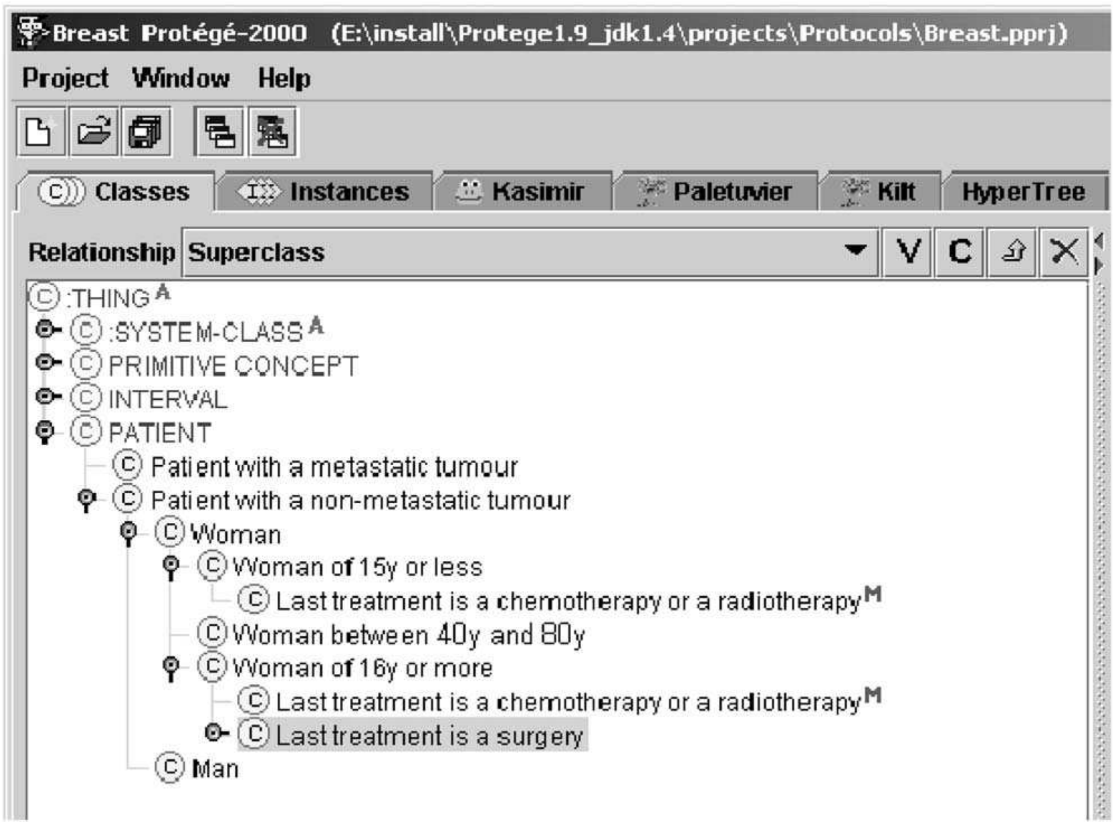
- *Navigation*: Knowledge visualization should allow a quick and easy cover of the concept hierarchy, so that users could browse the concept hierarchy. Users should be able to navigate into the hierarchy, for example to search for a specific concept (maybe to modify it) or to seek for its relationships with other concepts. Moreover, users should be able to follow its own path into the hierarchy.
- *Global view*: Knowledge visualization should allow an estimation of some characteristics of the whole knowledge base. It can be its size, its general shape, its structure, etc. As explained by the Gestalt Theory (Ware, 2000), the human visual system is especially sensible to the global view of a set of elements. Such a global view puts elements back into their context and allows for their comparison. Moreover, it allows users to see properties of the set that are not the sum of the elements' properties. A global view enables to visualize relations and patterns that are not explicitly described in the knowledge base.
- *Accuracy*: Knowledge visualization should reflect the *precise* information contained in the knowledge base. For example, a concept having multiple subsumers must be shown as a single node having several super-class relations, just as it is represented and used by the reasoner. Thus, each concept of the knowledge base must correspond to exactly one node of the visualization hierarchy and vice versa, and each subsumption link must correspond to exactly one edge and vice versa. This is needed for correctly understanding the system behaviour that is based on this knowledge.
- *Usability*: Knowledge visualization should be easy to use for different users. The way of viewing hierarchies must be suited to what the users intuitively are expecting to see. This can be achieved by using classical layouts for hierarchical visualization or layouts frequently used in the application domain to share knowledge (e.g. between an expert and a knowledge engineer).

Unfortunately, there is no known visualization technique meeting all these requirements. Therefore, our approach relies on a combination of visualization modules, each fulfiling one or more of these requirements, and on the integration of these modules in a common environment, thanks to the Protégé plugin architecture. Three visualization techniques are currently used: the first one is provided by Protégé, the second one, called Palétuvier, has been developed specifically for Kasimir, and the last one is a hyperbolic tree visualization based on HyperTree, a free API (Bouthier, 2003). An important point here is that each of these three visualization modules must be fully integrated to the global editing environment in order to provide a coherent view of the knowledge hierarchies, according to the current state of the knowledge base, to the two other visualization modules, and to the Kasimir reasoner running in the background. This coherence is needed for offering a working and complementary combination of these visualization techniques during the editing process.

4.2. Combining visualization techniques

For knowledge navigation and manipulation, Protégé includes a first hierarchical visualization facility for classes as a classical tree widget (see Fig. 6). The main advantage of this visualization is that it is well-known by the majority of users. Nevertheless, it is insufficient whenever visualization needs like navigation and global view have to be taken into account. Indeed, the number of visible nodes in this kind of widget is very small. When having a huge hierarchy (and this is usually the case in medical domains), it is impossible to appreciate the whole set of nodes at once. Moreover, only a restricted view is provided during navigation. A last important problem of this tree visualization is that nodes with multiple ancestors are duplicated. This way of showing multiple inheritance brings additional difficulties for navigation and does not accurately express the content of the knowledge base.

The second visualization module used is called Palétuvier (see Fig. 7). This module can be used to visualize any kind of hierarchies, meaning that Palétuvier can deal with multiple inheritance without node duplication. Another particularity of Palétuvier is that it has been developed specifically for Kasimir. This module shows hierarchy with a layout that recalls the decision tree-like structures used during the knowledge acquisition process with the experts in oncology. Moreover, Palétuvier includes a zoom feature enabling large as well as



```
:THING A
  :SYSTEM-CLASS A
  PRIMITIVE CONCEPT
  INTERVAL
  PATIENT
    Patient with a metastatic tumour
    Patient with a non-metastatic tumour
      Woman
        Woman of 15y or less
          Last treatment is a chemotherapy or a radiotherapy M
        Woman between 40y and 80y
        Woman of 16y or more
          Last treatment is a chemotherapy or a radiotherapy M
          Last treatment is a surgery
      Man
```

Fig. 6. The tree widget visualization provided by Protégé.

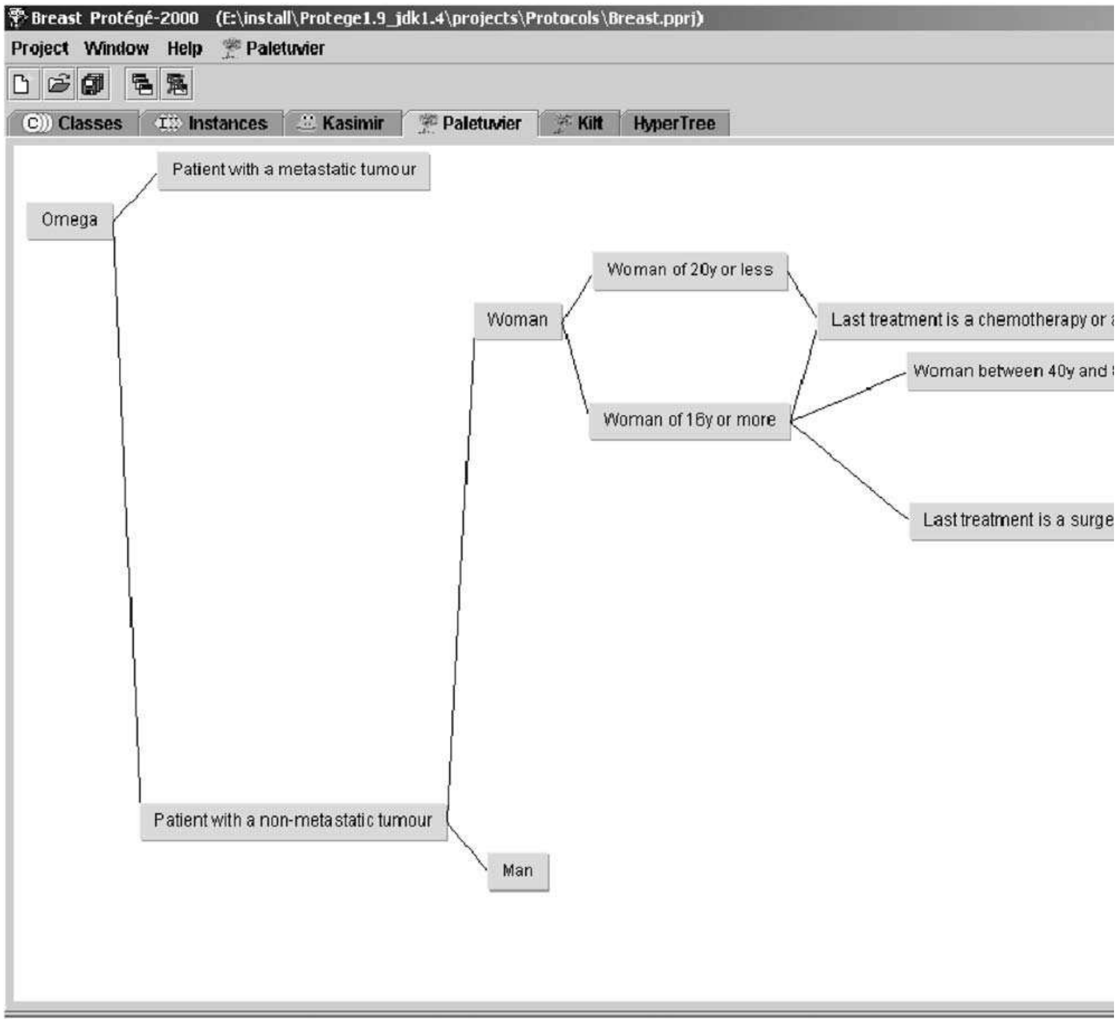

Breast Protégé-2000 (E:\install\Protege1.9_jdk1.4\projects\Protocols\Breast.pprj)
Project Window Help Paletuvier
Classes Instances Kasimir Paletuvier Kilt HyperTree
Omega
Patient with a metastatic tumour
Patient with a non-metastatic tumour
Woman
Man
Woman of 20y or less
Woman of 16y or more
Last treatment is a chemotherapy or
Woman between 40y and
Last treatment is a surge

Fig. 7. The PALÉTUVIER visualization.

detailed views. This functionality corresponds to a suggestion given in Shneiderman (1997): “Overview first, zoom and filter, then details on demands”. Nevertheless, when using PALÉTUVIER, navigation can be made difficult because of the small number of visible nodes in the screen.

The two previous visualization modules do not provide any satisfactory solution for navigation. The hyperbolic tree visualization (Lamping and Rao, 1996) is known to be the most efficient with respect to navigation (Pirolli et al., 2001). This kind of visualization relies on principles of hyperbolic geometry and of the Poincaré model to map a whole hierarchy on a simple plane disc, with a kind of “fish-eye” effect (see Fig. 8). Hyperbolic tree visualization is a “focus + context” technique, in which it is possible to see details on elements of the hierarchy (focus) while keeping a view of the place of these elements in the whole hierarchical structure (context). Another advantage of the “fish-eye” effect is that, even if some nodes are too compressed to be seen, the hyperbolic tree visualization enables much more nodes to be seen than a classical tree visualization, thus allowing a better global view. This visualization

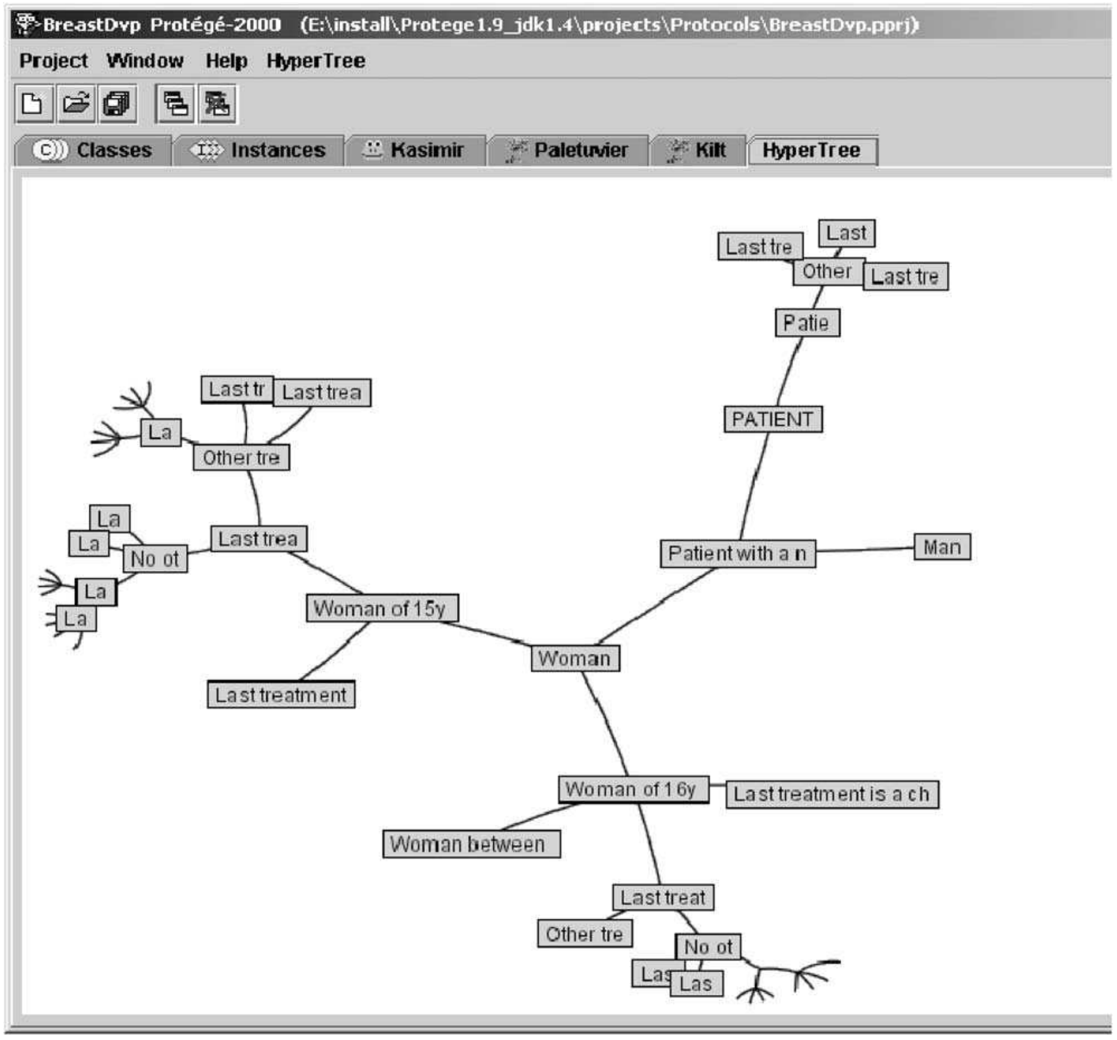


Fig. 8. The HyperTree visualization.

technique is integrated in Protégé for knowledge visualization, thanks to the HyperTree open source API described in (Bergé and Bouthier, 2003) and (Bouthier, 2003). Note that HyperTree is dedicated to the visualization of trees, and then deals with multiple inheritance by duplicating nodes.

Each of these three visualization techniques is specialized in a subset of the four knowledge visualization requirements described in the previous section (see Fig. 9). The module provided by Protégé is classic and known by the majority of users: system developers, knowledge engineers, experts, physicians, etc. It is daily used by millions of Windows users to navigate into their files and drives. Palétuvier gives a precise and coherent view of the hierarchy with respect to the knowledge representation model, taking into account multiple inheritance. Moreover, this visualization module is well-known by knowledge engineers developing the Kasimir system. HyperTree provides a good support for navigation through knowledge hierarchies. Palétuvier and HyperTree both supply partial solutions

	navigation	global view	accuracy	usability
PROTÉGÉ	2	1	1	3
PALÉTUVIER	1	2	3	2
HYPERTREE	3	2	1	1

Fig. 9. Contribution of each visualization technique (3: excellent, 2: adequate, 1: weak).

for global view. Each of these two modules enables the visualization of the whole set of elements and of the structure of the knowledge base, but cannot be used to compare these elements.

It can be noticed that the approach described here—consisting in the combination of several "specialized" techniques for knowledge visualization—is similar to what is developed in JAMBALAYA (Storey et al., 2001). Indeed, JAMBALAYA is a PROTÉGÉ plugin that adapts different visualization tools from software development to knowledge editing. The use of JAMBALAYA in the KASIMIR framework would have been complex because knowledge visualization for KASIMIR has to match its requirements and constraints, e.g. link with the reasoner, specific knowledge model, habits of the users, etc.

The PALÉTUVIER visualization module is also similar to the ONTOVIZ plugin (Sintek, 2003) and the TGVIZ plugin (Alani, 2003). The ONTOVIZ plugin uses the general graph drawing GRAPHVIS library, which allows the representation of any graph and provides several graph layout algorithms. The TGVIZ plugin uses the dynamic graph drawing TOUCHGRAPH library, which allows graphs to dynamically layout themselves, following the spring graph layout algorithm. In the present case, the PALÉTUVIER visualization has been designed for the specific needs of the KASIMIR system. This module gives—at the moment—full satisfaction to users and developers, and thus has been kept within the system.

Finally, it should be noted that, for the moment, no cross-link has been done between the different visualizations. This functionality, allowing users to jump from a view to another with the same node selected, is envisioned in a future version of the system.

5. The steps towards a semantic portal architecture for the KASIMIR system

At present, the KASIMIR system can be considered as a knowledge-based system, relying on a specific knowledge representation language (with an XML serialization) and a specific inference engine (see Fig. 1). The purpose of the KASIMIR project is the diffusion of standard knowledge in order to improve health-care practice in oncology. The main usage scenarios involve the KASIMIR system for physicians during consultations or as a basis for deliberation between experts.

The local architecture of the KASIMIR system has shown a number of problems. Knowledge contained in oncology protocols evolves frequently, and new protocols are created. The KASIMIR system has to provide a direct, updated, and intelligent access to the current knowledge, for geographically distributed users. Moreover, the

Kasimir system has to take advantage of other knowledge sources and tools, for oncology as well as more general domains. This leads to standardization requirements for both knowledge and software engineering within the Kasimir system. For these reasons, the Kasimir system is currently evolving into a semantic portal architecture (see Fig. 10). A semantic portal is a Web server used to supply and share knowledge and intelligent services, for a particular domain, thanks to ontologies and other semantic Web technologies (Maedche et al., 2003).

The standardization of the knowledge components of Kasimir is based on the OWL language (Web-Ontology (WebOnt) Working Group, 2004) for representing the protocols (this standardization is currently under development). The OWL Lite part of OWL is sufficient for the purpose of this representation. Indeed, classes of patients are defined by conjunctions of restrictions (mainly existential quantifications) on properties corresponding to patient characteristics (age, localization and size of the tumor, etc.). Links between classes of patients and the corresponding solution classes (treatments) are made using inclusion of primitive concepts. The characteristics of a patient description are defined as a *necessary and sufficient condition* of membership to a class of patients. The corresponding treatment is then a *necessary condition* associated to the class of patients (stated in terms of the latest version of the OWL plugin for Protégé (Knublauch, 2003). This can be likened to a rule attached to a class of patients. Protocols are described according to a common ontology, named KasimirOnto, describing the basic classes (patient, treatment, etc.) and the relations between these classes.

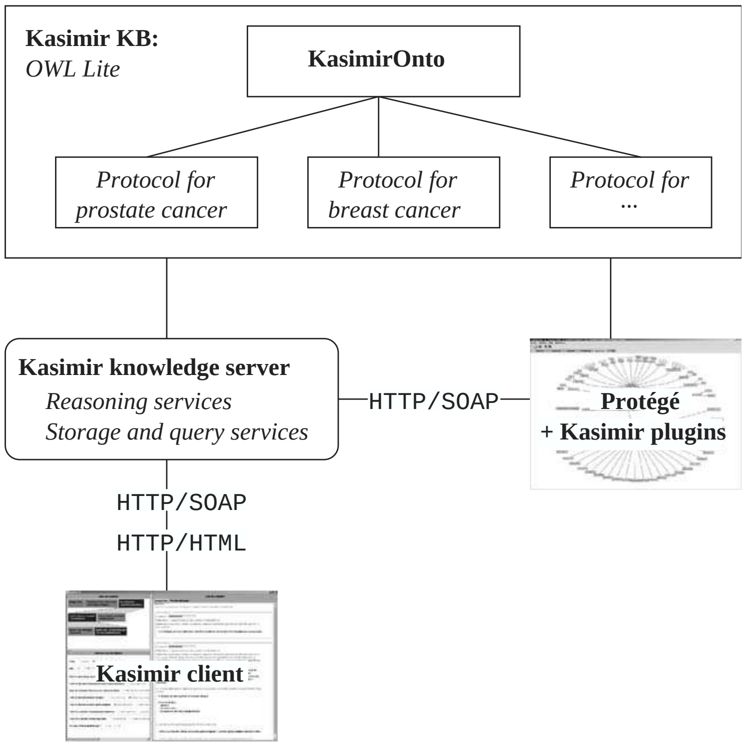


Fig. 10. Evolution of the Kasimir architecture towards a semantic portal (compare with Fig. 1).

Protégé has an important place in the design of this architecture (see Fig. 10). The KasimirOnto ontology is based on different knowledge sources, with different representation languages. These sources include ontologies from the Web and protocols described with the XML serialization of the Kasimir representation language. Editing and maintaining the ontology and the formalized protocols is assisted by tools developed for the Kasimir system, and by other tools provided by Protégé, e.g. Prompt, PromptDiff and the OWL plugin. Finally, the storage and reasoning features of the Kasimir system are implemented within a knowledge server, based on Web services and SOAP communication (XML Protocol Working Group, June 2003). This server implements classification in the OWL Lite representation of the protocols is used within Protégé for controlling the editing task, and for providing an intelligent access to protocols for distant users.

6. Conclusion

The Kasimir system is aimed at knowledge management in oncology. The main objectives within the design of the system are to acquire, represent, and expand knowledge, for allowing a better exploitation and diffusion of knowledge in oncology. According to these objectives, the architecture of the Kasimir system relies on a set of modules for knowledge editing, representation and reasoning, visualization and maintenance. The knowledge editing is taken in charge by the Protégé system, that has been adapted to the Kasimir system and that is connected with the Kasimir reasoner. Knowledge visualization is an important issue in the conception of a knowledge system, and this is why a particular attention has been paid to it for Kasimir. Indeed, three complementary visualization modules have been integrated in the Protégé environment, for achieving different tasks, namely navigation, global and local views. The knowledge maintenance is aided by a specific module that compares two versions of a knowledge base, and that reports the modifications.

The Kasimir system can be considered as a knowledge system evolving towards a semantic Web knowledge system, taking into account the semantic Web principles and techniques. The system will be embedded within a semantic portal architecture, and will take into account knowledge management on the Web, as well as knowledge diffusion and Web services in the domain of oncology. This semantic Web architecture is currently under development, with the idea of becoming a generic architecture for a semantic Web knowledge system.

Another future work is related to the interest, evaluation and acceptability of the Kasimir system within the medical domain. A set of tests have already been carried out, giving positive results and showing that the experiments with the Kasimir system have to be continued (Rios et al., 2003). Moreover, some present functionalities of the system, especially the three visualization modules, still must be concurrently tested and used in actual medical configurations. In addition, the knowledge evolution and the interoperability problems raised by the semantic portal architecture have to be studied more deeply for an evaluation on a real-world basis. Moreover, the research work on the KASIMIR system is continuing, offering a number of theoretical and practical issues, especially in the framework of the semantic Web, that the authors consider as one of the most interesting aspects to be investigated.

References

Alani, H., 2003. TGVizTab: an ontology visualisation extension for Protégé. In: Proceedings of Knowledge Capture (K-Cap'03), Workshop on Visualization Information in Knowledge Engineering, Sanibel Island, Florida, USA.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (Eds.), 2003. The Description Logic Handbook. Cambridge University Press, Cambridge, UK.

Bechhofer, S., Horrocks, I., Goble, C., Stevens, R., 2001. OilEd: A Reasonable Ontology Editor for the Semantic Web. Lecture Notes in Computer Science, vol. 2174, pp. 396–408.

Bergé, B., Bouthier, C., 2003. Mathematics and algorithms for the hyperbolic tree visualization. Technical Report, A05-R-023, 15, September.

Bouthier, C., 2003. Hypertree Java Library, http://hypertree.sf.net/.

Brachais, S., d'Aquin, M., Lieber, J., Napoli, A., 2003. Vers un Web sémantique en cancérologie. In: Journée Web sémantique médicale, WSM 2003. Rennes, www.wsm2003.org.

Broekstra, J., Kampman, A., Harmelen, F.V., 2003. Sesame: an architecture for storing and querying RDF data and schema information. In: Fensel, D., Hendler, J., Lieberman, H., Wahlster, W. (Eds.), Spinning the Semantic Web. The MIT Press, Cambridge, MA, pp. 197–222.

Card, S.K., Mackinlay, J.D., Shneiderman, B., 1999. Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann, Los Altos.

Evidence-based medicine working-group, 1992. Evidence-based medicine. A new approach to teaching the practice of medicine. Journal of the American Medical Association 17, 268.

Fensel, D., Hendler, J., Lieberman, H., Wahlster, W. (Eds.), 2003. Spinning the Semantic Web. The MIT Press, Cambridge, MA.

Haarslev, V., Möller, R., 2001. Description of the racer system and its applications. In: Goble, C., McGuinness, D.L., Möller, R., Patel-Schneider, P.F. (Eds.), Proceedings of the 2001 International Description Logics Workshop, pp. 46–55.

Horrocks, I., 1998. Using an expressive description logic: FaCT or fiction? In: Cohn, A.G., Schubert, L., Shapiro, S.C. (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98). Morgan Kaufmann, Los Altos, pp. 636–647.

Knublauch, H., 2003. OWL Plugin: A Semantic Web Ontology Editor for Protégé, http://protege.stanford.edu/plugins/owl/.

Lamping, J., Rao, R., 1996. A Focus+Context technique for visualizing large hierarchies. Journal of Visual Languages and Computing 7 (1), 33–55.

Lieber, J., Napoli, A., 1998. Correct and complete retrieval for case-based problem-solving. In: Prade, H. (Ed.), Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98), Brighton, UK. Wiley, Chichester, pp. 68–72.

Lieber, J., d'Aquin, M., Bey, P., Bresson, B., Croissant, O., Falzon, P., Lesur, A., Lévêque, J., Mollo, V., Napoli, A., Rios, M., Sauvagnac, C., 2002. The Kasimir project knowledge management in cancerology. In: Proceedings of the Fourth International Workshop on Enterprise Networking and Computing in Health Care Industry, HealthCom 2002, pp. 125–127.

Lieber, J., d'Aquin, M., Bey, P., Napoli, A., Rios, M., Sauvagnac, C., 2003. Adaptation knowledge acquisition and modeling, a study for breast cancer treatment. In: Artificial Intelligence in Medecine Europe (AIME'03), October 2003, Springer, Berlin, pp. 304–313.

Maedche, A., Staab, S., Stojanovich, N., Studer, R., Sure, Y., 2003. SEmantic PortAL: the SEAL Approach. In: Fensel, D., Hendler, J., Lieberman, H., Wahlster, W. (Eds.), Spinning the Semantic Web. The MIT Press, Cambridge, MA, pp. 317–359.

Napoli, A., Laurenço, C., Ducournau, R., 1994. An object-based representation system for organic synthesis planning. International Journal of Human–Computer Studies 41 (1/2), 5–32.

Nebel, B., 1990. Reasoning and revision in hybrid representation systems. Lecture Notes in Computer Science, vol. 422. Springer, Berlin.

Noy, N.F., Musen, M.A., 2002. PromptDiff a fixed-point algorithm for comparing ontology versions. In: Proceedings of the 18th National Conference on Artificial Intelligence and 14th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-02). AAAI Press, Menlo Park, CA, USA, pp. 744–750.

Noy, N., Fergerson, R., Musen, M., 2000. The knowledge model of Protégé-2000: combining interoperability and flexibility. In: Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management, EKAW 2000, pp. 17–32.

Pirolli, P., Card, S.K., Wege, M.M.V.D., 2001. Visual information foraging in a focus + context visualization. In: Proceeding of the ACM Conference on Human Factor in Computing Systems (CHI-01). ACM Press, Seattle, pp. 506–513.

Rios, M., Desandes, E., Bresson, B., Klein, I., Demange, V., Bey, P., 2003. Référentiels de bonnes pratiques cancérologiques : étude comparative de trois supports d'aide à la proposition thérapeutique pour les cancers du sein et de la prostate en lorraine. Bull Cancer 90 (4), 363–370.

Sauvagnac, C., 2000. La construction de connaissances par l'utilisation et la conception de procédures. Contribution au cadre théorique des activités métafonctionnelles. Thèse d'Université, Conservatoire National des Arts et Métiers.

Shneiderman, B., 1997. Designing the User Interface, third ed. Addison-Wesley, Reading, MA.

Sintek, M., 2003. OntoViz Tab: Visualizing Protégé Ontologies, http://protege.stanford.edu/plugins/ontoviz/ontoviz.html.

Stefik, M., 1995. Introduction to Knowledge Systems. Morgan Kaufmann Publishers Inc., San Francisco, CA.

Storey, M.-A., Musen, M., Silva, J., Best, C., Ernst, N., Fergerson, R., Noy, N., 2001. Jambalaya: interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. In: Workshop on Interactive Tools for Knowledge Capture (K-CAP 2001). Victoria, BC, Canada.

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., Wenke, D., 2002. OntoEdit: Collaborative Ontology Development for the Semantic Web. In: Proceedings of the First International Semantic Web Conference 2002 (ISWC 2002), June 9–12, 2002, Sardinia, Italia. Lecture Notes in Computer Science, Vol. 2342, Springer, Berlin, p. 221.

Ware, C., 2000. Information is Visualization. Morgan Kaufmann, Los Altos.

Web-Ontology (WebOnt) Working Group, February 2004. Web Ontology Language (OWL) Reference Version 1.0. W3C Recommendation, http://www.w3.org/TR/owl-ref.

XML Protocol Working Group, June 2003. SOAP Version 1.2 Part 0: Primer. W3C Recommendation, http://www.w3.org/TR/soap12-part0/.