Like Pedagogical Design, Educational System design can Benefit from a Framework-Ooriented Approach

Ruddy Lelouche, Tho Toan Ly

Like pedagogical design, educational system design can benefit from a framework-oriented approach

Ruddy Lelouche*

LIRMM, CNRS, Montpellier II University,
161 rue Ada,
34392 Montpellier Cedex 5, France
E-mail: ruddy.lelouche@gmail.com
*Corresponding author

Tho Toan Ly

Department of Computer Science and
Software Engineering,
Laval University,
Que. G1K 7P4, Canada
E-mail: thotoanly@gmail.com

Abstract: Instructional designers and teachers, at all levels of education and training, try to put in common, to share and to reuse the practices affecting their pedagogical design and expertise. However, the instructional environment, and in particular the design and development of educational systems, can benefit from similar knowledge sharing, especially because of the difficulty, length and cost of this development process. In this paper, we aim to show how building educational systems can be more effective and beneficial using a framework-oriented approach. To do so, we describe different levels of abstraction in the design and the development of an Intelligent Tutoring System (ITS). We first recall what an ITS is, showing its advantages and drawbacks, and identifying different knowledge types. To tackle the limitations of the ITS construction, we then propose to build a framework, including guidelines and tools to ease its development. We finally show that a collaborative framework-oriented approach is indeed feasible, describing how to build such a framework. We thus try to promote reusability and extensibility in many aspects of the design and development of ITSs.

Keywords: architecture; collaborative design; framework; Intelligent Tutoring System; ITS; knowledge component; knowledge type; problem-solving; reusability.

Reference to this paper should be made as follows: Lelouche, R. and Ly, T.T. (2008) 'Like pedagogical design, educational system design can benefit from a framework-oriented approach', *Int. J. Continuing Engineering Education and Life-Long Learning*, Vol. 18, No. 3, pp.323–337.

Biographical notes: Ruddy Lelouche is an Engineer from École Nationale Supérieure des Télécommunications in Paris, and holds an MS and a PhD in Engineering and Computer Science from the University of California at Berkeley. He worked as a Systems Engineer, as a Research Engineer and was responsible for the users' educational training, at the Laboratoire d'Informatique pour les Sciences de l'Homme of the C.N.R.S. (Centre National

de la Recherche Scientifique) in Paris. There, he was one of the chief designers and implementers of RESEDA, an expert system to process biographical data (1979–1984). From 1984 to 2004, he has been a Professor at the Université Laval in Quebec City, Canada, where he created three advanced courses, on knowledge representation and expert systems, on natural language processing, and on communication and research in sciences. Since 2005, he has been an Associate Researcher at LIRMM, the Laboratoire d'Informatique, de Robotique et de Microelectronique de Montpellier of the CNRS in Montpellier, France. His research interests include knowledge representation, the design and implementation of knowledge-based systems, intelligent tutoring systems and interactive learning environments, intelligent computer-human interfaces, in particular using natural language processing and/or multimedia and hypermedia techniques. His research led to numerous publications in international journals and conferences, and has been sponsored by the NSERC (Natural Science and Engineering Research Council) of Canada and by the FCAR Fund (Fonds pour la Formation des Chercheurs et l'Aide à la Recherche) of the province of Quebec. He was also a Coordinator of the database activities at the former *AFCET* in Paris (1982–1984), a founding member of *GIRICO*, the Inter-university Research Group in Cognitive Informatics for Organisations in Quebec (1985), and a member of the coordination committee of the special interest group in software engineering of the F.I.Q. (Fédération de l'Informatique du Québec, the Quebec data-processing federation). In 2003, he founded the research team AMICA (Interactive Acquisition and Modelling of Knowledge for Learning), and since 2003, he has represented Université Laval at the GRITI (Groupe Interuniverstaire de Recherche sur les Tuteurs Intelligents), which organises the biennial ITS International Conference, which will celebrate its 20th year in 2008.

Tho Toan Ly, originally from Viet-Nam, has done his Master's thesis work under the supervision of Ruddy Lelouche. Since 2007, he has been working as a Computer Analyst and Developer in San Diego, California, USA.

1 Introduction

It is unquestionably profitable for instructional designers and teachers, at all levels of education and training from elementary school to continuing education, to put in common, to share and to reuse the practices affecting their pedagogical design and expertise. However, we think that this knowledge sharing can also be extended to the instructional environment, at least when it uses technology, i.e. to the design and development of educational systems, especially because of the difficulty, length and cost of this development process.

This paper aims to show how building educational systems, more specifically Intelligent Tutoring Systems can be more effective and beneficial using a framework-oriented approach.

To do so, we describe different levels of abstraction in the design and the development of an Intelligent Tutoring System (ITS). Section 2 recalls what an ITS is, shows its advantages for the student and the drawbacks of its construction, and identifies different knowledge types. To tackle drawbacks of ITS construction, Section 3 proposes to build a framework, including guidelines and tools to ease its development. Section 4 shows that a collaborative framework-oriented approach is indeed feasible, and describes

in more detail how to build such a framework. Throughout our research and as we show in this paper, we try to promote reusability and extensibility in many aspects of the development.

2 Intelligent Tutoring Systems

Before dealing with the framework and its advantages, in this section we present: What is an ITS? How is it useful to the students? What are the types of knowledge that it incorporates? What does its architecture look like? and Why does its construction bring problems?

2.1 Intelligent Tutoring Systems are useful

An ITS is basically a tutoring system that is designed and developed using artificial intelligent techniques. As to the tutoring domain, it is a particular type of expert system. Indeed, an ITS has some expertise in the domain that it is expected to teach. As such, it can reason on the subject matter, it can solve problems and it can explain or show a trace of its inferences (Sleeman and Brown, 1982; Lelouche, 1998). Thanks to these designed capabilities, an ITS can coach students in problem-solving; examples are Guidon in medicine (Clancey, 1987), Andes in physics (Gertner and VanLehn, 2000), TIGE in cost engineering (Morin, 1998), or the various tutors developed at Carnegie Mellon University (Anderson, 1986). Thus, ITSs were proven to be highly effective as learning aids and many ITS research and development efforts are still taking place (El-Sheikh and Sticklen, 1998).

2.2 Three knowledge types

In their recent research on an ITS in cost engineering, Lelouche and Morin classify the different learning domains into three groups, depending on the type of knowledge which is to be acquired by a student: know, know-how and know-how-to-be (Morin, 1998). Domains such as cost engineering, geometry and physics are categorised in the know-how type or problem-solving type, since these domains require the student to learn how to solve domain problems. The authors state that the knowledge in an ITS in such a domain consists of three types: Domain Knowledge (DK), Problem-Solving Knowledge (PSK) and Tutoring Knowledge (TK).

Domain Knowledge is of the know type. It consists of static, descriptive knowledge to be acquired by the student, and it can be modelled in the system as concepts, relations, facts, rules, etc.

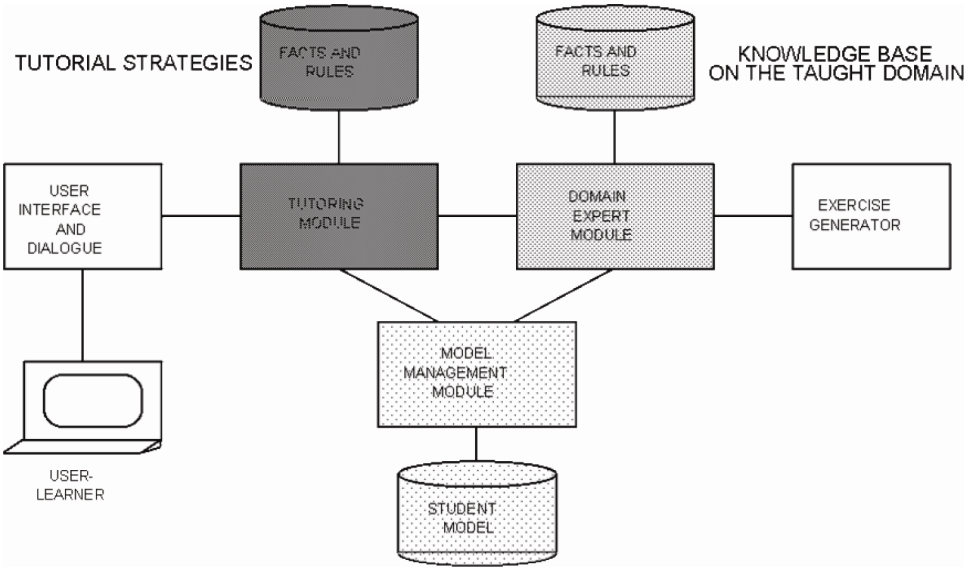
Problem-Solving Knowledge is of the know-how type. It consists of dynamic, procedural knowledge, including all the computational and inferential processes that may be used to solve a problem in the domain to be acquired by the student.

Tutoring Knowledge contains all tutoring entities enclosed in the ITS. It is not directly related to the teaching domain or to problem-solving, for it is not to be taught to or acquired by the student, but it mainly helps him/her to understand, assimilate and master more efficiently the domain knowledge included in DK and PSK. It happens to include also some know-how-to-be knowledge, the quality of which will influence the quality and appropriateness of the actual tutoring strategies and actions.

2.3 Architecture of an ITS

The development of an ITS can be inspired by the architectural views of many existing systems. An example is the one shown on Figure 1, called in 1987 educational psychologist ITS. For a short presentation of ITSs, see Lelouche (1998).

Figure 1 General architectural view of an ITS



Source: Lelouche (1998).

2.4 The construction of ITSs brings problems

He spite of their effectiveness as learning aids, few tutoring systems have made the transition to reach the commercial market so far, although there are notable exceptions, like Anderson's Math Tutor (Clay, 1998). One main reason for this failure to deliver is that the traditional development of ITSs is difficult, time-consuming, and costly (El-Sheikh and Sticklen, 1998): many tasks and procedures are repeatedly programmed and performed from scratch. By identifying these recurring procedures and standardising them, we can define a common ground of development for all ITSs, and benefit from it. That is what this paper aims to show.

3 An ITS framework proposal

To circumvent these problems, for easing the design and development of an ITS, we thus propose to use a framework, to provide a common ground of all development aspects, including knowledge definition and reusability in both the descriptive knowledge and the programming components. The following subsections present the advantages and limitations of using a framework, the three conceptual levels of an ITS using a framework, the ITS framework as guidelines and the ITS framework as tools.

- The abstract level, which simply accounts for the three knowledge types identified in Section 2.3 (they are colour-coded) in the colour version: yellow for DK, maroon for PSK, and green for DK.
- The structural level, which magnifies the abstract view and incorporates the architectural modules identified in Section 2.4, showing their relationships to the knowledge types.
- The components level, which magnifies the structural view, showing the ITS components and making explicit their relationships with the framework components that are used.

3.1 Advantages and limitations of using a framework

The proclaimed advantages of using a framework are thus reusability, extensibility and flexibility, and, as a consequence, an expected shorter development timeline. One early development framework, although it was not named framework then, was the Knowledge Acquisition and Documentation Structuring (KADS and later CommonKADS) methodology for the development of knowledge-based systems (Schreiber et al., 1994), of which Unified Problem-Solving Method Language (UPML) is a long-term development (Fensel et al., 1999).

To give a more complete picture, we find it fair to note also that these advantages are partly offset by well-known disadvantages, listed by Fayad and Schmidt, (1997): development effort, learning curve, what they call integratability, maintainability, validation and defect removal, and lack of standards. However, our opinion is that these disadvantages are worth being tackled to ease the design and development of ITSs.

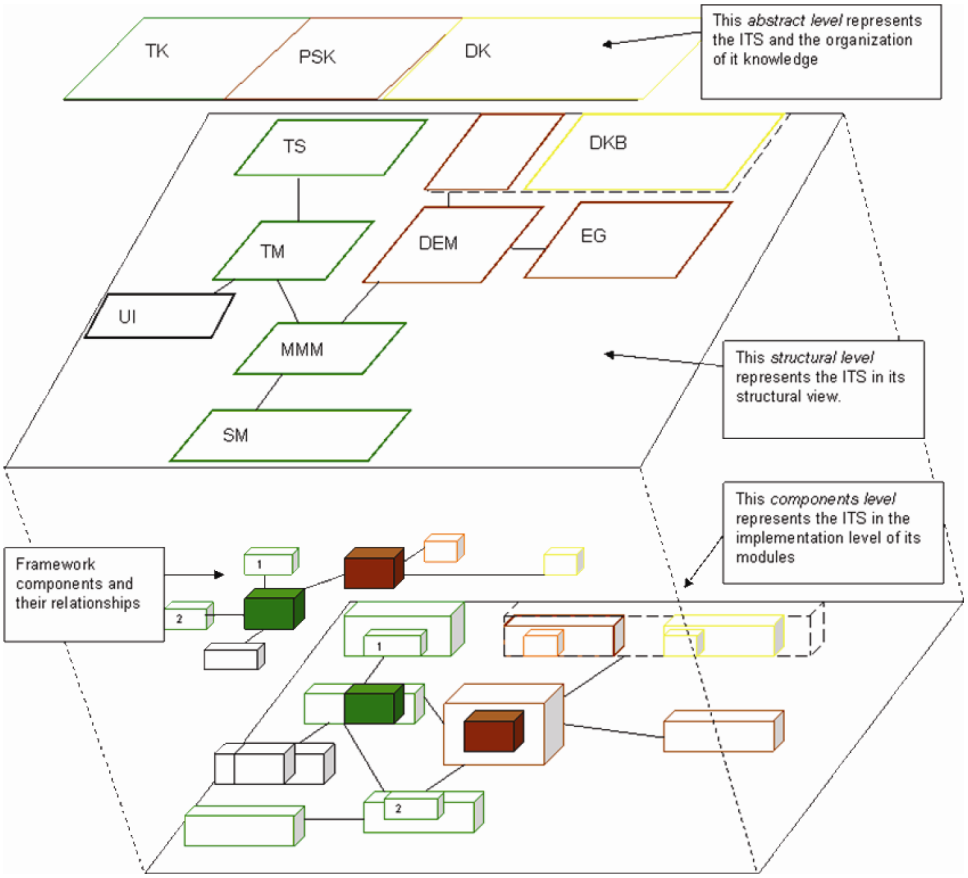
3.2 Three conceptual levels of an ITS using a framework

Figure 2 shows three conceptual levels of an ITS using a framework. For more details, see (Lelouche and Ly, 2003).

Thus, our proposed framework, that we call an Intelligent Tutoring Systems Framework (ITS-FW), is an attempt both to define a common model for ITS development and to provide an application framework (Booch, 1991). That large project involves many research areas, such as knowledge representation, knowledge sharing, knowledge-based development, object-oriented frameworks, user interfaces, etc. To foster reusability, we are trying to reuse concepts found in many existing technologies from different domains: object-oriented development, distributed applications, etc. Their efficiency and strength have been proven useful in real-world systems (OMG, 2003–2006; Microsoft, 2004–2007).

In short, our ITS-FW comprises tools and guidelines to facilitate all facets of the development in its various stages. The following subsections present these two aspects of the framework in more detail.

Figure 2 The three conceptual views of an ITS using a framework (see online version for colours)



Note: At each level, the colours correspond to the three basic types of knowledge found in an ITS: yellow to DK, maroon for PSK and green for TK. The acronyms in the abstract view and in the structural view denote, respectively, the basic knowledge types identified in Section 2.3 and the main modules identified in Figure 1.

3.3 The ITS framework as guidelines

First, an ITS-FW is a set of guidelines. These can be used to define a Common Knowledge Representation (CKR) scheme, to define a structure to locate and access a specific piece of domain knowledge, to design and implement an ITS using the ITS-FW components, and to define a way to communicate between components.

3.3.1 Defining a Common Knowledge Representation

The CKR is a flexible and extensible scheme to define and represent any domain knowledge. It is based on the Extensible Markup Language (XML) notation, and is a subset of the existing pre-defined semantic web markup language (WWW Consortium, 1994–2007). In February 2004, the World Wide Web Consortium (W3C) released the Resource Description Framework (RDF) to complement XML, and the Web Ontology

Language (OWL), as W3C official recommendations. The RDF is used to represent information and to exchange knowledge in the web. The OWL is used to publish and share sets of terms called ontologies, supporting advanced web search, software agents and knowledge management. Thus, the RDF/XML language is well-defined, flexible and easy to use. As a result, it supports different knowledge structure types, at the meta-knowledge level as well as at the various implementation knowledge levels. Knowledge and meta-knowledge can thus be represented in different ways: procedures, concepts, production rules, propositions, frames and scripts, semantic networks, etc.[1] Another benefit of the RDF/XML notation is that, ideally, being a simple human-readable language, it allows different stakeholders such as domain experts or computer agents to read and express their knowledge, if they are willing to do so, without needing a knowledge engineer. Thus, our framework CKR provides a mean to define, reuse and extend knowledge or meta-knowledge. Finally, the CKR also facilitates knowledge sharing: the knowledge in any given domain can be exchanged from one ITS to another.

3.3.2 Defining a structure to locate and access a specific piece of domain knowledge

Our proposed framework assumes that domain knowledge is spread throughout a distributed environment such as the world wide web, and attempts to account for that distribution. Thus, it needs a structure to locate and access a specific piece of domain knowledge from various locations. Without such a structure, knowledge engineers would not be able to access a piece of existing knowledge in one domain to reuse it within another. Although not essential, this ability enhances the capability to exchange and access existing knowledge.

3.3.3 Designing and implementing an ITS using the ITS framework components

The ITS framework provides documentation to permit the integration of ITS-FW programmed components within an ITS. It also documents how to use, extend and personalise many aspects of the development, including the expression of different levels of abstraction for the system design and implementation, for both components and knowledge. This documentation should enable stakeholders such as developers and knowledge engineers to integrate and use ITS-FW features in production.

3.3.4 Defining communication between components

Finally, the ITS-FW defines a way to communicate between components: the Common Component Interaction (CCI). This interaction capability is borrowed from existing similar ones, which are employed in many current technologies such as the Component Object Model (COM) (Microsoft, 2004–2007), the Common Object Request Broker Architecture (CORBA) (OMG, 2003–2006), etc. The CCI is a guideline that specifies how each module should be connected and should communicate with the others. As long and as soon as all involved components conform to this pre-defined specification, any component can communicate with the other existing components within an application. For instance, at a high level of design, the main purpose of using CCI is to integrate relatively easily, into some ITS, components developed and sold by independent vendors. This leads to the ability to construct modules disregarding physical development sites.

Moreover, the CCI scheme provides the flexibility to integrate or substitute components into two or several ITS built for related learning domains. If two domains have some common characteristics, e.g. in user interface presentation, student monitoring techniques or reasoning techniques (Sleeman and Brown, 1982), etc., some components used in one domain can be reused or inserted into another. For instance, if we suppose that geometry is a subset of mathematics, the domain expert module in geometry can be replaced by the one in mathematics (assuming that one exists, which is a major endeavour in itself!) without affecting the overall functionalities of the geometry ITS.

3.4 The ITS framework as tools

For the practical design and implementation aspects of the target ITS, the ITS-FW provides:

1 a set of low-level operations or domain-independent knowledge activities, called runtime knowledge;

2 an object-oriented application framework.

3.4.1 The runtime knowledge

Any problem-solving activity or procedure, whether it is performed by the system or by a human (teacher or learner), includes, at the lowest level, the execution of simple 'atomic' operations such as adding, subtracting or comparing two simple values, displaying a point, a line segment or a number, etc. Such operations are usually necessary in all domains. We call them the runtime knowledge or runtime procedures. This knowledge is part of, and is used at, the execution-level of a problem-solving activity carried out by an ITS. These runtime routines are implemented directly as low-level executable components. We consider the knowledge they implement as domain-independent, but we are not trying here to prove whether all of this knowledge is truly domain-independent.

3.4.2 The object-oriented framework

Jacobson, Booch and Rumbaugh define as an object-oriented framework as a micro architecture that provides an incomplete template for systems within a specific application domain (Gamma et al., 1995; Jacobson, Booch and Rumbaugh, 1999). For example, it can be a system built with the explicit purpose to be extended and/or reused. The benefit of such a framework is the ability to reuse its components (Booch, 1991). In our case, such an object-oriented framework is a part of the overall ITS-FW described in this paper. It is composed of

1 A high level architectural view, which is based on a common component architectural view for all existing ITSs.

2 A set of unfinished programmed components (UPC). The UPC set consists of object-oriented abstracted classes and their relationships. These relationships contribute to the 'core' of software execution and component integration. In addition, the UPCs are developed based on software design patterns to maximise flexibility and reusability. The UPC set is in charge of defining the mechanism used to plug different components onto a main component; it also provides the mechanism for an

ITS to be executed within a given operating system. Because all UPCs are developed in conformity with the CCI, they should be replaced or should evolve without affecting the structure and functionality of the overall ITS-FW.

4 An ITS framework implementation proposal

In this paper, following the separation of the ITS knowledge into three knowledge types, we are proposing a framework that, we believe, should ease the development of ITSs in any teaching domain, because it is based on the reusability, extensibility, flexibility and sharing of knowledge structures, ontologies, tools and techniques. Recall that these activities and goals (design and implementation of the ITS) are above, but connected to, and complement, those dealing with pedagogical procedures and behaviours (use of the ITS). In this section, we focus on the low-level implementation of the framework. How is the separation of knowledge types effected in relationship to the framework? How should the framework be implemented? and Which tools should be included in the framework? In the following subsections, we focus on the implementation of the proposed guidelines (Section 4.1), on the interaction between knowledge components (Section 4.2), on tools to model knowledge and to acquire knowledge entities (Section 4.3) and, lastly, on the ITS environment execution (Section 4.4).

4.1 Knowledge component design and implementation guidelines

In regard to the top view of an ITS, we need to implement the system in such a way that the three knowledge types are clearly defined and independent from one another. Having these principles in mind, we approach our simple architecture of the system by defining a knowledge component for each knowledge type: a Domain Knowledge Component (DKC) for DK, a Problem-Solving Knowledge Component (PSKC) for PSK and a Tutoring Knowledge Component (TKC) for TK. In our context, a knowledge component could and should be more than just one 'programmed' component, since its size would depend on the complexity of the knowledge encoded in a domain. Indeed, a knowledge component has its own ontology, its own defined structures and its own sub-components. It is independent from other knowledge components in the same system.

By organising each type of knowledge as its own knowledge component, it is possible to reuse and integrate such a knowledge component into different systems without knowing the structures and components of the system in which it executes. We can define knowledge representation and an ontology for each knowledge component. For example, we could use a frame representation for domain knowledge, an object-oriented representation for tutoring knowledge, e.g. to build a student model, and a task-oriented representation for problem-solving knowledge. The following subsections describe our approach to the implementation of each of these knowledge components.

4.1.1 Domain Knowledge Component

As mentioned in Section 2.3, the domain knowledge consists of static, descriptive knowledge that, at the low-level of the implementation, is subsequently represented as concepts, relations, facts, rules, etc. When discussing further about our framework,

we proposed CKR (Section 3.1) to provide a mean to reuse and extend knowledge or meta-knowledge. To do so, we need to conceptualise a common set of vocabulary and structures, known as ontology (Mizoguchi and Bourdeau, 2000; Devedzic, 2001), in the teaching domains. We believe that by defining the DK ontology, we can enhance the reusability in many different teaching domains. Within the German Research Center for Artificial Intelligence (DFKI), Ullrich's research group has being focusing on an 'instructional ontology'. Their goal is to provide an ontology to describe a learning resource from an instructional perspective (Ullrich, 2004).

In our research model, we use 'instructional ontology' as a CKR for DK to define any teaching domain knowledge structures. We believe that the modelling of the teaching domain knowledge for an Intelligent Educational System (IES) and for an ITS is very similar: both systems are educational software and contain artificial intelligent components (AAAI, 2000–2007). Extending this small and common ontology, we can easily define, in a teaching domain, a specialised ontology; then we may use it as a basis to acquire knowledge entities: e.g. in physics, Newton's law is a theorem and 'gravitational acceleration is 9.8 ms^{-1}' is a fact.

4.1.2 Problem-Solving Knowledge Component

The PSKC is a programmed component that encodes the problem-solving knowledge. It is responsible for solving a domain problem through a sequence of computational and inferential procedures. Throughout the years, many techniques and programming languages have been used to model this type of knowledge: command-based presentation in imperative programming languages (Fortran, Algol, C, etc.), rule-based presentation in Prolog and LISP, object-based presentation in C++, Java, etc. (Morin, 1998). How it should be implemented is based on the preference of the developers who build the component. In our framework, we simply provide a mean for a component of this type to interact with other components in the same system. In our endeavour, we focused on the UPML (Fensel et al., 2003), a complete framework providing means to reuse and integrate problem-solving methods in the development of a knowledge-based system. First, UPML is a software architecture for knowledge-based systems providing components and adapters, and a configuration of how to connect the components using the adapters. In addition, it is a set of design guidelines specifying how to develop a system constructed from components and connectors (Fensel et al., 2003). Thus, UPML thoroughly defines how components interact and adapt to one another in a knowledge-based system. We believe that it is a solid base that we can use and customise to our ITS framework.

The UPML framework defines not only how to reuse parts or all of the knowledge components, but also how to interact with such components. Moreover, it provides a well-defined ontology for problem-solving methods (Fensel et al., 1999) a descendent of the Common KADS methodology (Schreiber et al. 1994). In the previous sections, we defined our framework characteristics as similar to those of the UPML framework. For instance, our framework purpose is a foster knowledge reuse, to favour a component-oriented development, and to propose guidelines and tools to ease ITS development. That is what the UPML framework does, but for more general knowledge-based systems. In consequence, we believe that, by reusing the UPML framework and specialising it, we may devote less effort to develop our framework, since we start from an existing one, while eventually providing an efficient (because less general) and well-tested framework.

For our PSKC, similarly we reuse the UPML problem-solving method ontology to define problem-solving in our framework. By adhering to this ontology, we believe that the PS knowledge can be reused and extended for any ITS without worrying about how it is encoded or implemented.

4.1.3 Tutoring Knowledge Component

As to the tutoring knowledge, due to the limited time devoted to our work (a master's thesis), we decided not to concentrate on this component, but rather to focus on the teaching domain knowledge component and on the problem-solving knowledge component. Depending on the underlying teaching and learning theories, the TKC could be very complex and very *ad hoc*. However, how this component can or will be built is not our concern here. Besides, we think that building any TKC would not cause any problem in a system that is constructed with our framework, because knowledge components are independent from one another.

4.2 Common Component Interaction

So far, we have three independent knowledge components. Each one does not 'know' anything about the existence and capabilities of other components in the system. It has its own ontology and some protocol communication: namely, a specification of how external components interact with it in a system. Because we do not impose any common communication protocol that all components should adhere to in order to interact with one another, we must define a way to 'map' the ontology of a given component with the ontologies of the external components interacting with it. Thus, a set of mapping relations express the connections between the ontologies of two components (Crubézy, Pincus and Musen, 2003). For that purpose, the UPML framework defines knowledge adapters (Crubézy and Musen, 2004), expressing how to map a domain knowledge component to interact with problem-solving methods in a knowledge-based system. That approach is a great source of inspiration for us to define how to map a DKC to a PSKC in our framework. In fact, we believe that we can reuse the UPML framework to design the mapping for DKC and PSKC, and that we could also propose a similar mapping for TKC to interact with DKC and PSKC.

To help the reader understand this concept better, we show how it works on a simple example. Suppose that our projected ITS consists of a DKC specialised in physics and a PSKC specialised in arithmetic calculation. A simple problem statement is given: '2 km + 2 km = ? km'. The PSKC only knows that addition is the sum of one or many values, which must be numeric. As a consequence, we must map each '2' from the problem statement into a numeric value 2 for PSKC, so that PSKC can do its calculation. When the PSK component terminates its calculation, it will return the numeric value 4, and the mapping will then 'tell' the DSK component that the (physics) result is '4 km'.

The mapping process becomes important if we want two components to communicate with each other. Certainly, defining a mapping is an extra step in the development process. However, its cost should remain reasonable if we have access to many ready-to-use components: the cost of developing a mapping is lower than that of adapting an existing component to make it work with other components. Moreover, our ITS framework does not impose any kind of structure on any of its components.

4.3 Tools to ease the knowledge implementation

To make a new proposal framework for ITSs, we effectively promote reusability and extensibility by applying the existing UPML framework and concepts, and by adding more functionality to it. Thus, our framework becomes an extension of the UPML framework. We take the concept knowledge component and adapter from UPML. One advantage of reusing the UPML framework is that UPML is perfectly integrated with Protégé (Gennari et al., 2003). Protégé is an ontology editor and a knowledge acquisition system developed by a group of researchers at Stanford University. Protégé is a tool, which allows users to construct domain ontologies, to customise data entry forms, and to enter data. It is also a platform, which can easily be extended to include graphical components such as graphs and tables, various media such as sound, images and video, and various storage formats such as OWL, RDF, XML and HTML (Noy et al., 2001).

Thanks to the extensibility built in Protégé, UPML communities have created a plug-in that provides an editor to edit a problem-solving method and an editor to edit its mapping. Thus, we can use those very editors to add and edit our ITS-specific problem-solving methods, and to define their mappings. In addition, Protégé is a perfect tool for modelling our teaching domain ontology: it has already been used successfully to model instructional ontology and UMPL ontology. Thus, with Protégé, we expect to be able to define in a seamless fashion our teaching domain knowledge, be it mathematics, physics or geometry.

4.4 The ITS execution environment

As we mentioned in Section 3.2, our ITS framework provides a runtime knowledge which is a set of low-level operations. At the implementation level, the runtime knowledge is a process that translates a high-level programming language into a machine language. Thus, it takes place in an execution environment where an 'atomic' operation, as mentioned above, can be translated into machine code. Moreover, it is in charge of connecting different knowledge components in an ITS (see Section 4.2), thanks to their mapping relationship(s). However, defining a mapping is not enough: we need to add to the execution environment an interpreter for translating a given knowledge instance from one knowledge component to the others.

The execution environment provides the executed steps of a problem-solving method. The result of each step can be mapped back to some domain knowledge instance. This 'back-mapping' provides a mean, especially for a tutoring component, to examine the reasoning process used, and to suggest a student certain execution steps during a tutorial session if necessary.

In order to maximise reusability, flexibility and extensibility, we carefully design the execution environment using the object-oriented framework concepts. This execution environment should be flexible to replace any programmed component in the environment without corrupting it. It should be reusable to customise any component for the needs of a specific domain. To make it extensible, as mentioned in Section 3.2.3, we are developing a UPC and the associated plug-in mechanism so that any programmed component can be interacted with the environment at runtime.

4.5 In summary

Using our ITS framework to build an ITS, one will need to define the domain-dependent knowledge, the problem-solving knowledge and the mapping knowledge. Because these knowledge are independent from one another, we can always reuse them in the development of an ITS specialising in various domains. As we look back at the example in Section 4.2, we can reuse the PSKC specialised in arithmetic calculation in the development of an ITS in geometry by simply defining the mappings necessary for the DKCs and PSKCs to communicate with one another. Throughout the paper, we have tried to show that our proposal framework relates to many fields such as knowledge representation, knowledge engineering, object-oriented development, etc. During the master's project of the second author, we hope to establish mainly:

- a common ontology for DK and PSK;
- a guideline on how to design DKC and PSKC based on our common ontology and on how to define the associated mappings;
- an ITS environment execution, as mentioned in Section 4.4, that is highly extensible and customisable to meet most, if not all, of the needs to build any type of ITS.

We hope that our proposal framework could be a subject of interest to other researchers, so that it can be carried on and be completed in some near future.

5 Conclusion

As we described our approach to build an ITS, we progressively mapped our design activities into different levels of abstraction. At the highest level of abstraction, which is the ITS knowledge modelling level, we separated ITS knowledge into three distinctive knowledge types. Next, at the design level, we proposed an ITS framework, the purpose of which is to ease the process of ITS development. That ITS framework comprises:

1 a set of guidelines, including common knowledge representation, knowledge sharing and common component interaction, to help us reuse existing knowledge structures and entities;

2 a set of tools, taking care of common operations in all domains, to help us focus on developing and extending knowledge components for a specific domain need.

Lastly, at the implementation level, we proposed instead to use more concrete existing well-proven tools and guidelines to develop the framework. Thanks to the reuse of these existing tools and guidelines, the development of our ITS framework should be effective and cost-saving. Because of their inclusion in our ITS framework, the latter should be well tested, reliable and ready to be used in the development of a real-world ITS.

References

AAAI (2000–2007) *Intelligent Tutoring Systems, in the AI Topics Site*. Menlo Park, California, USA: Association for the Advancement of Artificial Intelligence (formerly the American Association for Artificial Intelligence). Retrieved March 2007. Available at: http://www.aaai.org/AITopics/html/tutor.html.

Anderson, J.R. (1986) *Cognitive Modelling and Intelligent Tutoring*. Washington, DC, USA: National Science Foundation.

Booch, G. (1991) *Object-oriented Design with Applications*. Redwood City, California, USA and Don Mills, Ontario, Canada: Benjamin/Cummings Publ. Co.

Clancey, W.J. (1987) *Knowledge-based Tutoring: the Guidon Program*. Cambridge, Massachusetts, USA: The MIT Press.

Clay, R.A. (1998) 'Math software allows students to subtract fear, multiply confidence – psychologist builds an artificially intelligent math tutor', *The APA Monitor*, Vol. 29, March, p.40. Retrieved March 2007. Available at: http://www.apa.org/monitor/mar98/math.html.

Crubézy, M. and Musen, M.A. (2004) 'Ontologies in support of problem solving', in S. Staab and R. Studer (Eds), *Handbook on Ontologies* (pp.321–342). Heidelberg, Germany: Springer.

Crubézy, M., Pincus, Z. and Musen, M.A. (2003) 'Mediating knowledge between application components', Demo presented at the *Semantic Integration Workshop of the Second International Semantic Web Conference (ISWC-03)*, Sanibel Island, Florida, USA, p.6. Retrieved March 2007. Available at: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-82/SI_demo_02.pdf.

Devedzic, V. (2001) 'The semantic web – implications for teaching and learning', in C.H. Lee, S. Lajoie, R. Mizoguchi, Y.D. Yoo and B. du Boulay (Eds), *Enhancement of Quality Learning Through Information and Communication Technology (ICT), Proceedings of ICCE/SchoolNet 2001 Conference* (pp.29–30). Incheon, South Korea: National University of Education.

El-Sheikh, E. and Sticklen, J. (1998) 'Framework for developing intelligent tutoring systems incorporating reusability', in J. Mira and A.P. Del Pobil (Eds), *Methodology and Tools in Knowledge-Based Systems,* Paper presented in the Proceedings of the *11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-98*. Castellón, Spain, 1–4 June, Vol. I. *Lecture Notes in Computer Science*, Vol. 1415. Heidelberg, Germany: Springer, pp.558–567.

Fayad, M.E. and Schmidt, D.C. (1997) 'Object-oriented application frameworks', *Communications of the ACM*, Vol. 40, pp.32–38.

Fensel, D., Benjamins, V.R., Decker, S., Gaspari, M., Groenboom, R., Grosso, W., Musen, M., Motta, E., Plaza, E., Schreiber, G., Studer, R. and Wielinga, B. (1999) 'The component model of UPML in a nutshell', Paper presented in the Proceedings of the *1st Working IFIP Conference on Software Architectures (WICSA1)*, San Antonio, Texas, USA, February. Retrieved March 2007. Available at: http://users.ece.utexas.edu/~perry/prof/wicsa1/final/fensel.pdf.

Fensel, D., Motta, E., van Harmelen, F., Benjamins, V.R., Crubézy, M., Decker, S., Gaspari, M., Groenboom, R., Grosso, W.E., Musen, M.A., Plaza, E., Schreiber, G., Studer, R. and Wielinga, B.J. (2003) 'The unified problem-solving method development language UPML', *Knowledge and Information Systems*, Vol. 5, pp.83–131.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J.[2] (1995) *Design Patterns: Elements of Reusable Software Architecture*. Reading, Massachusetts, USA: Addison-Wesley.

Gennari, J.H., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubézy, M., Eriksson, H., Noy, N.F. and Tu, S.W. (2003) 'The evolution of protégé: an environment for knowledge-based systems development', *Int. J. Human-Computer Studies*, Vol. 58, pp.89–123.

Gertner, A. and VanLehn, K. (2000) 'Andes, a coached problem-solving environment for physics', in G. Gauthier, C. Frasson and K. VanLehn (Eds), Paper presented in the Proceedings of the *Intelligent Tutoring Systems: 5th International Conference, ITS 2000*, Springer, Berlin, pp.131–142.

Jacobson, I., Booch, G. and Rumbaugh, J. (1999) *The Unified Software Development Process*. Reading, Massachusetts, USA: Addison-Wesley.

Lelouche, R. (1998) 'The successive contributions of computers to education: a survey', *European Journal of Engineering Education*, Vol. 23, pp.297–308.

Microsoft (2004–2007) *COM: Component Object Model Technologies*. Redmond, Washington, USA: Microsoft Corp. Retrieved March 2007. Available at: http://www.microsoft.com/com/.

Mizoguchi, R. and Bourdeau, J. (2000) 'Using ontological engineering to overcome common AI-ED problems', *Int. J. Artificial Intelligence in Education*, Vol. 11, pp.107–121.

Morin, J-F. (1998) *Conception of an Intelligent Tutoring System in Cost Engineering: Knowledge Representation, Pedagogical Interactions, and System Operation*. Master's Thesis. Université Laval, Québec, Canada.

Noy, N.F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R.W. and Musen, M.A. (2001) 'Creating semantic web contents with protégé-2000', *IEEE Intelligent Systems*, Vol. 16, pp.60–71.

OMG (2003–2006) *Catalog of OMG CORBA/IIOP Specifications*. Needham, Massachusetts, USA: Object Management Group Inc. Retrieved March 2007. Available at: http://www.omg.org/technology/documents/corba_spec_catalog.htm.

Schreiber, G., Wielinga, B., Akkermans, J.M., Van de Velde, W. and de Hoog, R. (1994) 'Common KADS: a comprehensive methodology for KBS development', *IEEE Expert*, Vol. 9, pp.28–37.

Sleeman, D. and Brown, J.S. (1982) 'Introduction: intelligent tutoring systems', in D. Sleeman and J.S. Brown (Eds), *Intelligent Tutoring Systems* (pp.1–13). London, UK/Orlando, Florida, USA: Academic Press.

Ullrich, C. (2004) 'Description of an instructional ontology and its application in web services for education', Paper presented in the Proceedings of the *3rd International Semantic Web Conference ISWC 2004*, Hiroshima, Japan, November 7–11.

WWW Consortium (1994–2007) *W3C Semantic Web Activity*. World Wide Web Consortium. Retrieved March 2007. Available at: http://www.w3.org/2001/sw/.

Notes

[1]Note that, although much knowledge is heuristic, this characteristic is not mentioned here: it refers to the deep structure of the considered knowledge and not to some representation technique.

[2]Because of the popularity of their book (it was in its 32nd printing in 2005), these authors are often nicknamed, even in citations, the 'Gang of Four', or 'GoF', or 'Go4'.