On the Social Rational Mirror’s architecture:
Semantics and pragmatics of educational interactions

Daniele Maraschi, Germana M. Da Nobrega, Stefano A. Cerri
{maraschi,nobrega,cerri}@lirmm.fr

LIRMM, Laboratoire d’Informatique, de Robotique et
Micro-électronique de Montpellier
161, rue Ada – 34392 Montpellier – Cedex 2 – France

Abstract

The paper presents the overall architecture developed in two
different educational application contexts allowing to manage
both semantic and pragmatic aspects of student and teacher
interactions with the tutoring system. A methodology will be
exposed along with some other project considerations. All the
discussion will be exploited presenting in parallel two practical
applications.

1. Introduction

Starting from a general overview of existent tutoring systems having an interface on the Web,
we can observe that the methodology used to design them follows mainly the classical
approach of Web site development. We think this kind of approach does not take really in
consideration pedagogical requirements coming from the teaching experience. In this article
we describe our approach taking into account innovative technologies to develop dialogue
based tutoring systems on the Web interface. In such a context, we have developed two
applications deployed in the domain of e-commerce. From the design perspective, both the
applications follow the guidelines of our approach, while from the functional objectives, they
are essentially complementary. The combined solution offers powerful and innovative tools in
the new dialogue oriented Web interaction for pedagogical purposes.

The first application considered was developed in the context of an European Project [1] with
the main goal to provide a course to learn the e-commerce vocabulary in the English language
by students coming from the East of Europe. The ambition of this type of application, as
explained in [2], is manifold: allow to capitalise from dynamically available Web Information,
not just from locally generated learning resources; allow as much as possible Agents to engage
both in dialogues with learners and with teachers-developers-experts that wish to enrich,
modify, customise etc., on the fly the XML documents generating dialogues with learners;
transform the current HTML Web technologies into fully Conversation Oriented, server-based
dynamic technologies in order to multiply the potential use world-wide of produced CALL
material.

The second application is a Web-served learning environment, called PhilInEd, to assist both
the planning and the execution phases of a course. The work within PhilInEd is based upon
educational theories that privilege autonomous learning from experience, by trial and error,
founding the discipline Discovery Learning. The server has been experimented in a student
class on Business Contracts. In a complementary paper [10] we report the experiment. In this
paper we introduce PhilInEd through a hypothetical example, and we present the main
components of its architecture.
2. P.R.I.M.A Architecture overview

We have formalised an architecture, called P.R.I.M.A. (Program-centered Reusable Interactive Multimedia Architecture) describing both the fundamental software components and their organisation. We can identify three main temporal phases: a dynamic Information acquisition meant to maintain a continuously updated information repository in order to produce Knowledge; the structuring of Information into Knowledge phase meant to organise and to provide semantical notations to the Information collected; and the Knowledge reuse phase meant to be the preferential source to build learning materials. In all of these phases, other mechanisms are considered. For example, since all of them have to provide a user interface in order to interact with the user, the organisation of such interfaces is described by the Logic, Content and Style (LCS) methodology (Figure 1) that modularise a single HTML page into three different files, one structuring the page content, the XML, another one structuring the layout on the Web browser, the XSL and the last one structuring the page interaction in Java. The same methodology has already used to develop an XML course starting from the correspondent offline version on CD-ROM [3].

One of the advantages is to associate in one-to-many relations different components: for example, many Web page contents described in different XML files can be associated to the same XSL file providing a uniform layout, or in the other hand, the same content described by an XML file can be displayed in different way on the Web browser. Another mechanism considers the web interface itself (in this case the XML/XSL page) the message structure of the dialogue between the user and the cognitive agent managing the interaction. This means that the Web page have to allow the same expressiveness freedom to the two ways of communication: from user to the agent and viceversa. Moreover, they have to facilitate the pragmatic comprehension by explicitly labelling of such messages.

2.1. Basic framework

Since our intention is to provide an easy access to the platform via the Web, we have chosen HTTP as the basic protocol, but this is not enough since, as stressed in [4], HTTP is a stateless protocol which make difficult to track the user session and hence to remember the history of the user interaction. So we have developed the platform using the Java Servlet framework that allows to deploy applications listening on the same socket port of the Apache web server and, at the same time, consider a sequence of web interaction like a whole user session. This provides the main advantage to build next Web pages using information coming from the past and not only from the last page visited, as in the case of many scripting languages. Session tracking is important both from the pedagogical point of view, for example to monitor the student progression, and from dialogue point of view since it allows to maintain the partner dialogue knowledge, as justified in [5].

The Servlet framework is also used to implement the LCS methodology explained before allowing to mix the Logic component with the Content and Style ones; this mean, in other words, to program dynamically interaction of an XML and/or an XSL page. The XML...
programming allows to manipulate dynamically the page content proposed to the user; for example an XML file structuring an exercise will contain, along with other information, the solutions, but is not really useful to show them immediately to the student. In analogue way, XSL programming allows to program the page layout; this can be useful, for example, to adapt the page content to a particular learning strategy that the system considers more efficient for the student.

2.2. Serendipitous Information acquisition

We state as a precondition that the Course content, view as a set of information, is already available on the Web taking into account also the advantage of Web dynamics allowing to access to a continuously updated information source. For this reason, we have placed some searching Agents, based on the Madkit agent platform [6], collecting the content of Web pages (Figure 2) coming from serendipitous queries to most common web search tools, like Google [7] and Altavista [8], and saved in a database.

![Figure 2. Serendipitous Information search by Agents](image)

We have placed a filtering parser that transform the HTML page in a XML-like tree structure, called DOM, and select only the text content nodes discarding those concerning the images and other multimedia types as shown in (Figure 3).

![Figure 3. Structure parsing of an HTML page](image)
2.3. Structuring Information into Knowledge

All the information collected in HTML pages stored in the database is essentially not structured. For this reason, a person with a particular role, called the Semantic Author, is charged to attribute semantic notations to the information collected. An XML web editor was implemented and presented along with the HTML page preview in order to allow the Semantic Author to do the process manually. The XML web editor, developed in [9], allows to define XML templates describing the component structure of a document; such a structure can be modified immediately in the case the Semantic Author considers it not semantically appropriated for the HTML page content shown on the left-hand side of Figure 4.

For each HTML page a new XML document is generated describing semantically the information. In this way, a consistent set of XML documents is stored in a different database.

---

**Figure 4. The XML web semantic editor**
2.4. Knowledge reuse

Once a significant collection of XML files is set up, it is possible to reuse such Knowledge asking for precise meaningful document components in order to build the content of a new Web page. In this context, a user playing a different role, called the Course Author, is charged to build a new set of XML documents composing a new Course content for the domain of e-commerce. Only the resulting XML pages will be used to deploy the final Web application along with the layout described by XSL files and the user interaction implemented in Java. Of course, in the case of changing the goal, correspondingly also the criteria used to structure the new XML pages necessarily change, but neither the Course Author process or the Semantic Author process described above can be automatically managed by the architecture. This means that human contribution will be always necessary in order to end successfully the process.

3. The PhiInEd application

Figure 5: Phases of a course supported by PhiInEd.

Within the server PhiInEd, two phases of a course are currently taken into account, namely, Planning and Running (Figure 5). Planning a course consists on the elaboration of a Plan by the one who administrates the course, to which we refer as the Teacher. Running a course consists on the execution of the Plan, by the ones who follow the course, to which we refer as the Learners, guided by the Teacher. The result of the work in the Planning phase, called Plan, is composed of a sequence of Lessons to be studied by Learners. Executing a Plan for the Learners means to study the Lessons from the Plan. We consider the existence of a Plan as a requirement for a course to be run within PhiInEd. The result of the work in the Running phase is called a Reasoning Framework, that represents knowledge to be used by a machine in order to assist humans to reason in the corresponding knowledge domain. In a complementary paper [10], we detail each phase, as well as report an experiment carried out with a class of twenty-seven students in Law.
The server PhiInEd have been designed in such a way to support both the individual work and the communication between participants of a course. Therefore, the main components of the server's architecture are: the Framework for Individual Work (FIW) and the Message Manager (MM). Hereafter we detail each one.

3.1. The Framework for Individual Work

The Framework for Individual Work (FIW) supports, as suggested by its name, the work that each participant of a course is supposed to perform by himself. For the Teacher, the FIW supports the elaboration of the Plan during the Planning phase, while for the Learner, the FIW supports the elaboration of the Reasoning Framework, as a result of the study of Lessons from the Plan.

Let us explain, by means of a hypothetical example, the rationale underlying individual work.

Let us suppose that the subject of study for Learners is the so-called *trading conditions*. Learners are instructed to play the role of a lawyer who intends to get assistance from a machine, in order to analyse, verify, and write contracts on trading conditions. According to the business, they are informed that it is necessary to distinguish, for instance, *products* from *services*. Also, products considered *dangerous* are supposed to be treated differently from those considered *normal*, and so on. Let us now suppose that Learners find out that they would need distinguished assistance, in cases they deal with trading conditions for *sellers*, or *buyers*. Maybe buyers of dangerous products intend to impose to sellers the responsibility of the transport. On the other hand, maybe sellers do not want at all lawyers to consider such a constraint when writing their trading conditions. The example is illustrated in Figure 6: the same discourse could be used by lawyers (Learners) to write both sellers' and buyers' trading conditions, but the discourse is constrained differently according to each context of interest (seller or buyer).

![Figure 6](image-url)  
Figure 6: (Hypothetical) Trading Conditions in different contexts: buyers and sellers.

According to the example, the resulting Reasoning Framework is composed of: (i) the terms, in a hierarchy, establishing the discourse, (ii) the contexts that are defined by a number of logical constraints among those terms, and eventually (iii) the documents that are classified according to contexts, and are supposed to respect the constraints of their corresponding contexts.

One important feature of a Reasoning Framework is the dialectics between documents and contexts. From the one hand, existing documents may be used as a source of information in order to define contexts. From the other hand, once they are defined, contexts may "accept" or not arriving documents. By accept we mean that a document verifies the constraints in a
context. Also, one might think in a context as something evolving continuously, possibly due to coming documents. In order to actually assist a human to reason in a domain starting from the Reasoning Framework, a machine should provide him to build and to revise contexts, as well as to verify/create documents with respect to contexts.

At this point, we are able to present how individual work is actually provided by the FIW. This is accomplished by means of FIW's three main sub-components, namely, Terms, Folders, and Documents.

The component Terms provides the edition of hierarchies of labels. Functionalities like the addition/suppression/movement of labels are provided. A label created through this component as a part of a hierarchy, has the same name of the component itself: a Term.

Like the component Terms, the component Folders provides the edition of labels organised as hierarchies. However, a label created through this component as a part of a hierarchy is called a Folder. A Folder represents what we have called above a context. The Folder components provides as well the creation of Constraints, as logical relations among Terms.

![Figure 7: Document Description in Terms Hierarchy.](image)

The component Terms provides the verification/creation of a document with respect to the Constraints in a Folder. One must point out what Terms he has observed as Present or Absent in the case of an existing document, or what Terms he wants to be Present or Absent, in the case of a document being created. Let us recall the example about trading conditions. Suppose that the Terms observed as Present in a certain document are “dangerous” and “Buyers transports”. The verification of such a document with respect to the Sellers’ context from our example is shown in Figure 7. The two Terms are highlighted because the context “rejects” both Terms simultaneously Present. By reject we mean that the Constraints in the Folder are violated (Figure 8).
The list of Terms together with their Present/Absent values is called a Description. In PhiInEd, we are particularly interested in documents from Web pages. The structure that represents a document in the system is called a Document. A Document is then a list composed of a Description, a URL, a Name, and Comments. Name is a string identifying the Document. Comments is also a string. The creation of a Document is provided by the component Folders, since the user must point out in which Folder he wants to classify the Document. The maintenance of a Document is provided by the component Documents.

Data generated through the FIW component are all stored as a tree in an XML file.

3.2. The Message Manager

The Message Manager (MM) is composed of two sub-components, namely, Sender/Receiver and History Manager. The Sender/Receiver is responsible for the composition and sending a message from a participant to another, as well as for retrieving the new messages arriving for a participant. The composition of a message provides the attachment of the XML file with the Reasoning Framework. Sending a message causes the storage of message information in a Repository containing all the waiting messages. Whenever a participant asks the retrieval of waiting messages, the pending messages that are addressed to him are removed from the Repository.

The Sender/Receiver communicates with the History Manager, which is responsible for storing/retrieving the messages of a participant. Whenever a message is sent, message information is also stored in the History of the sender. Whenever a message is received by a participant, message information is stored in his History as well, after being removed from the Repository by the Sender/Receiver. The History of a participant is organised as a sequence of Dialogues (Figure 9).
4. Conclusions

The goal of the paper is to show that enhancing Web technologies we would profit from several opportunities offered in order to:

- use Information available on the Web is a necessary complement to the generation of ad hoc proprietary learning material;

- transform the resulting learning – authoring process into dynamic and conversational ones, in order to bypass the current limitation of a page-centred client dominated view.

In the first application Information is transformed into Knowledge by the creation of XML files in which semantics is organised through XML tags. In the second application Learners use Web pages as a social information source to produce explicitly organised knowledge resulting in a Reasoning Framework. The complementarity of the two applications lies on the fact that in the PhiInEd application the course content is supposed to be achieved in HTML pages composing the Plan. The methodology presented in the first application can be applied in order to compose the content Lessons of a Course.
5. References


