SEWAGO : A Generic Adaptive Educational Hypermedia System

Cédric Jacquiot
Ecole Supérieure d'Electricité (Supélec)
Plateau du moulon
91192 Gif sur Yvette CEDEX
France

Yolaine Bourda
Ecole Supérieure d'Electricité (Supélec)
Plateau du moulon
91192 Gif sur Yvette CEDEX
France

Fabrice Popineau
Ecole Supérieure d'Electricité (Supélec)
2 rue Edouard Belin
57070 Metz
France

<first name>.<last name>@supelec.fr

ABSTRACT

In this document, we explain the main principles and architecture of SEWAGO (SEveral WAys to GO), a generic adaptive educational hypermedia system we are working on. More precisely, SEWAGO is a platform for creating adaptive educational hypermedias, using the strictest and most reusable architecture. This system is based on first order logic and situation calculus (through a modified version of GoLog).

Keywords

adaptive hypermedia, education, golog, first order logic

1. INTRODUCTION

When one wishes to create an adaptive educational hypermedia system (AEHS), one needs to create all the concepts for the adaptation and the modelisations. It would be much simpler to reuse a generic architecture, which is what we wish to create, in order to develop and run AEHSs. Many papers deal with describing one particular AEHS, but much fewer with its generic architecture. Even [1] , which deals with a global approach for creating an A(E)HS, doesn't give many clues for such an architecture.

Our system will only be designed to provide link adaptation, and not content adaptation. The different methods for adapting contents do not seem to be ruled by a common generic layer, whereas link adaptation is merely generic, even if calculation methods may be very different.

2. ARCHITECTURE COMPONENTS

First of all, we consider that an AEHS is made up of three independent components that work together. Those components are respectively the learner model, the document space model, and the inference engine. We try to get the most generic architecture for each of those components. First, we describe the way we represent these components, and then how they work together. However, we always keep in mind the fact that the models for the different components must be coherent with one another in order to co-operate in a non-twisted way.

Our learner model (LM) is based on the conceptual graphs (CG, cf [5]), i.e. we represent the knowledge we have about the learner in the CG formalism. In order to get the best out of the CG, we split our LM into two parts : the scheme and the knowledge base. The scheme describes the different possible conceptual relations, their signature (i.e. their arity and domains), and logic rules that exist between the different relations. Its purpose is to describe the relation semantics as far as possible, in order to provide well-defined, finely reusable conceptual relations. The CG formalism is close to first order logic (FOL) (it is FOL with interesting restrictions, cf [5]), which makes us think that describing rules in logic is a reasonable idea. The knowledge base is made up of a representation of the CG which gathers information about the learner, such as what he knows, who he is, what educational methods he prefers, his examination results, global appreciations, personal information, categorization...

The Document Space Model (DSM) is based on the same principles as the LM, as we represent the document space (DS) in the CG formalism. As for the LM, we split the DSM into a scheme, describing the semantics of the relations as much as possible (through FOL rules), and a documents base, which describes the actual relations between the different learning entities (concepts, chapters, electives...) in the CG formalism. We intend to provide as much freedom as possible in describing the resource's connections with one another. The resources we deal with can be documents, examples, or any kind of learning entities. The desired relations can be described in the scheme and used in the base, as long as the maximum possible semantics is given in the scheme. For example, we can describe which documents are prerequisites for others.

The inference engine is the component of the system that

adapts the path through the documents to the learner. It is based on the situation calculus, and of a set of rules to describe the required way to adapt the path through the documents. The idea of using the situation calculus is found in [4]. We consider the learner's knowledge, his place in the learning path, and the DSM as a situation. Visiting a page, reading it or answering questions are possible primitive actions that modify the situation. In the situation calculus, the actions must be "possible" in order to be actually executed, i.e., some conditions are required to make an action. This notion of possibility is close to its common meaning. We also use an other idea from [4], which consists in including the concept of desirability in the situation calculus. The desirability is more relative to the learner (what is good for him to get access to), whereas the possibility is relative to a "physical" notion of what can be done (one might access all documents if he is online, and those previously downloaded if not). For example, visiting a document can be possible if it is the next document in a linear order, but not desirable if the current document has not been entirely scrolled yet. The rules describing the possibilities and desirabilities are supposed to be FOL rules, in a way very similar to [3].

3. GLOBAL RUNNING OF THE SYSTEM

As we have described our three main components, we now wish to present the way they operate with one another. As we said earlier, our system is mainly based on the CGs and FOL. The rules that are created for adapting the links (e.g. "it is desirable to get a link to a document if it is reachable from the current document in the DS") can easily be described in FOL. It seems clear to us that the simplest way to coordinate the different elements of our system is to create them as layers over FOL. We use GoLog (cf [2]) as an implementation of the situation calculus, and complete it with the notion of desirability described earlier. GoLog is naturally developped in prolog, so we will use the latter to create and coordinate the different elements we tend to provide.

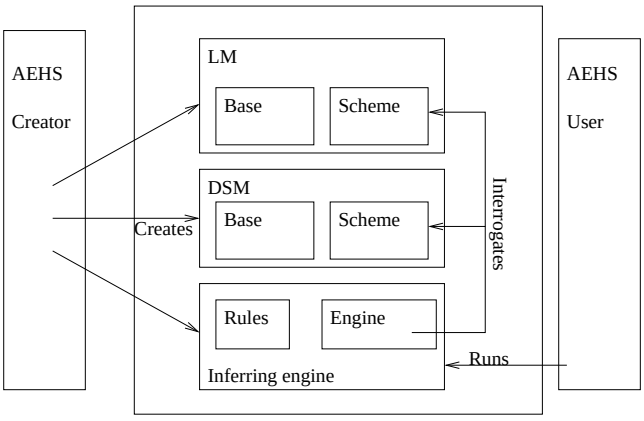

Figure 1: Global Architecture Schema

Fig.1 describes the system architecture and the coordination of the different elements. The rules, which are described in the different schemes, are called by the inference engine in order to be applied. Those rules, when applied, use the corresponding base elements. In other words, the schemes are being used for interfacing the different elements of our system.

Let us see a simple example of our system functioning. We have a document space made up of 4 documents (doc1, doc2, doc3, doc4), doc1 is followed by doc2 and doc3 (no order), and doc4 has doc2 and doc3 for prerequisites. The main rule for the user is that a read document is considered to be known. The one for the adaptation is that all of its prerequisites need to be read in order to get access to a document (i.e. for it to be desirable for the learner). The initial situation for the learner is on doc1. He can access doc2 and doc3 freely in any order from there. He cannot access doc4 until he has read both doc2 and doc3. If he chooses to read doc3 first, he can still access doc2, but not doc4 yet, for he has not read doc2 yet. Then he goes from doc3 to doc2, from where he can finally access doc4. This example, which is of course very simple, shows how adaptation components can provide several ways for going, as the learner could have chosen to go to doc2 before doc3. Moreover, it shows how FOL rules can allow a simple description of the system.

The main functioning steps are the following :

1. The engine calculates the reachable and desirable documents using the rules described for it.
2. The learner is free to make an action (e.g. going to a document, reading it...), which is interpreted as a GoLog primitive action.
3. The situation is recalculated after the action.
4. The rules that need to be checked and applied for the LM are parsed and the necessary modifications in the LM are made.

4. FUTURE WORK

Our goal is now to achieve the implementation of the system we just described. We have decided to fully implement it in prolog, as most of the formalisms we use are based on FOL. We also wish to provide some pre-defined rules and settings that seem to be useful in most AEHS.

5. REFERENCES

[1] P. Brusilovsky. Methods and techniques of adaptive hypermedia. *Adaptive Hypertext and Hypermedia*, pages 1–44, 1995.

[2] F. L. Hector J. Levesque, Raymond Reiter and R. B. Scherl. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.

[3] N. Henze and W. Nejdl. Logically characterizing adaptive educational hypermedia systems. *Proc. of AH2003 Workshop, World Wide Web Conference*, MAY 2003.

[4] S. M. Ilraith. Adapting golog for programming the semantic web. *Proc. of the Eighth International Conference on Knowledge Representation and Reasonning (KR2002)*, APRIL 2002.

[5] J. F. Sowa. Conceptual graphs. http://users.bestweb.net/~sowa/cg/.