Reusability in GEAHS

Conference Paper · January 2004

Source: DBLP

CITATION
1

READS
40

3 authors, including:

Yolaine Bourda
École Supérieure d'Electricité
88 PUBLICATIONS **367** CITATIONS
SEE PROFILE

Fabrice Popineau
Laboratoire de Recherche en Informatique
46 PUBLICATIONS **117** CITATIONS
SEE PROFILE

REUSABILITY IN GEAHS

CÉDRIC JACQUIOT
Supélec, Plateau de Moulon, 91192 Gif/Yvette, France
E-mail: cedric.jacquiot@supelec.fr

YOLAINE BOURDA
Supélec, Plateau de Moulon, 91192 Gif/Yvette, France
E-mail: yolaine.bourda@supelec.fr

FABRICE POPINEAU
Supélec, 57000 Metz, France
E-mail: fabrice.popineau@supelec.fr

GEAHS (Generic Educational Adaptive Hypermedia System) is a platform designed to ease the development of Adaptive Educational Hypermedia, using standard formalisms. In this document, we explain the underlying principles of this platform. Genericity is achieved thanks to an adaptation engine based on situation calculus and RDF. This paper describes the main aspects of our system, as well as the use we make of situation calculus to create a more reusable adaptive hypermedia system.

1 Introduction

Many Adaptive Hypermedia Systems (AHSs) have been developed for educational purpose in the past few years. Most of them have been created for a precise purpose and are not reusable. In other words, these systems have to be build from scratch any time one wants to develop a new teaching platform. P. Bruzilowsky[1] described a methodology for creating an adaptive hypermedia. P. de Bra[2] studied the theoretic and generic ground that should be used for the creation of Adaptive Hypermedia Systems. An engine - AHA![3] -, was even developed, based on AHAM specifications. Our goal has been threefold. First, we wanted to create an engine - based on principles close to those of AHAM - using standard or recommended formalisms, in order to make it as reusable as possible. Second, we wanted to provide a model both simple and powerful. We wish that as many people as possible are able to reuse our system. Even if the creation interface can really ease the AHS creation work, the underlying system must be founded on clear enough grounds to reach a really simple reusability. In order to achieve this, we also provide an extensible set of reusable adaptation rules. This avoids as far as possible the creation from scratch of new adaptation rules. The adaptation is provided with generic built-in rules and metadata, that can be reused (and modified if necessary) by the AHS creators. Third, we wished to prove that situation calculus, introduced by McIlraith[4] for adapting the semantic web, can be applied to the problem of generic adaptive hypermedia. We also wish to show that it is a

simple enough solution for the second goal we want to achieve.

Our engine is based on situation calculus - a calculus based on first order logic (FOL) - for the adaptation and on RDF for data representation. As we shall see, situation calculus allowed us to create an entirely reusable engine based upon an easy-to-understand ground. RDF, as a W3Crecommendation, allows us to easily import/export data to/from this language. Moreover, no meta-information about the documents is stored directly in the documents, it is stored separately.

We shall see the global functioning of our system in part 2 of this document. In part 3, we will show how we use situation calculus and FOL rules to provide an easy-to-reuse system. In part 4, we will see how we wish to improve data reusability in AH systems.

2 Global architecture of GEAHS

This section will provide a global overview of GEAHS architecture, which is made of three parts: the learner model, the domain model and the adaptation component (engine and rules), as suggested by DeBra[2].

Our system is divided into three interacting components. The first one is the learner model (LM), which provides information about the learner, for example his/her name, his/her age, his/her former passed classes... The second one is the domain model (DM), which represents information about the documents that can be recommended to the learner. The third and last one is the adaptation engine (AE), which provides - in our case - link adaptation to help the learner navigate through the documents. Let us describe the information handled by each of these modules.

2.1 The learner model

Our learner model (LM) is based on a conceptual approach of the problem. The different concepts are linked by (conceptual) relations. In order to be able to describe the semantics very precisely, our learner model is divided in two parts: the schema and the knowledge base. The schema lets us describe the properties of the classes and of the relations, whereas the knowledge base contains the actual information about the learner.

This LM allows us to describe the properties of the relations concerning the learner, i.e. the arity of these relations, their range and domain, and possible logic rules that exists between these relations. For example, we can provide a relation date_of_birth to allow us to describe the learner's date of birth. Our representation system allows us to define the class of this relation (learner_relation_type), its arity (2, one learner in, one date out), and the fact that it is related to the relation has_age by the rule $date\_of\_birth(learner, date) \Rightarrow has\_age(learner, current\_date - date)$.

For the moment, we use a simple pre-defined schema. It allows us to

describe the topics studied by the learner, the grade he got when passing the latest examination concerning this topic, and the date of the examination. We are currently working on integrating well-known schemas such as PAPI or LIM in OWL.

2.2 The domain model

The DM is based on the same principles as the LM. It is also split into a schema and a database. The schema allows us, as for the LM, to describe the semantics as far as possible, whereas the base contains the actual data about the domain. Even though we intend to provide as much freedom as possible in describing the resource's connections with one another, we also provide built-in learning objects categories (e.g.: concepts, chapters, electives...).

Most of all we had to choose a DM including the major relation for adapting links: the prerequisite relation. It is a simple yet inescapable fact that a learner must know all the prerequisite of the concepts presented in a document in order to fully understand this document. We are currently testing several DM that we have translated in OWL (LMML[5], INT) in order to determine exactly what relations, what document hierarchy, what concept levels we should use.

2.3 The adaptation component

The adaptation engine is the component of the system that adapts the path through the documents. We have decided to focus on link adaptation, content and presentation adaptation will be part of our future work. Our engine is based on the situation calculus, and on a set of rules to describe the required way to adapt the path through the documents. The idea of using the situation calculus is introduced by McIlraith[4], where it is used for the semantic web. Here, we use it for a closed architecture, with formalisms that do not vary from one page to another and potentially more complex adaptation. As for using FOL for describing rules, the reasons are threefold. First, FOL is a machine-understandable, very powerful and well-known formalism. All other rules systems, which are many in other systems, are founded on FOL but in a non-standard form. Most of the time, they are not even simpler than FOL. Second, situation calculus is based on FOL, which makes communication between the two calculi very natural. Third, transcribing rules from natural language to FOL can be very easy. For example, if one wants to express the following rule: "A document that is read by a learner is considered to be known by this learner", one will use the following FOL rule: $read(user, document) \Rightarrow knows(user, document)$

The rules describing the possibilities and desirabilities are FOL rules which are very similar to those introduced by Henze[6].

In order to actually provide adaptation, we consider the learner's knowledge, his place in the learning path, and the DM, as parts of the current

situation. Visiting a page, reading it or answering questions are possible primitive actions that may modify the situation. In the situation calculus, an action must be "possible" in order to be actually executed, i.e., some conditions are required to do an action. This notion of possibility is close to its common meaning. We also use another idea from [4], which consists in including the concept of desirability in the situation calculus. The desirability is more relative to the objective itself - what is good for the learner to get access to in order to reach his goal, whereas the possibility is relative to the what the learner himself is able to learn - for example, one might (possibility) access tons of documents if one already knows many of them, but should (desirability) only access his goal's prerequisite. As another example, visiting a document can be possible if it is the next one in a linear order, but not desirable if a shorter way leads to the learner's goal.

3 Reusability of the adaptation component

3.1 GEAHS engine

We have written an engine in GOLOG[7], which implements the principles seen in 2.3. Not only is this engine fully reusable for any AH developped reusing GEAHS (i.e. AH with potentially different rules and data), but it also provides a major improvement when compared to similar systems: its functionning is simple to understand.

Many systems work as black boxes. The underlying principle is a large amount of code, designed to use rules in a more or less complex way. Unless you have a complete enough understanding of these systems, they remain difficult to reuse. What we tried to achieve here is the exact opposite of what already exists: a glass box even understandable by people with low computer science skills.

Basically, all that one has to understand is that:

- His rules are applied in the order they are provided to the system;
- Adapted links are calculated with these rules;
- When the user makes an action (clicks a link, reads a page...) the rules are all triggered again, the learner modeled is updated according to these rules and the situation is changed.

Fig. 1 demonstrates, using a simple example based on a database class, how easy to get the inner mechanism of our system is. In situation 1, the learner discovers the desirable lessons for him to take. Links are displayed according to the current knowledge of the learner: If something is already known, it is not desirable, if some document requires knowledge that is not learnt yet, it is neither available. Our leaner first action is to click on the link leading to the page describing the INSERT command. INSERT has been linked because

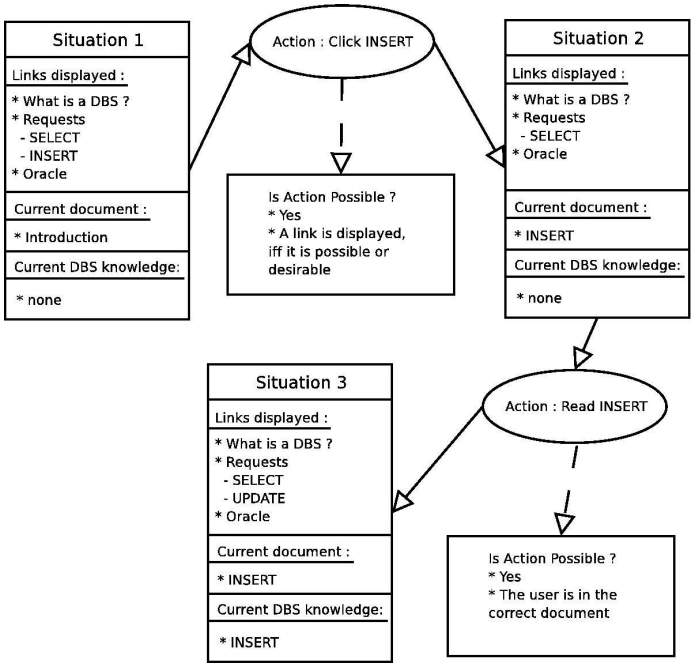

Figure 1. Situation Calculus: an easy mechanism to perform adaptation

it was desirable, therefore, the action of going to the INSERT document is possible. In situation 2, the INSERT link is no longer available, because the current document ıis INSERT.

Then, the learner reads the document. It is a possible action because INSERT is the current document (we assume that being on document is necessarily the consequence of the possibility to access it, i.e. to get to know it). Therefore, we consider that he knows the INSERT-related concepts. This leads to situation 3, where the UPDATE document (which requires the knowledge of INSERT) is available to click, because it is now desirable.

What we just described is not just a model of our system's behavior, it is this way it actually works. It is our assumption that a system based on a very clear but yet powerful mechanism is a more reusable system, for more people can understand its apparent and inner functionning.

3.2 FOL rules

We decided to implement a FOL rule module. Even if FOL is far too expressive considering the needs of an AH system, it allows us to transcribe rules from almost any system, since most systems have subset of FOL as rule languages. This way, it is possible to create modules to translate rules from an external

language to FOL and have thoses rules work with our system.

Moreover, we provide a set of predefined rules with our system. Thoses rules have been implemented for most common use of an educational AH system. They can be used as provided, modified or fully replaced by another set of rules.

Here is an example of the rules we provide:

$$\begin{aligned}\forall Link, \exists element, current\_user(User, Situation) \wedge \\ rdf\_triple(Link, rdf:type, element) \wedge \\ \neg is\_linked(Link, Situation) \wedge \\ (\forall Prerequiste, red\_triple(Link, preq, Prerequisite) \\ \Rightarrow rdf\_triple(User, knows, Prerequisite)) \wedge \\ \neg rdf\_triple(User, knows, Link) \\ \Rightarrow is\_possible(display(Link), Situation)\end{aligned}$$

This means that it is possible to get access to a document (i.e. to display a link to this document) iff in the current situation, the user does not already have access to the document, he already knows all the prerequisite of the document, but he does not know the document itself.

As we saw in sect. 3.1, rules are triggered in the specified order. The rule system used in AHA![8] works quite differently. In their system, conditions trigger actions, and actions can make conditions come true. Therefore, cases of infinite loops are possible. Their solution to this problem is to restrict the condition/action system. Some rule compilers are even available to check whether or not a set of rules can lead to infinite loops. However, we consider that this step makes it more difficult for somebody with a small amount of knowledge about logics or the system to reuse it. Our solution might certainly not be final. Nevertheless, it is quite simple to understand, and yet able to express all the basic rules we have chosen to provide link adaptation, and which are quite usual. We still need to test this rule system for other kinds of adaptation. We will also work on choosing an appropriate subset of FOL: a subset that would allow us to express all our needs while being much less expressive than FOL, which is too wide for the current problem.

4 Reusability of the data

Even if one must, of course, provide data about his domain, GEAHS is provided with an ontology (coded in RDF) to describe the documents and their organisation. We are currently working on ways to transform ontologies to make them compliant with ours. We also provide a simple ontology for describing the learner model and are currently working on a more general ontology.

One of the main points in our system is that metadata are in RDF, which is simple, very expressive, and web-oriented. Thus, it is easier to im-

port/export data from/to another formalism, when a correct translator exists. It is also possible to reuse existing metadata written in RDF. Since conceptual graphs are often used by teachers and professors to provide a map of their course.

All the metadata we use are stored separately from the documents. This allows us to reuse external documents. Moreover, this means that one can use much more different document types than in many systems, where the adaptation information is stored within the document's body. With separate semantic levels stored in different structures, we also facilitate the possible exchanges.

5 Conclusion

As we saw, our main purpose has been to create an easy-to-reuse educational hypermedia system. By easy, we mean that somebody with low computer skills will still be able to reuse it and even change rules if needed. Our system is also based on standard metadata languages, allowing portability and reusability of preexisting resources. We are aware that some improvements have to be done in order to achieve as many functionalities as other existing systems. However we firmly believe that finding simpler but still very powerful solutions is a prerequisite to create an AH system that can be widely reused.

Our infering engine is fully reusable and its functionning is easy to understand. Provided rules are also reusable and can be changed. Schemas designed to ease the creation of the system as well as the reuse of pre-existing metadata and/or documents (in this case, with no transformation required) are provided in the system.

We are currently working on improving the learner schema, making it compliant with well-known standard as PAPI[9] or LIM[10]. We are also working on authoring tools to automate the transformation of the ontologies. In a long-term view, we wish to provide a very simple authoring tool for building AH systems based on a GEAHS architecture.

References

1. Peter Bruzilovsky. Methods and techniques of adaptive hypermedia. *Adaptive Hypertext and Hypermedia pp.: 1-43*, 1995.
2. Paul De Bra, Geert-Jan Houben, and Hongjing Wu. Aham: A dexter-based reference model for adaptive hypermedia. *UK Conference on Hypertext*, 1999.
3. Paul De Bra, Ad Aerts, Bart Berden, Barend de Lange, Brendan Rousseau, Tomi Santic, David Smits, and Natalia Stash. Aha! the adaptive hypermedia architecture. *Conference on Hypertext*, 2003.
4. Sheila McIlraith and Tran Cao Son. Adapting golog for programming the

semantic web. *Fifth International Symposium on Logical Formalizations of Commonsense Reasoning, pages 195–202*, 2001.
5. Christian S and Burkhard Freitag. Lmml – the learning material markup language framework. *Int. Worshop ICL*, 2002.
6. Nicola Henze and Wolfgang Nejdl. Logically characterizing adaptive educational hypermedia systems. *AH2003: Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2003.
7. Hector J. Levesque et al. Golog: A logic programming language for dynamic domains. *Logic-based artificial intelligence pp 257 - 279*, 2000.
8. Hongjing Wu. A reference architecture for adaptive hypermedia applications. *http://wwwis.win.tue.nl/ah/thesis/wu.pdf*, 2000.
9. S. Browne, J. Dongarra, G. Garner, N.and Ho, and P. Mucci. A portable programming interface for performance evaluation on modern processors. *he International Journal of High Performance Computing Applications*, 2000.
10. Ims learner information packaging information model specification. *http://www.imsglobal.org/profiles/lipinfo01.html*, 2001.
11. Jose Palazzo Moreira de Oliveira, Lydia Silva Munoz, Veronice de Freitas, Viviane P. Marcal, Isabela Gasparini, and Marilia Abrahao Amaral. Adaptweb: an adaptive web-based courseware. *3rd Annual Ariadne Conference*, 2003.
12. Richard Scherl, Michael Bieber, and Fabio Vitali. A situation calculus model of hypertext. *Proc. 31st HICSS Conf.*, 1998.
13. Hongjing Wu, Erik de Kort, and Paul De Bra. Design issues for general-purpose adaptive hypermedia systems. *Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, 2001.
14. Nicola Henze and Wolfgang Nejdl. Extendible adaptive hypermedia courseware: Integrating different courses and web material. *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2000)*, 2000.
15. Zoltan Miklos, Gustaf Neumann, Michael Sintek, and Uwe Zdun. Querying semantic web resources using triple views. *CInternational Symposium on Wearable Computers*, 2003.
16. Raymond Reiter. Proving properties of state in the situation calculus. *Artificial Intelligence, 64(2):337-351*, 1993.
17. Amel Bouzeghoub et al. A model of reusable educational components for the generation of adaptive courses. *SW-WL'03*.