Meta-levels of Adaptation in Education

Maurice Hendrix and Alexandra Cristea
The University of Warwick,
Department of Computer Science
Gibbet Hill Road, CV4 7AL, Coventry
United Kingdom
{maurice, acristea}@dcs.warwick.ac.uk

ABSTRACT
Large amounts of research currently exist into the design and implementation of adaptive educational systems. Research has shown that due to the high production cost of such adaptive courses, *semantics-based reuse* is one of the main requirements. E.g., the same adaptation strategy could be used for different courses, or the same course could be used with different strategies. Whilst using the LAG Adaptation Language to express reusable strategies, we noticed that strategies have common patterns that, in turn, could also be reused. Thus, higher level of reuse is possible than initially envisioned. Templates based on these patterns could significantly reduce the designers' work. This led us to research the best definition of a *meta-language for the LAG Adaptation Language*, facilitating the extraction of common design patterns. This paper describes one of the possible *template LAG languages* and reflects on its uses.

KEY WORDS
LAG; Grammar; CAF; LAOS; AHA!; Educational Adaptive Hypermedia; Authoring; Adaptation Engine

1. Introduction

Adaptive Hypermedia can generate valuable personalized (learning) experiences [4]. The driving force behind Adaptive Hypermedia is *adaptation engines*. One of the popular examples of such an adaptation engine is AHA! (Adaptive Hypermedia for All) [3]. An adaptation engine allows the adaptation of the hypermedia content according to a certain strategy. This can be, for example, a pedagogical strategy. In order for adaptation engines to deliver personalized educational material, this material has to be authored [6]. This comprises both the creation of the content, as well as the specification of the adaptation strategy. In this sense, the LAOS [7] framework offers a five layer model for Authoring of Adaptive Hypermedia. This framework is the basis of the MOT (My Online Teacher [6]) authoring system [6], which, in turn, is used as a generic authoring tool for Adaptive Hypermedia Systems such as AHA! [3], WHURLE [11], and even the commercial (non-adaptive) learning management system, Blackboard [2]. MOT, and its extension, LAG [7], use strategy files to describe the adaptation of hypermedia content towards users needs. These strategy files allow description of (pedagogical) strategies, which then determine how the content is personalized. The strategy files compile the Adaptation Layer in the LAOS framework and are written in a language called LAG [5]. LAG is one of the most generic adaptation languages currently in use, and a great variety of pedagogical strategies can be written in it (preference-based strategies, cognitive or learning style-based strategies, etc.).

However, adaptation engines have inherent run-time limitations which also limit the expressivity of the authoring systems. In particular, adaptation strategies that can be created are limited, thus adding restrictions to the pedagogical flexibility of the educational product. The exact nature of these restrictions will be discussed later in this paper. This issue has seemed so far a *bottleneck: surely authoring systems cannot express more than the delivery systems (adaptation engines) can deliver?* However, our current research is aiming to solve exactly this difficult problem.

A possible solution to these issues is to allow for prior processing of adaptation strategies, thus before the strategy is sent to the delivery engine. In semantic web terms, this means creating *meta-strategies*, which contain the template of later 'full-blown' strategies. This would allow for quite a different level of reuse, as well as try to solve some of the difficult issues of allowing the best type of adaptation on the authoring side. Meta-strategies have been proposed before [14], but the processing has been usually left on the adaptation engine side, and not the authoring side.

The proposed solution is more attractive because it allows *working with existent adaptation engines*. Thus, no changes need to be done in existent adaptation engines for the new solution to work.

The meta-level template strategies would use some knowledge about the structure and meta-data of the content, which is not currently possible with the AHA! system, or in fact any adaptation engine, because it requires processing the whole content (or large parts of it) at each step. Such a content description with structural information and meta-data is, for instance, CAF (Common Adaptation Format) [8], a format commonly

used by some Adaptive Hypermedia systems to transport domain contents. An author could examine the content and structure of the lesson by hand and compile a strategy file which uses fixed knowledge of the exact content - such as specific concept names or label names for a particular lesson. However, such processing is time-consuming and not realistic for large lessons. Such current limitations can be overcome by introducing a meta-level. At this level a template strategy file would be processed together with the content (e.g., with the CAF content). Before the lesson is created (for instance, in a delivery system such as in AHA!) a pre-processor could generate the lesson-specific strategy in a just-in-time fashion.

The remainder of this paper is organized as follows. First in section 2 we present an illustrative scenario, section 3 describes the current limitations of adaptation engines (adaptation delivery systems), and summarizes the key problems. Next, in section 4, the LAOS framework upon which our authoring is based is shortly sketched. A more detailed introduction of LAOS can be found in [7]. In section 5 we introduce CAF, the content interchange format. In section 6 we shortly introduce the LAG adaptation language and we present our *meta-level adaptation language* addition in section 7. A brief system setup overview is given in section 8 and finally, in section 9, we draw our conclusions from the current research.

2. Scenario

We illustrate the use of the proposed meta-levels of adaptation by means of a brief scenario in which a teacher needs to express a strategy that shows to students relevant information depending on their field of interest.

Dr. Johnsson[1] prepares a new on-line course on Biochemistry for first year undergraduate students in both biology and chemistry degrees. He wants to define adaptive behaviour for an online course in such a way that students who are more interested in biology are presented with more biological information and students who seem to be drawn more towards the chemistry aspect of the course get more about the chemistry part of the material. He could do this by creating two different static courses. However this would not give the students the chance to switch between options during the course. Moreover, this would force Dr. Johnsson to duplicate information which is relevant for both the biology and chemistry part. Finally, the 'two courses' option does not allow for any later extensions, such as a different strategy of presentation. For example, Dr. Johnsson may want to add a method of discovering a student's interests automatically, instead of using the initial student-driven method as presented here.

He decides to keep it simple and provides the students with a method of deciding their interest by themselves, effectively turning his course in an *adaptable*[2] course. He uses a given, predefined strategy. This strategy would, for example, contain the following code snippet (in pseudo-code):

```
if ((the user's interest contains
'bio') and (there are concepts with
labels containing 'bio'))
then show these concepts
```

In an extended version of the LAG language [5], the above pseudo-code would be translated as follows:

```
if (UM.GM.interest = *bio*  &&
GM.concept.label = *bio*) then
(PM.GM.concept.show = true)
```

This rule determines that concepts labelled, e.g., 'bio', 'biochem', 'biochemistry', 'biology' would be shown to students whose declared interest is any of 'bio', 'biochem', 'biochemistry', 'biology'. If Dr. Johnsson has some programming knowledge, he might want to improve upon the code he has received. As it stands, in order to show chemistry related concepts to students with interest in chemistry, another rule would have to be added, in which he replaces 'bio' with 'chem':

```
if ((the user's interest contains
'chem') and (there are concepts with
labels containing 'chem'))
then show these concepts
```

Now, students with interests defined as any of 'chem', 'biochem', 'chemistry', 'biochemistry' will see concepts with labels that contain the string 'chem'.

Dr. Johnsson looks at the extant rule again and he realizes that this code could be more generic. For instance, if he limits the range of interests the students can have to stems of words, say, *bio* and *chem*, he could simply write the following rule:

```
if(the user's interest is contained in
the concept label)
then show that concept to the user
```

In an extended version of the LAG language [5], the above pseudo-code would be translated as follows:

```
if(UM.GM.interest IN GM.concept.label)
then (PM.GM.concept.show = true)
```

The rule means then that if the 'bio' string is found in the label of the concepts, it will be displayed to biology-interested students, and similarly for the 'chem' string. Note that using inclusion or regular expressions and these rules, Dr. Johnsson can define concepts which are appropriate both for the students with the main interest in biology as well as for those with the main interest in chemistry.

[1] Any resemblance with an existing person is purely accidental.

[2] **Adaptable** means student-driven, as opposed to *adaptive* which is system-driven.

Moreover, the above rule is even more general, as it can be also applied to users with interests in computer science – e.g., choosing the interest definition 'comp' could render both 'computer science' and 'computing', or mathematics ('math') and so on. Basically, Dr. Johnsson has achieved a much higher level of generality and reusability of his strategy, for which his colleagues in other departments would be thankful.

3. Current limitations of Adaptive Hypermedia delivery systems

A strategy like the one proposed by us, would require some knowledge about the whole content, but would not necessarily increase the runtime complexity. Another example can be the simultaneous use of two adaptation strategies; for example a strategy where the system finds out whether the user prefers text or images based on an interest deduction, and then shows the learner the preferred type of content. These strategies could, in principle, run simultaneously, without increasing the runtime complexity. The first strategy could be achieved by establishing which labels exist in the initialisation. The second strategy could be achieved by either allowing arrays of labels, or otherwise allowing multiple data to be stored in the label.

The previous scenario reflects a level of reuse and compactness that has been so far unavailable in adaptive educational hypermedia systems. There are a number of reasons why this is so, mainly rooted in the current structure of adaptation engines.

State of the art Adaptive Hypermedia delivery systems (adaptation engines) have some bottom-up limitations. The main problem is that delivery systems currently determine the adaptation on a per-concept base. A broad knowledge of the whole content at every step is unavailable. This is mainly due to run-time complexity limitations. An example of what cannot currently be implemented is a strategy with an arbitrary set of labels denoting topics of interest, and showing the user only concepts of his own topic of interest (such as the ones described in the above scenario).

We also analyzed Interbook [9] WHURLE [11], AHA! [3]. All these systems suffer from these limitations. In AHA! [5], [6] reasoning is mainly done on a per-concept base (for persistent attributes). Volatile attributes can contain expressions, which reference other attributes, allowing for backward reasoning. This method only allows for access to variables concerning concepts that have already been visited before or are in the same line of hierarchy. AHA! also does not allow for any free program variables. AHA! can benefit from strategy combination (by using the external LAG language [5]) but does not offer any solution to conflicting naming.

AHA! can benefit thus from a flexible way of defining the adaptation, that allows for re-use of the same strategies, by using the LAG language for describing the adaptation. However, currently, LAG does not allow for the creation of variables. One can use user model variables, but not arrays. Furthermore it is not possible to access at once all concepts (or labels, weights) in a course. This limits the type of strategy that can be expressed in LAG. An example of what one might want to do is using an arbitrary number of different labels, for example to indicate topics of interest (as shown in the scenario). The problem in the LAG language is that there is no way to keep track of how many nodes are labelled with a given label (e.g., 'bio'). With a fixed set of labels, this is not a problem, as one could add rules for each label separately. However, with an arbitrary set of labels this would not be achievable. Keeping track of which labels exist globally is not possible due to the lack of programming variables; morever, accessing each item of the lesson at every step would also not be a scalable solution. The current version of the LAG language does not allow this, because it would significantly increase the run-time complexity (the runtime complexity would become exponential with the size of the content).

InterBook[3] [9] uses a knowledge-based approach to create adaptive, interactive electronic textbooks. Adaptation is more limited than in AHA!: it uses a classification of domain concepts into a *spectrum* and allows for adaptation towards the user's current knowledge state. The prerequisites are computed on a per-concept base, and neither free variables nor combined strategies are at all possible.

In WHURLE[4] [11], the *lesson plan* specifies a path through the content *chunks*. Rules are defined on a per-concept base (L1), and no free program variables are allowed (L2) [10]. Multiple strategies are possible by using XML pipelines [16]. The issue of different strategies using conflicting naming (L3), however, remains.

There are however systems that can deal with some of these issues, like Personal Reader [1]. Personal Reader uses a complex reasoning ontology and might therefore be able to provide some of the before mentioned adaptation. This however comes at the cost of a bigger computational complexity and therefore is a less scalable system.

Moreover, it still does not offer free program variables. Combining rules in an RDF [13] ontology is less problematic, as multiple relationships can be defined at the same time. There are however limitations as to what can be implemented efficiently. For example, if we look at the OWL ontology language [12] (based on RDF), we see that although OWL Full is complete and has no limitations as to what can be expressed, only the very limited set of OWL Lite can be implemented efficiently.

Summarizing, the following are *issues and limitations identified as influencing the authoring flexibility of adaptive hypermedia (AH) systems*:

L1. Adaptation engines *don't allow substring processing*. Consequently, rules based on

[3] http://www.contrib.andrew.cmu.edu/~plb/InterBook.html
[4] http://whurle.sourceforge.net/

checking the inclusion of substring 'bio' or 'chem' in other labels cannot be written. This is mainly due to the problem defined in L2, below.

L2. Adaptation engines don't (usually) allow for *non-instantiated program variables* [3]. Thus, authoring strategies which involve an unknown number of types, categories, etc., are currently not permitted. All domain-related variables need to be instantiated in the authoring stage.

L3. Most adaptive hypermedia delivery systems determine the *adaptation on a per-concept base* [3]. A broad knowledge of the whole content at every adaptation step is (usually) unavailable, mainly due to run-time complexity limitations. Thus, adaptation strategies cannot specify complex inter-concept rules; e.g., a strategy with an arbitrary set of labels denoting topics of interest, displaying to the user concepts related to his topic, without limiting the possible topics at design-time.

L4. There are extreme difficulties arising when *combining multiple strategies* [3]. Adaptation engines usually update sets of variables based on some triggering rules, without knowing which high-level adaptation strategies these variables represent. An example of a combined strategy currently difficult to implement is one where the system checks whether the user prefers text or images, and then displays the preferred type of content, filtered via a beginner-intermediate-advanced strategy, where concepts are shown based on the user's knowledge.

4. The LAOS framework

In order for our work to be applicable to most educational adaptation systems, we base it on a very generic framework. The LAOS model is a generic, layered framework for Adaptive Hypermedia authoring, built upon AHAM [15], a well-known model for Adaptive Hypermedia authoring. The following layers are defined in LAOS [7]:

- Domain Model (DM): which defines the domains of content, the composing elements and the relations between these elements. Elements can be, for instance, course sections, such as introduction, main text, conclusion, etc.
- Goal & Constraints Model (GM): filters useful domain concepts (possibly from multiple domains) and groups them together, according to the current goal. A goal in the educational domain is a pedagogical goal, such as, e.g., the construction of a course for a small-size group of first-year students that should last one week. Interestingly, a GM can filter information from more than one DM, similarly to creating a presentation for students based on one or more course books. In an analogue fashion, a DM could generate multiple GMs: this would be equivalent to saying that a book can generate many presentations.
- User Model (UM): stores user specific variables, e.g., interest, knowledge level, learning style, preferences, etc.
- Adaptation Model (AM): defines how the content is adapted to user's (learner's) needs. This is usually done via a set of rules, e.g.:

```
IF (the learner has interest in biology)
THEN (show him more examples that are
based on biology)
```

However, the adaptation can allow for more complex mechanisms. In fact, the LAOS adaptation model is further refined by the LAG framework [5].

- Presentation Model (PM): determines the look and feel of the presentation, in an adaptive (thus changeable) way, defines navigation elements, as well as quality of service, and other, e.g., contextual parameters. The main issue is that presentation model variables are external to the user, thus cannot be included in the user model. Adaptation can also be performed based on the presentation model variables. For instance, the presentation can adapt to the bandwidth of the connection, and send, e.g., less video lessons and more text-based lessons when the bandwidth is low, even if the user preference is for a visual presentation. To compensate for the possible user frustration, the adaptation mechanism could insert more pictures in the text, but still be using less bandwidth than a video presentation.

The LAOS layers are presented in **Error! Reference source not found.**Figure 1.

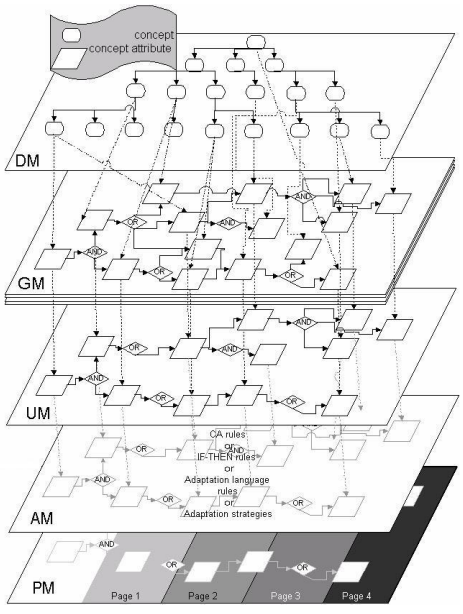

Figure 1 The five level AHS authoring model

5. The Common Adaptation Format (CAF)

To illustrate the course content representation, we use CAF [8]. CAF describes a Goal & Constraints Model (as in the LAOS framework) and all the Domain Models it uses. CAF represents these models using a relatively simple XML format. Below we give an example showing part of the biochemistry course of our illustrative scenario.

```
<CAF>
<domainmodel>
<concept>
<name>BioChemistry</name>
   <concept>
    <name>Biology </name>
    <attribute>
     <name>title</name>
     <contents>Biology</contents>
    </attribute>
   <concept>
    <name>Chemistry</name>
    <attribute>
     <name>title</name>
     <contents>Biology</contents>
    </attribute>
    <concept> …
</concept> …
</domainmodel>
<goalmodel>
<lesson>
<contents weight="0"
label="biochemistry">
BioChemistry\title</contents>
<contents weight="0" label="biology">
BioChemistry\Biology\title</contents>
<lesson>
<contents weight="0" label="chemistry">
BioChemistry\Chemistry\title</contents>
</lesson> ...
</lesson>
</goalmodel>
</CAF>
```

Figure 2 Example CAF file

6. The LAG adaptation language

The Adaptation Model thus defines how the content is adapted to users' needs. In the previous section we have shortly described a general framework for Adaptive Hypermedia authoring, LAOS. The functionality of the AM is further described by a three layer model, LAG [5], which defines: *direct adaptation techniques* (such as condition-action rules), *adaptation language* and finally, *adaptation strategies*.

A language that instantiates the adaptation language layer is the LAG language [5]. An example of a LAG strategy can be found below. As can be seen, the LAG language looks a little like a procedural programming language and, in fact, one can think of it as a domain specific programming language.

The example below shows a strategy which adapts to the particular field of interest of a student. Note that this example is the strategy which (the imaginary) Dr. Johnsson would use and is an adaptable strategy relying on the students to set their personal topic of interest.

The initialisation part is first performed once, after which, every time the user selects a (lesson) concept, the implementation part, which is the actual interaction with the user, is performed.

```
initialisation(
```

1) general: make every general (unlabeled) concept readable

```
 while(true)(PM.GM.Concept.show = true)
)
implementation(
```

2) concepts whose label contain 'setting' allow the student to signal that s/he has changed the topic of interest:

```
    if (GM.concept.label LIKE
    "*setting*") then(
        if (GM.concept.label LIKE
          "*bio*") then( UM.interest =
           "bio") else ( UM.interest =
           "chem")

     )
```

3) show the concepts appropriate for the user's topic of interest and hide the concepts which not appropriate.

```
    PM.GM.show = (UM.GM.interest IN
     GM.concept.label)

)
```

7. Meta-level addition to LAG

To solve some of the limitations mentioned in section 3, we want to add a pre-processing step to the authoring process. This step will take a template LAG file and the content in the form of a CAF file and pre-process it. The result would be a new LAG file which extends the strategy sketched by the template LAG file for the specific content described in the CAF file. We want to accommodate future changes to LAG, as well as our approach to be reusable and easily implementable and maintainable. Therefore we propose an extension based on the LAG language. Below we give the rules we intend to add to the LAG language for use in template LAG. We add the use of *regular expressions* to LAG with the combination of the '*LIKE*' keyword for use in wider scale comparisons. For example `GM.concept.label like "*bio*"` means the label of the concept is similar (but not identical) to 'bio'. A pre-processor can then retrieve all the possible values and translate this

pseudo-code of the meta-strategy to a concrete series of statements for all possible matching labels or values. This would mean, for large scale courses with multiple labels and meta-data, a great reduction in authoring time for a designer or teacher. Note that, in principle, this could also be implemented as part of the LAG parsers, but not without changing existing delivery systems. We also have the option to automatically create variables from labels, for example:

```
while (GM.concept.label LIKE *bio*)
then ( UM.GM._GM.concept.label_num++)
```

which would count all occurrences for each distinct label containing 'bio'. This would translate in rules like:

```
while (GM.concept.label==biology) then
( UM.GM.biology_num++)
```

if a label 'biology' was present in the course; if there was also a label 'bio', we would also obtain:

```
while (GM.concept.label==bio) then (
UM.GM.bio_num++)
```

Thus, for any label containing 'bio' we generate a separate rule – work which would be very time-consuming if it would have to be done manually.

7.1. Extended example

Below we discuss how our example would transform if we want to introduce a method of interest discovery instead of the student-driven approach. We also give an extended example of a possible implementation.

If our imaginary Dr. Johnsson wants to make his course adaptive, in other words if he wants to have the system automatically derive the students' topic of interest, he will have to use an extended version of the strategy shown before.

Let's assume that the number of concepts labelled with biology related labels and chemistry-related labels, respectively, is roughly the same (in the course, as well as in each chapter). In this case we can introduce a user model variable called *biochemcounter*. We could, e.g., let it range between 0 and 10, and initialize it with 5 (meaning the student has no known preference either for biology or for chemistry at the beginning of the course; 0 would mean strong biology preference, and 10 strong chemistry preference). Then we can use thresholds say 3 and 7 to determine if the students' interest is biology or chemistry respectively. This would be equivalent to say that if a student has consistently preferred for at least three steps one topic of interest to the other, than we could conclude upon the student's preference[5]. Then we would either add or subtract from the counter, each time a student visits a concept in chemistry, or biology, respectively (so, the step is of value 1). Thus we would have the following rule in the initialisation part of the strategy:

```
UM.biochemcounter = 5
```

and then the following rule in the implementation part of the strategy (the interactive part):

```
// student prefers biology
if (GM.concept.access=true &&
"bio" IN GM.concept.label )
then (UM.biochemcounter- -)

//student prefers chemistry
If(GM.concept.access = true &&
"bio" IN GM.concept.label )
then(UM.biochemcounter++)
```

Note that we use two 'if's rather then an if-then-else construct because a concept might contain both 'bio' and 'chem'. In this case, the two rules would cancel each other out: there is no determined preference for the student towards biology or chemistry in particular. Also note that the thresholds have to be determined by the teacher, and if the number of concepts for one topic (e.g., chemistry) differs greatly from that of the other topic (here, biology), more complex methods will be needed.

An example could be to define the adding/subtracting step as a numbered determining which portion of concepts are labelled with the specific topics of interest. This could lead to different step values for the different topics. For example, subtraction could be done by dividing the number of concepts labelled with a given topic, by the total number of concepts.

$$step_{topicA} = \frac{card(concepts_{topicA})}{card(concepts)}$$

Equation 1 Step width in case of nequal number of items between topics of interest

One might however argue that this computation needs to be done locally as well as globally (e.g., if a given concept is predominantly labelled with one label, it is natural to expect that the students will visit concepts with this label, but this might not say much about their real preference). Indeed, in such a case it would lead to more complex strategies beyond the current scope of our example.

8. System setup

In this section we propose a system setup for implementing the template LAG pre-processing. For the creation of regular LAG files based upon a template LAG file, both a template LAG file and knowledge about the content of the (CAF) lesson is needed. The template LAG files follow the extended LAG description introduced in section 2 and section 7. A pre-processor can replace these

[5] Please note that these thresholds are arbitrary and could be changed according to the exact pedagogical strategy. Depending on the number of concepts, a teacher might set the threshold at 20-30% of the concepts, etc.

constructs by regular LAG constructs based upon the content of the lesson. The resulting LAG file will then describe the same adaptation behaviour as the template LAG file, but only for this specific lesson. Implementing the pre-processor as a web-based application will enable it to transfer both the unchanged CAF file as well as the resulting LAG file to, e.g., the AHA! system. To facilitate the use of multiple strategies it will be possible to select multiple template LAG files. This process could, if the direct lesson creation is used, smoothly replace the current process without requiring any extra effort from the author. This process is shown in the figure below.

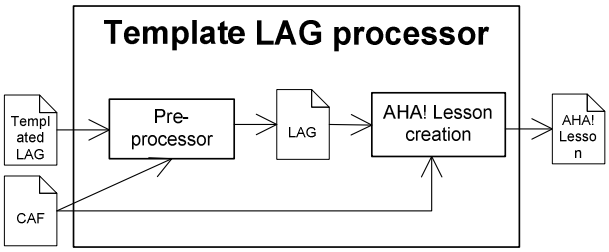

Figure 3 System setup of template LAG Pre-processor

9. Conclusions

Currently, available adaptive engines for Adaptive Hypermedia have limitations, mainly due to run-time issues. In this paper we have shown how some of these limitations can be overcome, by adding a pre-processor to the system setup. The pre-processor has access to knowledge about the learning content at compile-time, knowledge which is not available at run-time. Using this knowledge, more general LAG strategies can be written in a slightly extended LAG language called template LAG. The pre-processor can then produce a regular LAG strategy based on the template LAG strategy and the content. Although this work is illustrated by using a specific adaptation language, LAG, and existent systems, such as MOT and AHA!, it has a more generic value. The fact that we can simulate more complex adaptation strategies for learners, without a high effort from the author, can help to make adaptive hypermedia a tool for wider use than it is today.

References

[1] F. Abel, I. Brunkhorst, N. Henze, D. Krause, K. Mushtaq, P. Nasirifard and K. Tomaschewski, Personal Reader Agent: Personalized Access to Configurable Web Services. ABIS 2006 - 14th Workshop on Adaptivity and User Modeling in Interactive Systems, Hildesheim, October 9-11 2006.

[2] Blackboard website, http://www.blackboard.com/

[3] De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Stanic, T., Smits, D., Stash, N., AHA! The Adaptive Hypermedia Architecture. Proceedings of the ACM Hypertext Conference, Nottingham, Uk, August 2003, pp. 81-84.

[4] Brusilovsky, P., Adaptive hypermedia, User Modelling and User Adapted Interaction, Ten Year Anniversary Issue, 2001 (Alfred Kobsa, ed.) 11 (1/2), 87-110.

[5] Cristea, A.I., Calvi, L., The three Layers of Adaptation Granularity. UM'03, 2003, Pittsburg, US.

[6] Cristea, A.I., De Mooij, A., 2003, Adaptive Course Authoring: My Online Teacher. Proceedings of ICT'03, Papeete, French Polynesia.

[7] Cristea, A.I., De Mooij, A., 2003, LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators. WWW03 (The Twelfth International World Wide Web Conference), Alternate Track on Education,Budapest,Hungary, 2003.

[8] Cristea, A.I., Smits, D., and De Bra, P., 2005, Writing MOT, Reading AHA! - converting between an authoring and a delivery system for adaptive educational hypermedia -, A3EH Workshop, AIED'05, Amsterdam, The Netherlands

[9] Eklund, J. and Brusilovsky, P. (1999) InterBook: An Adaptive Tutoring System UniServe Science News Vol. 12. March 1999. p. 8-13.

[10] Kostelník, R., Bieliková, M., Web-Based Environment using Adapted Sequences of Programming Exercises. In Proc. of Information Systems Implementation and Modelling - ISIM 2003. M. Beneš (Ed.). MARQ Ostrava, Brno, April 28-30, pp.33-40.

[11] Moore, A.; Stewart, C.D.; Zakaria, M.R. & Brailsford, T.J.; (2003). WHURLE - an adaptive remote learning framework, International conference on Engineering Education (ICEE-2003), July 22-26, 2003, Valencia, Spain.

[12] OWL, Web Ontology Language, W3C, http://www.w3.org/TR/owl-ref/

[13] RDF, W3C, http://www.w3.org/RDF/

[14] Stash, N., Cristea, A., & De Bra, P., Learning Styles Adaptation Language for Adaptive Hypermedia, Adaptive Hypermedia (AH) 2006 conference, pp323-327, Dublin, Ireland.

[15] Wu., H., Houben, G.J., De Bra, P., AHAM: A Reference Model to Support Adaptive Hypermedia Authoring, Proc. of the "Zesde Interdisciplinaire Conferentie Informatiewetenschap", pp. 77-88, Antwerp, 1998.

[16] Zakaria, M.R., Moore, A., Stewart, C.D., Brailsford, T.J. (2003). "Pluggable" user models for adaptive hypermedia in education. Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia, August 26-30, 2003, Nottingham, UK. pp 170-171.