Defining Adaptation in a Generic Multi Layer Model: CAM: The GRAPPLE Conceptual Adaptation Model

Maurice Hendrix[1], Paul De Bra[2], Mykola Pechenizkiy[2], David Smits[2], and Alexandra Cristea[1]

[1] Department of Computer Science
The University of Warwick, Coventry
CV4 7AL, United Kingdom
{maurice,acristea}@dcs.warwick.ac.uk
[2] Faculty of Mathematic and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
debra@win.tue.nl, {m.pechenizkiy,d.smits}@tue.nl

Abstract. Authoring of Adaptive Hypermedia is a difficult and time consuming task. Reference models like LAOS and AHAM separate adaptation and content in different layers. Systems like AHA!, offer graphical tools based on these models to allow authors to define adaptation without knowing any adaptation language. The adaptation that can be defined using such tools is still limited. Authoring systems like MOT are more flexible, but usability of adaptation specification is low. This paper proposes a more generic model, CAM, which allows the adaptation to be defined in an arbitrary number of layers, where adaptation is expressed in terms of relationships between concepts. This model allows the creation of more powerful yet easier to use graphical authoring tools.

Keywords: Conceptual Adaptation Model; Adaptive TEL.

1 Introduction

Adaptive Hypermedia can potentially offer a rich learning experience with content adapted to the users' needs. However, this potential depends heavily on the ability of authors to create adaptive material. There exist several Adaptive Hypermedia reference models like AHAM [24] and LAOS [7] that are specifically developed for authoring. But even when using tools developed based upon these models, authoring remains a time consuming task [18]. A problem, even with graphical authoring tool like the Graph Author developed for AHA! [12] is that the adaptivity is specified in a single layer. Adaptation is based on *concept relationships* (of different *types* or *crt*s[1]) that have to be created one by one. The author will either have to use the *crts* defined by an expert or has to learn how to create new *crts* (for which there are no special design tools).

[1] Concept relationship types.

P. Dillenbourg and M. Specht (Eds.): EC-TEL 2008, LNCS 5192, pp. 132–143, 2008.

In this paper we present the authoring approach of the GRAPPLE project, an EU FP7 STREP project aimed at bringing adaptive technology-enhanced learning (or adaptive TEL for short) to the masses, by interfacing and/or integrating an *adaptive learning environment* (ALE) with different *learning management systems* (LMSs). The authoring approach in GRAPPLE is to offer a graphical tool to create a *conceptual adaptation model* (CAM). In Section 2 we explain the structure of a CAM with multiple adaptation layers. In Section 3 we show how an author can create concept relationships (leading to adaptation), either one by one or many at a time, and how the author can create *crts* in a similar graphical way. Although the multi-layer model is loosely based upon LAOS & LAG [11] authors are not required to write "pseudo code" as they do in LAG. We discuss the translation of a CAM to actual adaptation rules executed by an adaptation engine (while the user is using the learning application), and in Section 4 we discuss some issues regarding *termination* and *confluence* resulting from the CAM to adaptation rule translation.

2 The Conceptual Adaptation Model

In the GRAPPLE project, the structure of a *conceptual adaptation model* (CAM) is even more general and flexible than in previous frameworks [7], [24]: it contains an arbitrary number of layers, which may be different for each application. There will always be a domain model (DM) and user model (UM) layer and at least one layer with adaptation aspects, so the structure of CAMs in GRAPPLE is always a generalization of the AHAM model [24], and a refinement of the LAOS model [7]. Some example adaptation layers possible in a CAM include:

- *Prerequisite* layer: in this layer the author defines a structure of prerequisites between (sets of) concepts. Each prerequisite relationship connects two sets of concepts, the first of which contains prerequisite knowledge for the second set. This would correspond to part of the information stored in the Goal Model in LAOS, the ordering of information items.
- *Task (or Goal)* layer: in this layer the author connects sets of concepts with goals or tasks. All concepts of such a set need to be studied (and mastered) in order to reach the corresponding goal or complete the associated task. This would correspond with the overall goal of a particular goal model in LAOS, i.e., the metadata describing the whole instance (e.g., an introductory course for first year mathematics students in mathematical analysis).
- *Procedure* layer: in this layer the author may define a process model that must be followed during the learning process as it corresponds to the set of steps when actually performing a learning task. This would loosely correspond to the adaptation layer in AHAM and LAOS.

The relationships defined in the different CAM layers do not yet express the actual adaptation that will take place. A prerequisite may be translated to a rule that will change the presentation of links to concepts, but it may also be translated to the conditional inclusion of a prerequisite explanation (fragment). The translation of CAM structures to actual adaptation rules is described in Section 3 below.

3 Authoring CAMs

3.1 An Illustrative Scenario

We illustrate the authoring process of a CAM by means of a scenario:

Dr. Davies[2] prepares a new on-line course on the history of art for first year undergraduate students. He essentially has two options: he can either try to define a link structure between the course pages in such a way that students never see a link to information they cannot yet understand (because of missing foreknowledge) or he can define a CAM with prerequisite relationships and then rely on the ALE to ensure that students are only guided towards pages for which they have all the prerequisite knowledge. Although it is often argued that defining adaptation (a CAM in this case) means that creating an adaptive course is more work than creating a static course, the converse is actually true: the first option, to create a static course that is such that students can only follow links to information they are ready to understand is a nearly (or perhaps completely) impossible task and would require a lot of very careful work in selecting links to show to (all) students.

At first, Dr. Davies may think that it would be a good idea to create a prerequisite relationship from "Michelangelo" to "The Last Judgment", as the students should first learn something about the artist before learning about the artist's artworks. The authoring tool allows authors to draw a prerequisite relationship between a set of (prerequisite) concepts on the left and a set of concepts on the right. In this case the drawing would look like:

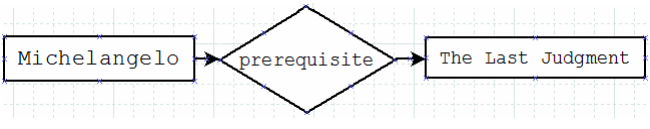

Fig. 1. Relation between Michelangelo and The Last Judgment

However, Dr. Davies then realizes that "Michelangelo" should not just be a prerequisite for "The Last Judgment" but for every artwork by Michelangelo. So he changes the drawing to and adds a constraint as follows:

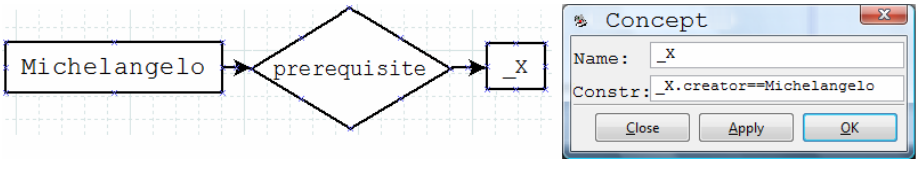

Fig. 2. Relation between Michelangelo and Placeholder Concept _X and constraints

The specific concept relationship thus becomes a partially generic one: there is still one specifically named concept but also a variable to express that the relationship

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

applies to all concepts _X that satisfy a certain condition. The underscore indicates that X is a variable and not a literal value.

Something perhaps not immediately obvious from this example is that there are two possible uses of this authoring tool (plus a combined third one):

- In the example, the "creator" attribute is a DM property, probably derived from a subject ontology. Which concepts have "Michelangelo" as prerequisite depends purely on the DM and this is thus independent of the learner taking the course.
- It is equally well possible to use an attribute from the UM in a relationship, thus creating relationships that are not only user-dependent but even dependent on the "current" instance of the user model.
- There is even a third possibility, by combining the previous two. The learning application can for instance *recommend* topics from a list that first of all depends on the DM but that also depends on the user's knowledge. For instance, only those recommended topics may be shown of which the user still has little or no knowledge.

Note that when the relationship only depends on DM information (like in the example) the replacement of _X by actual concepts could (but need not) be done at compile time, i.e. when translating the CAM into actual low level adaptation rules to be executed by the GRAPPLE ALE. When the relationship depends on UM information this is not possible.

Dr. Davies may later also go one step further in the definition of the prerequisite relationships. He may wish to state that for every artist and artwork the learner should learn about the artist before studying the artworks from that artist.
The drawing then becomes something like:

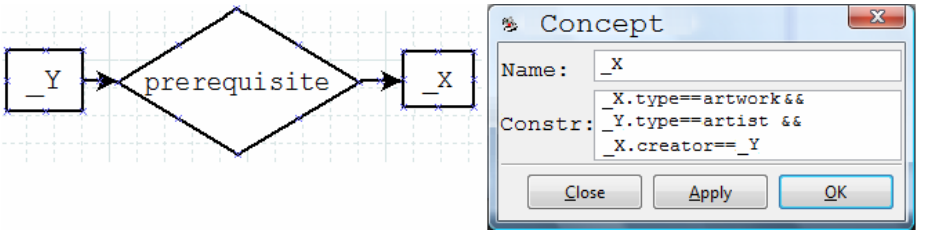

Fig. 3. Relationship for generalization of the Michelangelo example and constraints

Note that whereas creating a (set of) *specific* concept relationships does not require any knowledge of the structure of the DM or UM or any language to refer to DM or UM attributes of concepts, creating *generic* concept relationships, or *crts* does require some basic knowledge of the CAM language (to write `_X.creator==Michelangelo`). This language contains a still fairly high-level description of the semantics of the relationship. We consider it to be part of a *translation model* that defines how the relationships are translated to low level adaptation rules to be executed by the adaptation engine.

Each *crt* corresponds to a different layer in the CAM (and thus in the graphical presentation of the CAM editor). If Dr. Davies wishes to define a new type of relationship he can create a new layer and define a new *crt* as shown below:

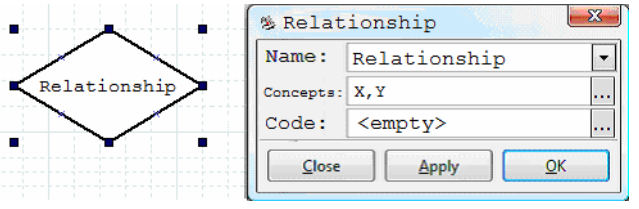

Fig. 4. Customizing a relationship

There are two ways to define the adaptation associated with the new relationship. Where it says "Code:" you can add the CAM language pseudo code for the adaptive behaviour. For instance, a statement:

```
_Y.suitability = ALL _X.knowledge > 70
```

could be used for indicating the desired behaviour for a prerequisite relationship. Although such code may look specific and implementation oriented, in reality it is not. The translation to the underlying adaptation engine may for instance define "suitability" to just be a *volatile* attribute of which the value is calculated when needed, or it may be a *persistent* attribute of which the value is updated each time the knowledge value of one of the prerequisites changes. Such implementation details are defined in a *translation model*. A single CAM may be translated to the actual adaptation language (and behaviour) of different adaptation engines, by using different translation models.

An alternative way to define the actual adaptive behaviour associated with a relationship is to just define a *method call* for a method that needs to be defined in the translation model. This approach makes the use of CAMs very powerful and generic but it also makes the behaviour dependent on a low level implementation rather than a high level specification. It is unlikely that teachers (like the imaginary Dr. Davies) will resort to writing program code for the adaptation engine.

3.2 Pedagogical Strategies in CAM

In the previous section we have seen a scenario illustrating how a teacher can create or customize an adaptive lesson. Previous research has defined interesting pedagogically sound adaptation strategies, representing different learning scenarios based on learners' needs, preferences, some also based on complex (and controversial) pedagogical foundations, such as learning styles, for Adaptive Hypermedia[3] [1]. In this section we will explore some of these strategies in relation to CAM. More specifically we will check how, in principle, such strategies can be expressed in the new CAM. As CAM is aimed to be richer than previous attempts, it should at least be able to express the basic strategies we have defined before. CAM is more flexible, however, and can

[3] See also our strategies page: http://prolearn.dcs.warwick.ac.uk/strategies.html

express strategies beyond what is analyzed here. While trying to express the (selection of) learning style related strategies we noticed some common issues:

- It is clear that we need to have some view of the Domain Model in order for the teacher to see what the available concepts are.
- A wizard like interface for ready-made strategies could be very helpful, while still allowing customization.
- The step-wise processing as previously implicitly assumed in LAOS/LAG based systems is still desirable. Otherwise some strategies like the Breadth- and Depth-First will not be possible, as inference rules will make sure the whole content will directly be visible. Thus, rules need to be triggered one-step-at-a-time, when certain events occur (e.g., a mouse-click). It is envisioned that, if desired, it should be possible to specify rules that trigger other rules, like in AHA!, however, in a visual way.
- In the LAOS/ LAG conversions to AHA!, one could control to a certain extent what kind of menus and other guidance the student would get. This represents adaptation of the presentation layer in LAOS, and reflects on interface changes and display for the student. It is desirable that in the new CAM-based systems this control will also be present to some extent.

Rollout

The rollout strategy is a very simple strategy that allows authors to decide when a certain concept or concept part should be shown: concepts to be shown after a certain number of steps could be classified as '*showafter*', and attached the meta-data containing the number of steps after which to be shown. Similarly, concepts classified as '*showatmost'* should only be displayed at most the given number of steps as again contained in meta-data. The roll-out strategy depends upon the tree hierarchy. We note hat it is straightforward to create such a hierarchy with the introduction of a parent-child relation.

First, authors need to be able to sort the concepts in the desired hierarchy (if this is not already available, e.g., if concepts are grouped in a graph). Next, we discuss the representation of the '*showafter'* part. The strategy demands that a concept is shown after its parent has been viewed a given number of times. As a constraint on _X, we have the following:

```
_X.metadata == 'showafter' && _X.parent ==_Y &&
UM._Y.showcount >= _X.showafter
```

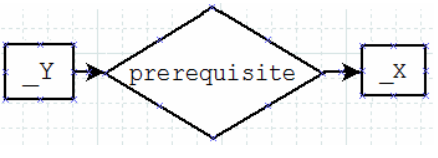

Fig. 5. 'Showafter' relationship

In Fig. 5, the relationship for 'showafter' is created via a prerequisite. This uses the prerequisite relation in its sense of condition on displaying concept _X based on

viewing concept _Y (and some supplementary conditions, as above). However, this does not use prerequisite in terms of knowledge update.

Depending on the implementation of prerequisite relationship, the '*showatmost'* part may or may not be needed. If the implementation of the prerequisite relationship makes sure that concepts for which previously the prerequisite was fulfilled, but for which this is no longer the case, are hidden, we do not need to do anything for the '*showatmost'* part. If this is not handled by the prerequisite we have to add a relationship that hides concepts once the have passed their '*showatmost'* threshold.

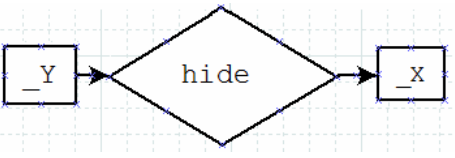

Fig. 6. 'Showatmost' via hide relation, only needed if prerequisite does not hide concepts

The constraint is then:

```
_X.metadata == 'showatmost' && _X.parent ==_Y &&
UM._Y.showcount > X.showatmost
```

Note that we also need to make sure that for each concept a count is kept in the user model. This can be done with a relationship '*countaccess'* relating a concept to itself.

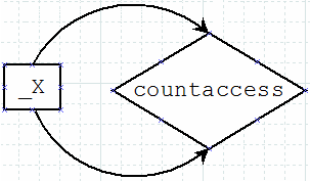

Fig. 7. 'Countaccess' relationship

The constraint will then be:

```
_X.access == true
```

The implementation of the countaccess relationship simply increases the count:

```
UM._X.showcount = UM._X.showcount+1
```

Depth First

The depth first strategy is used for sequential learners. One topic at a time is presented, and the student is allowed to go in-depth (hence, the name) in this topic first, before he proceeds with the next topic. Preferably, no menus' are shown to such students, and all they need to access is a 'next' button, taking them to their next study material, whether statically linked, or adaptively generated.

For the depth first strategy, again, the concepts have to be ordered in a hierarchy first. After this, a few relations are needed. Thus, we introduce a relation from each concept to each of its children, called *next child XOR next sibling,* see Fig. 7.

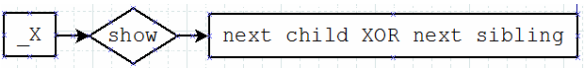

Fig. 8. The main relation implementing the 'Depth First', the logic in the constraint takes care of showing the appropriate next concept, either the next child or the next sibling

The condition must ensure that _X is the next sibling of _Y that needs to be shown, as well as update the User Model variable that keeps track of the current position of the learner within the hierarchical course. The condition shall only show the next sibling if the concept does not have any children left to be shown.

Finally we create a relationship from the root to the root, which shows first the concept unconditionally.

Breadth First

The breadth first strategy is used for global or holist learners. These learners like to see the global 'picture' first, before they dive into any topic. For such students, menus and other orientation devices are quite helpful.

Thus, implementation of this strategy has to start with the ordering of the concepts in a hierarchy. Next, we draw relations between each concept and each of its children, allowing them to show (all) the children if the parent has been shown. Finally we create a relationship from the root to the root, which shows the first concept unconditionally.

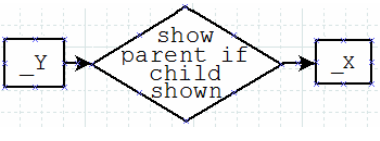

Fig. 9. The relation shows _Y if _X has been shown the condition is: _Y.parent==_X

4 Termination and Confluence in Multi-layer CAMs

The authoring process (for the concept structures and the adaptation) which is focused on the creation of concept relationships, appears to be fairly simple. Using different layers for different *crts* makes understanding the conceptual structure relatively easy too. However, this simplicity is partly an illusion. Depending on how the concept relationships are translated (using a translation model) to the low level adaptation rules for the adaptation engine, the (graph-like) structure of concept relationships of a single layer may already cause problems, and the combination of concept relationships from different layers may cause even more problems. We illustrate this with some examples.

Consider a simple structure where A is a prerequisite for B, B is a prerequisite for C and C is a prerequisite for A. This may cause a problem or not, depending on how prerequisites are used in the learning application.

- When "A is a prerequisite for B" results in links to B being recommended only after learning enough about A it is possible that the cycle of prerequisites causes the links to A, B and C to never become recommended to the learner. (Needless to say this is a problem.)
- When "A is a prerequisite for B" means that a short explanation of A will automatically be inserted into a page about B to compensate for the missing foreknowledge then there need not be a problem. If A is accessed first it will contain a prerequisite explanation of C, possibly preceded by a prerequisite explanation of B. (In this way the cycle does not cause a problem.)

Problems with undesirable structures like cycles are relatively easy to detect within a single layer. The problems become much more unpredictable when looking at the adaptation rules that result from translating the concept relationships from all layers together. The most common types of problems are *termination* and *confluence*.

4.1 Termination Problems

A simple example of where rule execution can run out of hand is when an author creates *knowledge propagation* relationships. A page that is essentially about Michelangelo may contain a brief description of some of his masterpieces, like "The Last Judgment". Our imaginary Dr. Davies may draw a "10% knowledge propagation" relationship from "Michelangelo" to "The Last Judgment". However, there may also be a generic rule that states that whenever you learn something about an artwork you also learn something (maybe also 10%) about the "creator" (artist) of that artwork. It is possible that the *knowledge propagation crt* has a translation model that will cause the translation of such a cycle to be an infinite loop of rule executions. (Each knowledge increase of "Michelangelo" may involve a knowledge increase of "The Last Judgment" and vice versa.) Disallowing cycles within a layer guarantees that there are no *termination problems* within that layer. However, even when each layer is without termination problems the interaction between rules of different layers may still cause an infinite loop.

The *static analysis* proposed in [24] results in conditions that may be too restrictive to apply them in multi-layer CAMs. The authoring tool might well disallow the creation of harmless concept relationships just because the static analysis detects a cycle, even when no infinite loop would be possible (when actually considering the conditions of the rules and the possible effect of the actions of the rules).

So rather than performing such static analysis, it is possible to apply a heuristic that is applied at runtime (in the adaptation engine) and that will ensure that there are no *termination problems*:

- The first step is to perform static analysis to ensure that no termination problem can be caused by the rules associated with the relationships of any single layer.
- The second step towards a solution for termination is to assign a (different) *priority* to each layer. (This is not to be confused with *execution phases* of

AHAM [24]. This is similar to priorities for adaptation strategies in the LAG language [5], [11].)

- The third step is to disallow updates to an attribute A of a concept C when C.A has been updated already by a rule associated with a higher priority layer or when an update to C.A already triggered the execution of a rule at a higher priority layer. (Note that just ensuring C.A has not been updated by a rule of a higher level is not enough. The C.A updates as a trigger is really a necessary additional condition.)

Although this method ensures that infinite loops are not possible, it makes the behaviour of the adaptation engine dependent on the choice of the priorities of the layers. We expect such problems to be rare, but nonetheless a system designer should determine the proper priorities for the "predefined" layers that are made available to authors (who do not define their own *crts* and translation models).

4.2 Confluence Problems

Confluence problems occur when more than one rule tries to update the same attribute of the same concept. The order in which such updates are performed may determine the resulting UM state.

- Static analysis can be used to ensure that there are no confluence problems within a single layer.
- In addition to this analysis we again assign a (different) *priority* to each layer and we disallow updates to attributes of concepts that were already updates at a higher (priority) level.

Like for termination, the assignment of priorities to layers may potentially influence the outcome (the UM instance) of the adaptation rule execution.

5 Related Work

Authoring of adaptive hypermedia is notoriously difficult work [2]. Research on improving this process ranges from ontology-based authoring [20], to integrating standards and their representations [16], [19] , using data mining techniques [23], web services [21], interfacing techniques between authoring systems [10], adaptation languages [11].

The current work is based on prior developments of adaptive hypermedia frameworks, like AHAM [24] and authoring frameworks for adaptive hypermedia, such as LAOS [7] and LAG [5]. Moreover, it is based on systems for adaptive hypermedia delivery, such as AHA! [12] and for authoring of adaptation, such as MOT, My Online Teacher [8], APels [13], ACCT [14].

Finally, this research is based on evaluations of authoring processes for adaptive hypermedia, as performed with various groups of students, in various locations, and with different versions of constantly improving tools [9], [4], [6], [10], [17], [15], [3]. Such research shows that, whilst having a higher flexibility and multiple layers for authoring is advantageous [3], [5] it is difficult for authors to actually program the adaptive behaviour of adaptation strategies [6], and it's thus much easier to have them

reuse strategies at a higher granularity level, in a graphical interface [3]. As the best paper of the 4th International Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia (A3H) shows [22], a template-based approach of a graphical nature is easier to handle by teachers, who in this way can better make use of the flexibility that the CAM GRAPPLE tool is offering.

6 Conclusions and Further Work

In this paper we proposed the structure of Conceptual Adaptation Models, as used in adaptive learning applications within the GRAPPLE project. We have shown that a graphical authoring tool helps authors in creating conceptual structures (of concept relationships) that guide the translation of CAMs to the adaptation rule language used by an adaptation engine. Using very similar graphical interface elements, an author can define a single *specific* concept relationship instance, a *generic* concept relationship or a new *concept relationship type* and its meaning, using a simple expression language.

The simple graphical approach to authoring does not alleviate the typical problems of *termination* and *confluence* in the generated adaptation rules. We briefly showed run-time heuristics that help avoid these problems in practice.

The graphical CAM authoring tool will be further developed in the coming months, and its usability evaluated with course authors. Within the GRAPPLE project work is proceeding in parallel, on the user modelling services and the adaptation engine. The progress of these components will determine the specification and implementation of *translation models* and a compiler from CAMs to low level adaptation rules.

Acknowledgment

This work has been performed in the framework of the IST project IST-2007-215434 GRAPPLE which is partly funded by the European Union. The authors would also like to acknowledge the contributions of their numerous colleagues from all 14 GRAPPLE project partners. This work is based on findings from the ALS project 229714-CP-1-2006-1-NL-MPP.

References

1. Brown, E., Cristea, A., Stewart, C., Brailsford, T.: Patterns in Authoring of Adaptive Educational Hypermedia: A Taxonomy of Learning Styles. International Peer-Reviewed Online Journal Education Technology and Society, Special Issue on Authoring of Adaptive Educational Hypermedia 8(3) (2005)
2. Brusilovsky, P.: Developing adaptive educational hypermedia systems: From design models to authoring tools. Authoring Tools for Advanced Technology Learning Environment. Dordrecht (2003)
3. Conlan, O., Wade, V.P.: Evaluation of APeLS - An Adaptive eLearning Service based on the Multi-model. In: De Bra, P.M.E., Nejdl, W. (eds.) AH 2004. LNCS, vol. 3137. Springer, Heidelberg (2004)
4. Cristea, A.I.: Evaluating Adaptive Hypermedia Authoring while Teaching Adaptive Systems. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357. Springer, Heidelberg (2004)

5. Cristea, A.I., Calvi, L.: The three Layers of Adaptation Granularity. In: Brusilovsky, P., Corbett, A.T., de Rosis, F. (eds.) UM 2003. LNCS, vol. 2702. Springer, Heidelberg (2003)
6. Cristea, A.I., Cristea, P.: Evaluation of Adaptive Hypermedia Authoring Patterns during a Socrates Programme Class. International Peer-Reviewed On-line & Print Journal Advanced Technology For Learning 1(2) (2004)
7. Cristea, A., de Mooij, A.: LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators. In: WWW 2003, The Twelfth International World Wide Web Conference, Alternate Track on Education, Budapest, Hungary (2003)
8. Cristea, A., de Mooij, A.: Adaptive Course Authoring: My Online Teacher. In: ICT 2003, International Conference on Telecommunications, Papeete, French Polynesia (2003)
9. Cristea, A.I., De Mooij, A.: Evaluation of MOT, an AHS Authoring Tool: URD Checklist and a special evaluation class. In: CATE 2003, Rhodos, Greece (2003)
10. Cristea, A.I., Stewart, C., Ashman, H., Cristea, P.: Evaluation of Adaptive Hypermedia Systems' Conversion. In: HT 2005, Salzburg, Austria (2005)
11. Cristea, A., Verschoor, M.: The LAG Grammar for Authoring the Adaptive Web. In: ITCC 2004, Las Vegas, US. IEEE, Los Alamitos (2004)
12. De Bra, P., Smits, D., Stash, N.: The Design of AHA! In: ACM Conference on Hypertext and Hypermedia, Odense, Denmark, p. 133 (2006)
13. De Bra, P., Brusilovsky, P., Conejo, R. (eds.): AH 2002. LNCS, vol. 2347. Springer, Heidelberg (2002)
14. Dagger, D., Wade, V.P., Colan, O., Developing Adaptive Pedagogy with the Adaptive Course Construction Toolkit, ACCT. In: AH 2004 (2004)
15. Dagger, D., Wade, V.P., Evaluation of Adaptive Course Construction Toolkit. In: ACCT, A3EH, Adaptive Authoring for Educational Hypermedia, Workshop AIED (2005)
16. Gutierrez, S.: Authoring of Adaptive Sequencing for IMS-LD. In: A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2007, Corfu, Greece (2007)
17. Hendrix, M., Cristea, A., Joy, M.: Evaluating the automatic and manual creation process of adaptive lessons. In: ICALT 2007, Niigata, Japan (2007)
18. Hendrix, M., Cristea, A., Nejdl, W.: Authoring Adaptive Educational Hypermedia on the Semantic Desktop. International Journal of Learning Technology, IJLT (2007)
19. Boticario, J.G., Santos, O.C.: A dynamic assistance approach to support the development and modelling of adaptive learning scenarios based on educational standards. In: A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2007, Corfu, Greece (2007)
20. Martin, B., Mitrovic, A., Suraweera, P.: Domain Modelling with Ontology: A Case Study. In: A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2007, Corfu, Greece (2007)
21. Meccawy, M., Stewart, C., Ashman, H.: Adaptive Educational Hypermedia Content Creation: A Web Service based Architecture. In: A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2006, Dublin, Ireland (2006)
22. Muñoz, F., Ortigosa, A.: An Adaptive Course on Template-based Adaptive Hypermedia Design. In: A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2006, Dublin, Ireland (2006)
23. Vialardi, C., Bravo, J., Ortigosa, A., Empowering, A.E.H.: Authors Using Data Mining Techniques. In: A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2007, Corfu, Greece (2007)
24. Wu, H.: A Reference Architecture for Adaptive Hypermedia Applications, doctoral thesis, Eindhoven University of Technology, The Netherlands (2004) ISBN 90-386-0572-2