Authoring of Learning Styles in Adaptive Hypermedia: Problems and Solutions

Natalia Stash
Faculty of Computer Science and Mathematics
Eindhoven University of Technology
Postbus 513, 5600 MB Eindhoven, The Netherlands
+31-40-247 3874
nstach@win.tue.nl

Alexandra Cristea
Faculty of Computer Science and Mathematics
Eindhoven University of Technology
Postbus 513, 5600 MB Eindhoven, The Netherlands
+31-40-247 4350
a.i.cristea@tue.nl

Paul De Bra
Faculty of Computer Science and Mathematics
Eindhoven University of Technology
Postbus 513, 5600 MB Eindhoven, The Netherlands
+31-40-247 2733
debra@win.tue.nl

ABSTRACT
Learning styles, as well as the best ways of responding with corresponding instructional strategies, have been intensively studied in the classical educational (classroom) setting. There is much less research of application of learning styles in the new educational space, created by the Web. Moreover, authoring applications are scarce, and they do not provide explicit choices and creation of instructional strategies for specific learning styles. The main objective of the research described in this paper is to provide the authors with a tool which will allow them to incorporate different learning styles in their adaptive educational hypermedia applications. In this way, we are creating a semantically significant interface between classical learning styles and instructional strategies and the modern field of adaptive educational hypermedia.

Categories and Subject Descriptors
H.1 [**Information Systems**] Models and Principles; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia - *architectures, navigation, theory, user issues*; E.1 [**Data**]: Data Structures - *distributed data structures, graphs and networks*; K.3.1 [**Computers and Education**]: Computer Uses in Education - *distance learning.*

General Terms
Design, Experimentation, Human Factors, Standardization, Theory.

Keywords
Learning styles, user modeling, adaptive hypermedia, authoring of adaptive hypermedia.

1. INTRODUCTION
Adaptive hypermedia tries to deal with the fact that the users are individuals. Most adaptive educational systems take into account learner features like goals/tasks, knowledge, background, hyperspace experience, preferences and interests [4].

However, less attention has been paid in adaptive hypermedia to the fact that people have different approaches to learning, namely that the individuals perceive and process information in very different ways. Recent researches [2][15][18][22] are trying to alleviate this and integrate learning styles in the design of their adaptive applications. Nevertheless, this is not an easy process. One of the difficulties in designing hypermedia software that incorporates learning styles is their actual representation in such an environment. The literature reveals that there have been very few studies, which have set out specifically to investigate the relationship between learning styles and hypermedia applications, especially adaptive versions.

From our point of view it is more interesting to let authors decide themselves which is the most appropriate learning style for their learner, and either select this particular learning style, or create it from scratch. Therefore, we don't advocate one particular learning style or another, but we are trying to create enough flexibility to make it possible for authors to design as many variations of learning styles as they like.

The remainder of this paper is structured as follows. In section 2 we describe background research and show what – to our knowledge - has been done so far in the direction of connecting adaptive hypermedia with learning styles. Section 3 is dedicated to study the different aspects of incorporating learning styles in AHA!, a mature adaptive hypermedia architecture [1][12]. Section 4 describes how the same and similar learning styles can be defined in MOT, an authoring tool for adaptive hypermedia [19]. Section 5 shows how the connection between MOT and AHA! can be made, via specific transformations. Finally, section 6 discusses the benefits and original points of our research and draws some conclusions.

2. POSSIBLE ADAPTATION TO LEARNING STYLES IN HYPERMEDIA ENVIRONMENTS
Currently several systems providing adaptation to users' learning styles have been created [11] [26] [25] [24] [20]. Most of the adaptive educational systems which incorporate learning styles are based on the notion that matching the learning strategies with the

learning styles improves learners' performance. Table 1 presents some of the existing systems and the learning styles they implement.

Table 1. Learning styles incorporated into adaptive systems

System	Learning style
ARTHUR [15]	visual-interactive, auditory-lecture and text styles
iWeaver [26]	auditory, visual, kinaesthetic, impulsive, reflective, global, analytical styles of Dunn and Dunn learning style model [13]
CS388 [6]	Felder-Silverman learning styles model [14] - global-sequential, visual-verbal, sensing-intuitive, inductive-deductive styles
AEC-ES [25]	field-dependent (FD) and field-independent (FI) style
LSAS [2]	global-sequential dimension of the Felder Silverman learning styles model
MANIC [24]	applies preferences for graphic versus textual information
INSPIRE [22]	Honey and Mumford [16] categorization of activists, pragmatists, reflectors and theorists based on Kolb [17]
Tangow [20]	sensing-intuitive dimension from the Felder-Silverman learning style model

Briefly, the kinds of adaptation provided by the systems are as follows.

In ARTHUR, iWeaver, CS388 and MANIC the adaptation is achieved by providing different media representations for each learner. ARTHUR and iWeaver are very similar in choice of learning style representation. *Auditory* representation is achieved using sounds and streaming audio. To appeal to *visual* and *kinesthetic* learners puzzles, animations, drag and drop examples and riddles are used. CS388 uses different types of media such as graphs, movies, text, slideshows. Similarly, MANIC uses graphic and textual information.

AEC-ES provides *field-dependent* learners with navigational support tools, such as concept map, graphic path indicator, advanced organizer, in order to help them organize the structure of the knowledge domain. The system guides them through the learning material via adaptive navigation support. *Field-independent* learners are provided with a learner control option - for them, the system shows a menu from which they can proceed with the course in any order. Learners can switch between different instructional strategies (designed for FD and FI learners).

In LSAS (Learning Styles Adaptive System) the *sequential* learners are provided with advanced organizers, maximum instruction and feedback, and more structured lessons. Symmetrically, *global* learners are guided via overviews and summaries of lessons. In the more recent Tangow and INSPIRE systems, adaptation lies in presenting a different sequence of alternative contents of the concepts. Concepts can be represented by 'example', 'activity', 'theory', 'exercise' in INSPIRE and by 'example', 'exposition' in Tangow. For example, for *Reflectors* in INSPIRE and *Sensing* users in Tangow the instructional strategy is *example-oriented*, meaning that the learners are presented with the example first and only afterwards with the other representations of the concept.

INSPIRE, as well as LSAS and AEC-ES, uses adaptive navigational support techniques with link annotation.

This review shows that different systems provide adaptation to learning styles in terms of content adaptation, navigation paths or usage of multiple navigational tools. However the choice of learning styles seems to be limited based on the suitable technology. Also, most of the presented systems, except iWeaver and MANIC, assess the learning styles through *psychometric questionnaires*. The disadvantage of this approach is that the learners are classified into stereotypical groups and the assumptions about their learning styles are not updated during the following interaction with the system. In the following we show how we try to avoid some of these limitations in AHA! and MOT.

Moreover, in this paper, we are looking, beside some classical approaches, also at some learning styles that are less treated in the literature, mainly because their representation and interpretation is considered more difficult, such as the learning styles proposed by Kolb [17] and extended by Honey and Mumford [16] as depicted in Figure 1. These styles will be discussed in more details when trying to implement them in AHA! and MOT respectively.

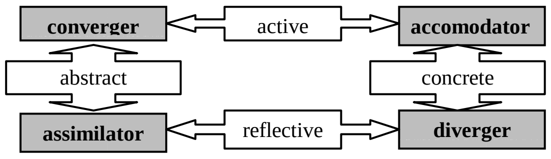


Figure 1. Kolb learning styles.

In the following, we describe various learning style implementation in AHA! and MOT.

3. HOW TO PERFORM ADAPTATION TO USERS LEARNING STYLES IN AHA!

3.1 Selected Learning Styles for AHA!

The review of existing systems shows that they provide adaptation to a selected learning or cognitive style. In many cases adaptation to learning styles assumes providing learners with different presentations of the learning material (example, theory, exercise, activity or image vs. text), by different type of media (audio, video). In all these cases the concept should be presented to the learner from various perspectives depending on his/her preferences and on the progress while working with the application. Therefore, in these cases, the main issue is presenting the aspects of a concept in different order. In AHA! this can be realized by using the similarity of relationships between a concept and its representations (which can be also defined as concepts). In the following subsection we show how this can be done by defining the '*illustrates*' relationship in the AHA! authoring tool 'Graph Author' [12]. This tool provides the authors with several typical *relationships* that occur in educational settings, like *prerequisites* or *knowledge propagation*. Moreover, by using the 'Graph Author', authors can define new concept relationships which they want to apply in their applications. These relationships will be automatically translated to the low-level adaptation rules used by the AHA! engine.

AHA! does not force the designer to create adaptive applications that will provide adaptation to a learning style selected by the authors of the system. With AHA! the authors have more flexibility in choosing a learning style and associating their own instructional strategies with it.

For their adaptive applications the authors may want to take into account various learning styles together; for example, the learning styles of the Honey and Mumford model, and the holist vs. serialist style. In that case different relationships cans be defined between the same concepts. The 'Graph Author' can combine these relationships into correct AHA! adaptation rules that express the meaning of all the given rules.

3.2 Associating an Instructional Strategy with the Selected Learning Style

Assume the author wants to create an adaptive application 'Learning the Java Programming Language'. He may want to make a distinction between *example-oriented* (Reflectors) and *activity-oriented* learners (Activists). According to Honey and Mumford's learning model [16], *Reflectors* are people who tend to collect and analyze data before taking an action. *Activists* are more motivated by experimentation and attracted by challenge.

The *example-oriented* learner may prefer to try a ready-made example first, then read the explanations. Only afterwards the learner can proceed to try building his/her own applet, similar to the one given in the example. Alternatively, the learner may be allowed to edit a given example and see how it works with the changes made.

The *activity-oriented* learner should be suggested first to try to create his own applet, compile and run it. Then he may have a look at a working example and compare it with the applet he/she created.

This kind of instructional strategy can be implemented in AHA!, by adding some special relationships, as follows. Some concepts of the application can be presented from different perspectives. For example, the concept 'WritingApplets' can be represented by an *example* applet, *explanation* of how the applet should be created or by the *task* of writing an applet. Figure 2 presents the 'Graph Author' interface that allows creating these alternatives; the domain concepts hierarchy is in the left frame, and specifying the behavior of the 'WritingApplets' concept is depicted in the right. The concept 'WritingApplets' can be represented by 3 concepts: 'AppletActivity', 'AppletExample' and 'AppletExplanation'.

The author may add, e.g., a new concept relationship type, *'illustrates'*. This type is a variant of the *prerequisite* and *propagation* relationships[1].

It is important to note that this newly created relationship type can be reused by other authors for their own adaptive applications.

The existing concept relationships use the values of the attributes of the domain concepts. The new concept relationship *'illustrates'* needs specific information about the learners, namely, information about the learning style.

This information about the learner is stored in the so-called concept '*personal*', which is created when the learner first logs into the system. The values of the attributes of this concept, like *name, login, password* are initialized through the registration form. The author may add arbitrary attributes to the concepts of the domain model as well as to concept 'personal'. In this way, the author can specify attributes which reflect the learner's style. In our example the author may use an attribute 'ActivistReflector', which can have values 'Activist', 'Reflector' or 'none' (if the learner can not be categorized as 'Activist' or 'Reflector').

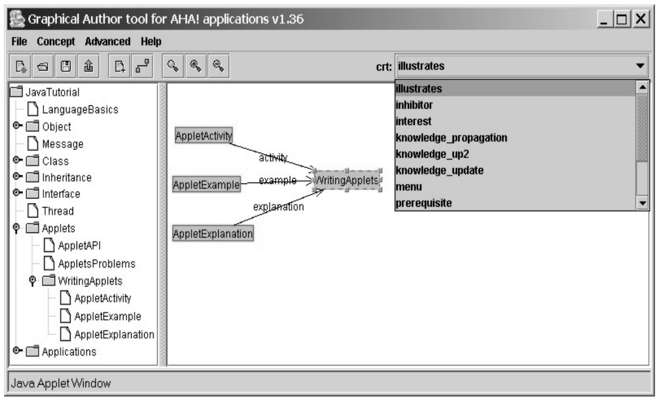


Figure 2. Specifying the behavior of the 'WritingApplets' concept.

The templates for concept relationships can use two variables (called 'source' and 'destination' or 'parent' and 'child'), as well as attributes of fixed, named concepts.

In our example, 'AppletActivity', 'AppletExample' and 'AppletExplanation' are source concepts. 'WritingApplets' is a destination concept. The 'illustrates' relationships can have a value associated with it. It reflects how the source concept represents the destination concept – 'by activity', 'by example', 'by explanation' and so on (for some other examples, by image/text or by audio/video). Destination concepts which can have different representations should have an attribute which reflects which of the 'representations' have been accessed. Therefore the author may specify that the 'WritingApplets' concept has 'activity' and 'example' attributes of Boolean type. When the learner accesses one of the source concepts and the concept is desirable, then the corresponding attribute of the destination concept is set to 'true'.

We can specify a general condition which describes under which circumstances each of the concepts becomes desirable. This condition is divided into six parts presented below, which are connected by 'OR' relationships (Table 2).

For the presentation this means that if the learner is 'Reflector' and he starts reading a page associated with a concept 'WritingApplets' he sees links to 'AppletActivity', 'AppletExample' and 'AppletExplanation' pages.

Links in AHA! can have 3 different colors indicating the desirability of the link. Links to 'AppletExample' and 'AppletExplanation' will be shown in blue (desirable) and to 'AppletActivity' in black color or in the color of the rest of the text (as it is not desirable). After the learner looks at the example, the link to 'AppletExample' becomes purple, meaning the concept is desirable but already read. Meanwhile, a link to 'AppletActivity' becomes blue as a prerequisite condition:

[1] indicating that knowledge about one concept is a prerequisite for another concept, respectively knowledge increase for some concept contributes to the knowledge of another concept.

personal.ActivistReflector==“Reflector” &&
WritingApplets.example &&
AppletActivity.representation==“activity”

becomes satisfied (the learner is ‘Reflector’, he read an example concept and the *AppletActivity.representation==“activity”*).

Table 2. Honey and Mumford learning styles for AHA!

Condition	Explanation
personal.ActivistReflector ==“none”	for the learner who is not categorized as ‘Reflector’ or ‘Activist’ all the concepts are presented as desirable
source.representation == “explanation”	the explanation concepts are desirable for different learners
personal.ActivistReflector== “Reflector”&& source.representation==“example”	if the learner is ‘Reflector’ then the ‘example’ concepts are desirable
personal.ActivistReflector== “Reflector” && destination.example && source.representation==“activity”	if the learner is ‘Reflector’ the ‘activity’ concept becomes desirable after he read the ‘example’ concept
personal.ActivistReflector== “Activist”&& source.representation==“activity”	if the learner is ‘Activist’ then the ‘activity’ concepts are desirable
personal.ActivistReflector== “Activist” && destination.activity && source.representation==“example”	if the learner is ‘Activist’ the ‘example’ concept becomes desirable after he read the ‘activity’ concept

AHA! allows to produce different versions of the pages by including different embedded objects. By ‘object’ we mean a piece of information which exists in pages or other objects. The XHTML pages use the ‘object’ tag to indicate where conditionally included objects should be placed. The author defines the behavior of an object in a concept, which he links to that object. This concept describes under which conditions, which base-object is included into the page. A base object is a well-formed document that can include other objects.

Assume that a page describing the activity has a link to an example. The author may want to insert a text block before the link. Under various conditions, different texts can be put before the link. In case the learner is an ‘Activist’ and he starts an experiment, the text block should be: ‘You may follow this link to see an example’. Or if the learner is a ‘Reflector’, and he already saw an example and starts with an activity, the text block should say: ‘You may return back to the previously visited example’. The author may define an object ‘TextBlock’ which he includes into the XHTML page, associated with the ‘AppletActivity’ concept:

<object name="JavaTutorial.TextBlock" type="aha/text"/ >

The ‘aha/text’ type tells the AHA! engine that ‘JavaTutorial.TextBlock’ is a conditionally included object (concept).

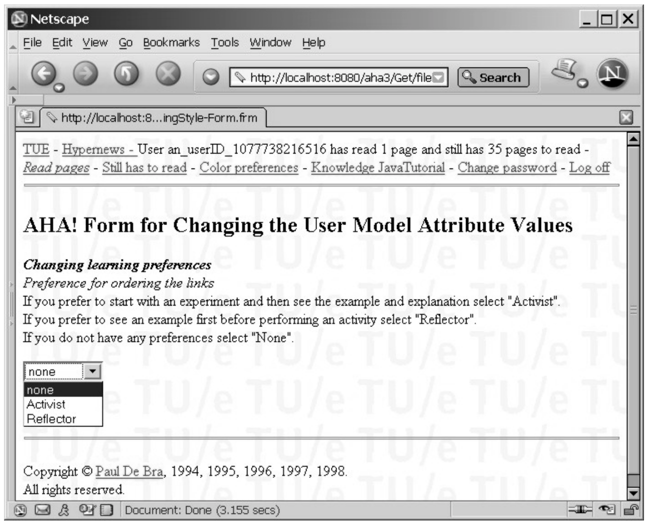
Various presentations of the concept can be defined in the same way as in the above example of including a text block.

Instead of defining these representations as concepts, an alternative solution is for the author to define an object concept ‘WritingAppletRepresentation’, and include this object into a page associated with the ‘WritingApplets’ concept. Then, for the ‘Activist’ the page will be presented starting with a description of an activity, followed by the links to an explanation and example, and vice versa for the ‘Reflector’. However, the links that will appear in this alternative presentation (pointing to an example and explanation) will not be shown as desirable or not (so no color scheme will be applied), as there are no concepts associated with them.

3.3 Assessing Learning Styles in AHA!

The majority of the existing systems assesses learners’ learning styles through psychometric questionnaires, which classify them into stereotypical groups. Afterwards, during the actual learning, the assumptions about the learner’s style are not updated.

AHA! currently does not provide any questionnaires for assessing learning styles. If the learner knows what his/her learning style is he/she can manually state it through the registration form (Figure 3). Learning preferences can be also specified based on the general description of instructional strategies designed for various learning styles. This description should be provided by the author of an application.


Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/aha3/Get/file
Search
http://localhost:8...ingStyle-Form.frm

TUE - Hypernews - User an_userID_1077738216516 has read 1 page and still has 35 pages to read -
Read pages - Still has to read - Color preferences - Knowledge JavaTutorial - Change password - Log off

AHA! Form for Changing the User Model Attribute Values

Changing learning preferences
Preference for ordering the links
If you prefer to start with an experiment and then see the example and explanation select "Activist".
If you prefer to see an example first before performing an activity select "Reflector".
If you do not have any preferences select "None".

none
none
Activist
Reflector

Copyright © Paul De Bra, 1994, 1995, 1996, 1997, 1998.
All rights reserved.

Document: Done (3.155 secs)

Figure 3. Form to change the user model attribute values.

AHA! can provide a mechanism for *inferring* the learner’s *preferences (patterns)* corresponding to particular learning styles. Based on the learner’s browsing behavior the system can make assumptions about preferences, for example, for reading order or different types of media. However we do not claim that by using AHA! we can assess the learning styles which are more general than just preferences.

In the transition template for a concept relationship, the author may specify the actions which are performed when the learner accesses a page associated with a concept (like the probability that the learner has a particular preference increases or decreases). If the learner specified his/her learning style/preference through the registration form and accesses the recommended concept, then the system’s confidence that learner defined his/her learning style correctly increases. Otherwise, if the learner accesses non-desirable concepts, this confidence decreases. In case it becomes lower than some threshold value (may be defined by the author) the system may ask the learner if he/she wants to change to an instructional strategy which corresponds to another learning style.

If the learner didn’t specify any preferences through the registration form, then the system may trace the order in which concepts' representations are accessed, thereby increasing or decreasing the confidence that the learner has a particular preference. In this case, when the system reaches some threshold value (also defined by the author) the system may inform the learner that his/her browsing behavior indicates a preference which corresponds to a particular learning style and he/she may switch to an instructional strategy which corresponds to that style.

If the learner is not satisfied with an instructional strategy he/she can always inspect the user model and make necessary corrections. AHA! provides a special tool that allows authors to create forms to let the learners change values of attributes of concepts in their user model. It is thus possible to create a form that lets learner to change their 'ActivistReflector' values (Figure 3).

4. DESIGNING LEARNING STYLES IN MOT

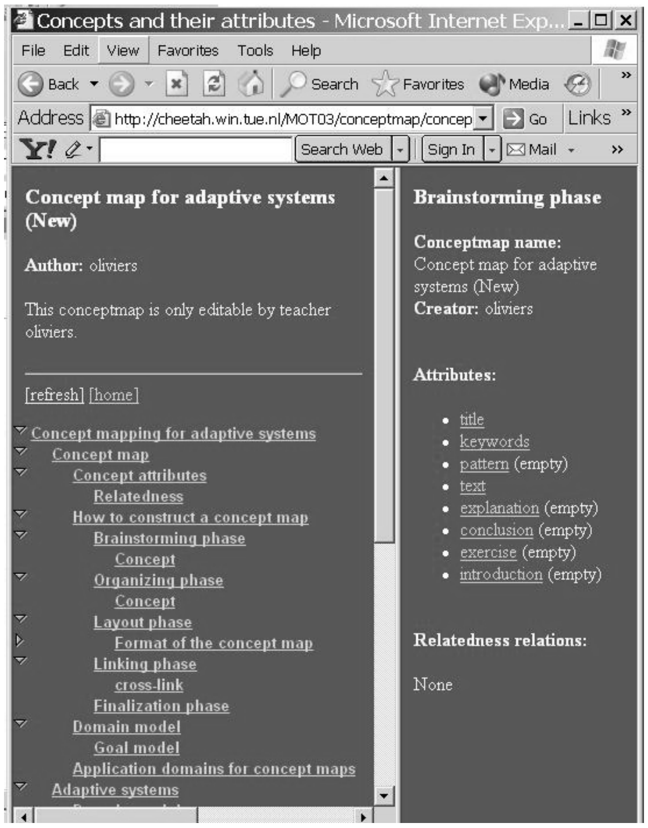
4.1 Selection of Learning Style Elements

In the previous sections we have seen how the actual instantiations of learning styles – translated into their respective teaching strategies – can be represented. AHA! allows a lot of freedom of expression, so basically anything is possible to represent. Moreover, the old, purely XML tagging authoring language has been replaced with frame tools, which are now-a-days advocated as being the most progressing form of adaptive hypermedia authoring [5].

However, the main problem with the strategies defined in AHA! is that they are instances, so they are bound to their conceptual representation. If the same strategy has to be applied again on a different domain or concept map, it has to be generated again from scratch, and no reuse is possible (with the exception of the new link types).

In [10] we have introduced the basis of an adaptation language, which tries to identify and represent the repetitive patterns that appear in adaptive hypermedia, not in terms of concept representation, but in terms of *(adaptive) concept use*. This language allows the usage of general concepts as well as concept instances. More importantly, for the purpose of the current paper, it allows to create adaptive strategies written in this adaptation language. This language is implemented as one of the newer components of MOT [19], an online environment designed for adaptive hypermedia authoring. In the following, we will analyze how the learning styles previously described and interpreted for AHA! can be expressed in MOT.

First let's look at the two major ingredients of the learning styles: providing *different learners with different presentations* of the learning material (such as explanations, theory, exercises, etc.), and providing *different learners with different ordering* of the material. Figure 4 shows how these different presentations are authored in MOT. The left frame represents the hierarchy of concepts created within the concept map entitled 'Concept map for adaptive systems'; the right frame shows the different possible presentations of a specific concept, called 'Brainstorming phase'. If we would be the creator (and not 'olivier') we would also see in the left frame a button called 'add attribute' which would allow us to add an unlimited number of other different attributes.


Concepts and their attributes - Microsoft Internet Exp...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://cheetah.win.tue.nl/MOT03/conceptmap/concep
Go Links
Search Web Sign In Mail

Concept map for adaptive systems (New)

Author: oliviers

This conceptmap is only editable by teacher oliviers.

[refresh] [home]

- Concept mapping for adaptive systems
  - Concept map
    - Concept attributes
      - Relatedness
    - How to construct a concept map
      - Brainstorming phase
        - Concept
      - Organizing phase
        - Concept
      - Layout phase
        - Format of the concept map
      - Linking phase
        - cross-link
      - Finalization phase
    - Domain model
      - Goal model
    - Application domains for concept maps
  - Adaptive systems

Brainstorming phase

Conceptmap name: Concept map for adaptive systems (New)
Creator: oliviers

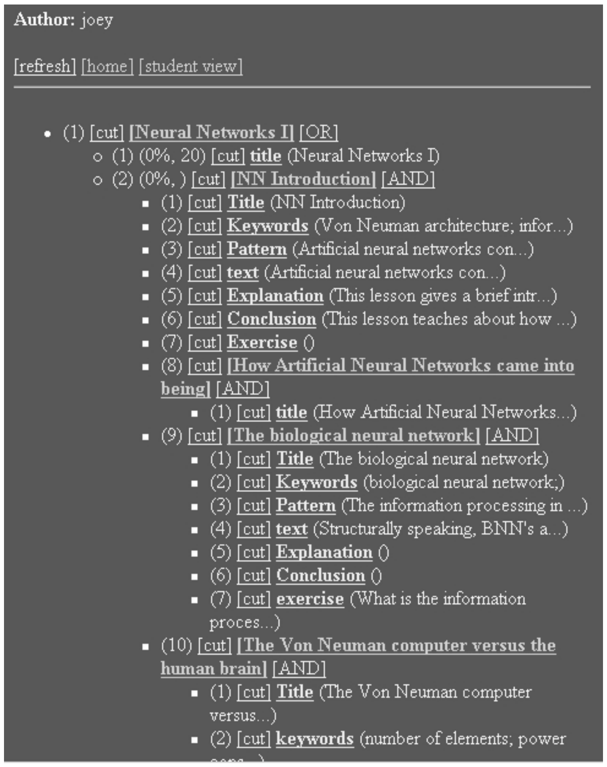
Attributes:

- title
- keywords
- pattern (empty)
- text
- explanation (empty)
- conclusion (empty)
- exercise (empty)
- introduction (empty)

Relatedness relations:

None

Figure 4. Alternative presentations of learning material.



Author: joey

[refresh] [home] [student view]

- (1) [cut] **[Neural Networks I]** [OR]
  - (1) (0%, 20) [cut] **title** (Neural Networks I)
  - (2) (0%,) [cut] **[NN Introduction]** [AND]
    - (1) [cut] **Title** (NN Introduction)
    - (2) [cut] **Keywords** (Von Neuman architecture; infor...)
    - (3) [cut] **Pattern** (Artificial neural networks con...)
    - (4) [cut] **text** (Artificial neural networks con...)
    - (5) [cut] **Explanation** (This lesson gives a brief intr...)
    - (6) [cut] **Conclusion** (This lesson teaches about how ...)
    - (7) [cut] **Exercise** ()
    - (8) [cut] **[How Artificial Neural Networks came into being]** [AND]
      - (1) [cut] **title** (How Artificial Neural Networks...)
    - (9) [cut] **[The biological neural network]** [AND]
      - (1) [cut] **Title** (The biological neural network)
      - (2) [cut] **Keywords** (biological neural network;)
      - (3) [cut] **Pattern** (The information processing in ...)
      - (4) [cut] **text** (Structurally speaking, BNN's a...)
      - (5) [cut] **Explanation** ()
      - (6) [cut] **Conclusion** ()
      - (7) [cut] **exercise** (What is the information proces...)
    - (10) [cut] **[The Von Neuman computer versus the human brain]** [AND]
      - (1) [cut] **Title** (The Von Neuman computer versus...)
      - (2) [cut] **keywords** (number of elements; power

Figure 5. Ordering of the learning material in MOT.

These attributes can be in concordance with a given learning standard (such as SCORM [23], LOM [21], etc.).

These attributes are the meta-data that can be used in various interpretations of the learning contents, as specified by different learning strategies, as we shall see.

Ordering does not happen in MOT at the level of the concept maps as in Figure 4. This is due to the fact that ordering has something to do with the goal of the presentation, with the audience we are aiming at. MOT therefore allows a different layer for the type of relations between concepts that are inherent to the presentation. This layer is called in MOT the 'lesson' layer. Figure 5 shows an instance of the lesson layer in MOT. The validity of the introduction of this extra layer has already been proven by testing with students [7].

As can be seen in Figure 5, the ingredients of the lesson layer are the same as the ones in the concept layer. Actually, the lesson layer is a restricted, constrained version of the concept layer. The type of restriction applied has something to do with the type of presentation desired – so can come as an answer to the requirement of a specific learning style. It is easy to see that restrictions can imply selecting only attributes of a specific type, such as only *explanations* or only *exercises*.

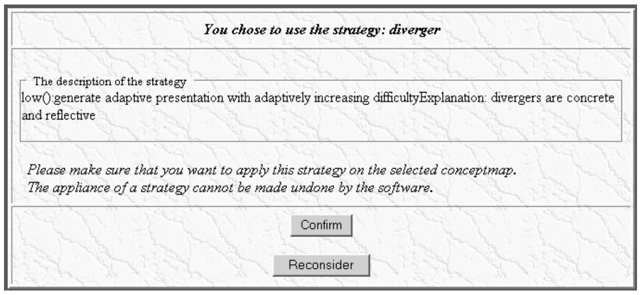
4.2 Associating Instructional Strategies with Selected Learning Styles in MOT

Here we will look at the combined effects of the *learning characteristics* analyzed for AHA!. In other words, we look at the combination of *reflector* and *concrete* tendencies, which together generate, as the cognitive science literature [17] tells us, the cognitive style *diverger*. Similarly, combining *abstract* and *active* tendencies generates the opposite, i.e., *converger*.

Here the major difference to the AHA! approach becomes clear: the definition of adaptive strategies corresponding to instructional strategies is enabled in MOT for generic concepts, and the same strategy can, in principle, be applied over different concept maps or lessons as described in the previous subsection. The MOT approach is inspired by the author-push, while the AHA! approach is inspired by the adaptation engine pull. In other words, MOT tries to realize what authors supposedly desire from an authoring tool, while AHA! tries to implement what is possible given the limitations of the adaptive hypermedia engine. Obviously, these two approaches are not totally independent of each other, and they both have to influence a final adaptive hypermedia authoring product.

For MOT, the author can, in principle, just select an adaptive strategy corresponding to an instructional strategy created by a different author, and apply it to an arbitrary concept map or lesson map. The author might not have created any of these two pieces, but still can use them in his/her class. This represents high-level adaptive hypermedia authoring. On a lower level, the author might have created a lesson, based on different concept maps, or even just one concept map – but still can select some strategy from a given list of existing ones. Only when having the urge to create his/her own adaptive strategies does an author in MOT need to specify the defining elements of this strategy. The result of the creation, however, can be reused by others.

In the following, we show how these instructional strategies can be written in MOT. We selected for exemplification two of the Kolb learning styles [16], *diverger* and *converger*. In MOT, instructional strategies corresponding to learning styles can be authored via a frame authoring tool [5]. First, the description of the strategy can be specified, as in Figure 6. The figure shows the description of the strategy for diverger.

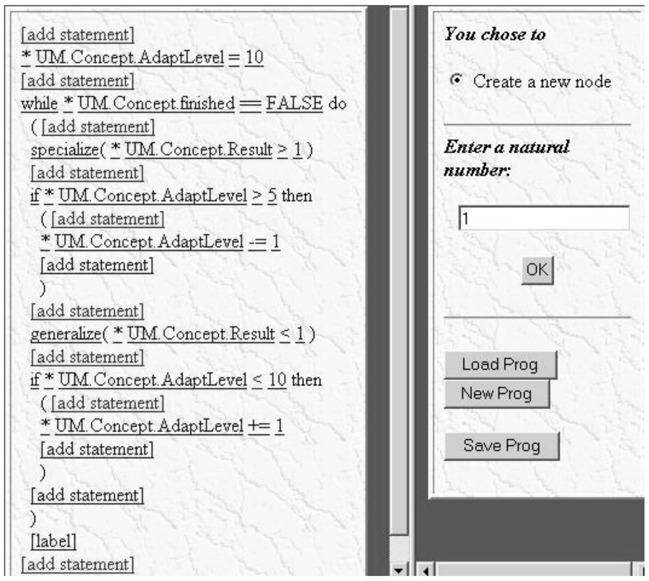

You chose to use the strategy: diverger

The description of the strategy

low():generate adaptive presentation with adaptively increasing difficultyExplanation: divergers are concrete and reflective

Please make sure that you want to apply this strategy on the selected conceptmap.
The appliance of a strategy cannot be made undone by the software.

Figure 6. Defining the description of the generic strategy for diverger in MOT.


```
[add statement]
* UM.Concept.AdaptLevel = 10
[add statement]
while * UM.Concept.finished == FALSE do
  ( [add statement]
  specialize( * UM.Concept.Result ≥ 1 )
  [add statement]
  if * UM.Concept.AdaptLevel ≥ 5 then
    ( [add statement]
    * UM.Concept.AdaptLevel -= 1
    [add statement]
    )
  [add statement]
  generalize( * UM.Concept.Result ≤ 1 )
  [add statement]
  if * UM.Concept.AdaptLevel ≤ 10 then
    ( [add statement]
    * UM.Concept.AdaptLevel += 1
    [add statement]
    )
  [add statement]
  )
  [label]
[add statement]
```

You chose to

◉ Create a new node

Enter a natural number:

1

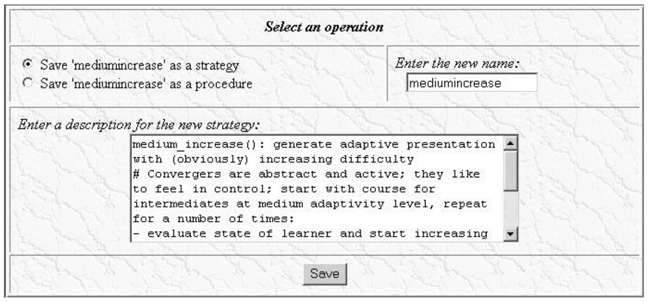
Figure 7. Writing the diverger generic strategy in MOT.

Figure 7 shows the creation interface for adaptive strategies corresponding to different learning styles. The interface allows a template (building block) type of programming, making in this way both the task of the author, and the task of the compiler easier. New blocks of adaptive language constructs can only be inserted in the places marked by 'add statement'.

This particular version of the expression of the adaptive behavior for the learning style *diverger* in Figure 7 has been first proposed in [10]. The written adaptive strategy just uses '*generalize*' to send the learner to more general (and easier) concepts, if the results (on some test, for instance) were poor, and, on the contrary, uses '*specialize*', if the results were good (see also Figure 14). Moreover, the adaptive strategy takes into consideration the tendency of the learner to diverge, so keeps him/her on track by keeping at all times a high level of adaptivity (i.e., the learner's choices are reduced, the system takes most of the decisions and there are none - or very few - user-tunable parameters in the user model). Adaptivity level (UM.Concept.AdaptLevel, Figure 7) can be slightly tuned, so that learners with good progress get more flexibility, and vice-versa. Please note that all the attribute values used in the example in Figure 7 are generic, i.e., they are not yet overlaid over an existing concept map (as in Figure 4). This means that they can be applied on any concept (or lesson) map that *has* the elements which are required by that specific instructional strategy.

In this way, in MOT, more complex behavior can be specified for the desired adaptive strategy, than just via a one or two attributes check such as in AHA!. It is not that it is impossible to represent more complex behavior in AHA! – it is however unrealistic to think that an author would be able to keep track of all the complex interactions of the created behavior. Unless things are kept simple, errors are hard to avoid.

An example from the other corner of the Kolb diagram (Figure 1) is the *converger* behavior. Figure 8 shows the description creation for this strategy and Figure 9 its implementation in MOT.


Select an operation

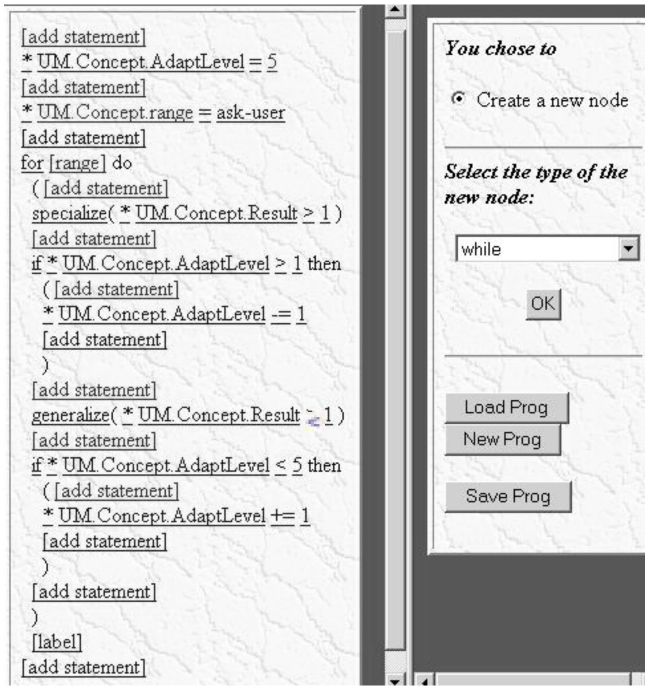
[✓ Save 'mediumincrease' as a strategy]
Save 'mediumincrease' as a procedure

Enter the new name: mediumincrease

Enter a description for the new strategy:

```
medium_increase(): generate adaptive presentation
with (obviously) increasing difficulty
# Convergers are abstract and active; they like
to feel in control; start with course for
intermediates at medium adaptivity level, repeat
for a number of times:
- evaluate state of learner and start increasing
```

Figure 8. Description of the generic strategy for converger.


```
[add statement]
* UM.Concept.AdaptLevel = 5
[add statement]
* UM.Concept.range = ask-user
[add statement]
for [range] do
  ( [add statement]
  specialize( * UM.Concept.Result ≥ 1 )
  [add statement]
  if * UM.Concept.AdaptLevel ≥ 1 then
    ( [add statement]
    * UM.Concept.AdaptLevel -= 1
    [add statement]
    )
  [add statement]
  generalize( * UM.Concept.Result < 1 )
  [add statement]
  if * UM.Concept.AdaptLevel ≤ 5 then
    ( [add statement]
    * UM.Concept.AdaptLevel += 1
    [add statement]
    )
  [add statement]
  )
  [label]
[add statement]
```

You chose to

[✓ Create a new node]

Select the type of the new node: while

Figure 9. Writing the converger generic strategy in MOT.

The implementation for converger is similar to the diverger one from the point of view of *specialization* and *generalization* conditions. The difference is that the learner should be able to tune more parameters, and choose how long the strategy is applied. The adaptation level is kept low at all times, although it varies slightly with the student achievement [10]. In such a way, different adaptive strategies, corresponding to instructional strategies aimed at different learning styles, can be authored in MOT. The adaptive language used is being developed and refined within an EU project, ADAPT.

4.3 Assessing Learning Styles with MOT

Here, just a few words need to be mentioned to make the MOT-AHA! parallel about the possibility of assessing of learning styles. The adaptation language in MOT was written to serve for the description of various adaptive behaviors. We expected, as mentioned in the previous sub-section, that some authors would want to create these adaptive strategies, while others would be content with just using them. It is therefore possible to determine the entry point for the application of one strategy or another via traditional questionnaires. However, the scope of the adaptive strategies written with MOT is not, as said, limited to implementation of instructional strategies corresponding to specific learning styles. We could envision a possible adaptive strategy that just monitors the browsing behavior of a learner, changing as a result some user model variables that define the user's preferred learning style, for instance.

Moreover, the MOT environment also has another interesting feature that can be exploited for the same purpose: MOT allows the extension of the adaptation language with new *adaptive procedures*. The definition of these procedures is very much the same as that of adaptive strategies, with the exception of the fact that procedures can be embedded into adaptive strategies. In other words, adaptive procedures should work the same way as other adaptation language constructs (Figure 10, 11).

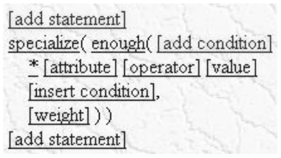
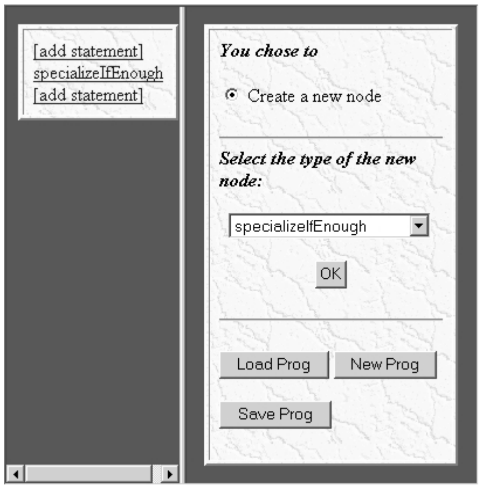

```
[add statement]
specialize( enough( [add condition]
   * [attribute] [operator] [value]
   [insert condition],
   [weight] ) )
[add statement]
```

Figure 10. Procedure specializeIfEnough.


```
[add statement]
specializeIfEnough
[add statement]
```

You chose to

[✓ Create a new node]

Select the type of the new node: specializeIfEnough

Figure 11. Using Procedure specializeIfEnough.

Figure 10 shows a procedure defined as an extension to the 'simple' *specialize* adaptation language construct (specialize if enough conditions are fulfilled). Figure 11 exemplifies using (i.e., calling) the newly created adaptive procedure.

Here we only show this to illustrate that the same mechanism between adaptive strategies and adaptive procedures can be used to combine *monitoring strategies* with instructional strategies: an adaptive monitoring strategy can call one or more instructional strategies, transformed into instructional procedures. The monitoring strategy can make the selection between the instructional strategies with respect to some change in user model variables suggesting an increased inclination towards one or another learning style.

5. MOT TO AHA! TRANSFORMATION

Some first attempts to analyze the translation of MOT into AHA! have been done in [8]. The main problem is that MOT can define behavior both at instance and at a more general level. The instance level can be, in principle, easily translated into AHA!. The general level has to be interpreted before it can become AHA! adaptation engine material.

As already briefly discussed in [8], there are many different layers to take into consideration when doing this translation. Here we only discuss the translation of the mentioned layers, concentrating on the adaptation strategy translation.

The concept maps, such as in Figure 4, represent instances, so are easier to translate. Such a translation implies creating an XHTML (basic) resource file for every attribute in MOT[2].

Unlike in [8], where we were discussing the translation into AHA! 2.0, the translation of full concepts into AHA! 3.0 implies less duplications and copying of basic resources, as it allows composing of different sequences from basic resources via a new construct called '*objects*', as also used in section 3. This new structure is closer to the MOT representation. The main idea is that the MOT grouping of attributes (as different aspects of a concept that should appear when certain instructional strategies are triggered) can be translated into another set of XHTML files, that contain lists of 'objects', pointing to the first set of created XHTML files (as shown in Figure 12). The actual conditions that determine which (or how many) of the alternatives are really shown to the student are written in AHA! rules during translation from the adaptive strategies, as shall be seen later.

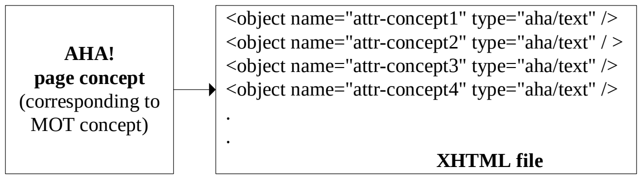

AHA!
page concept
(corresponding to
MOT concept)
<object name="attr-concept1" type="aha/text" />
<object name="attr-concept2" type="aha/text" / >
<object name="attr-concept3" type="aha/text" />
<object name="attr-concept4" type="aha/text" />
.
.
XHTML file

Figure 12. Translating MOT concepts into AHA! concepts.

Lesson translation into AHA! structure follows a similar fashion to the translation of the contents to be conditionally included (presented) for concepts. (Figure 13) To enforce the hierarchy and order relationship, the XHTML files translating lessons contain, beside the list of object alternatives, also a separate, ordered list of child sub-concept pointers. The children list can also be only partially desirable, depending on the instructional strategy, so the implementation is again via the new 'object' paradigm in AHA!. Moreover, a small trick is here necessary, as for children we really only want the link displayed and not the content of the child node – fact which causes in AHA! the need of creating extra concepts containing just a link each to a respective child concept.

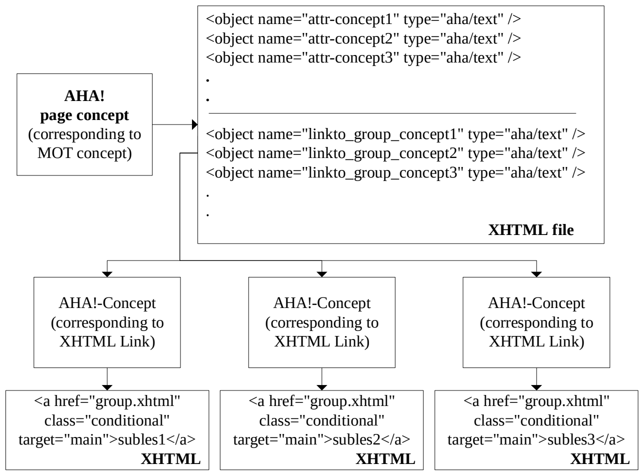

AHA!
page concept
(corresponding to
MOT concept)
<object name="attr-concept1" type="aha/text" />
<object name="attr-concept2" type="aha/text" />
<object name="attr-concept3" type="aha/text" />
.
.
<object name="linkto_group_concept1" type="aha/text" />
<object name="linkto_group_concept2" type="aha/text" />
<object name="linkto_group_concept3" type="aha/text" />
.
.
XHTML file
AHA!-Concept (corresponding to XHTML Link)
AHA!-Concept (corresponding to XHTML Link)
AHA!-Concept (corresponding to XHTML Link)
<a href="group.xhtml" class="conditional" target="main">subles1</a>
XHTML
<a href="group.xhtml" class="conditional" target="main">subles2</a>
XHTML
<a href="group.xhtml" class="conditional" target="main">subles3</a>
XHTML

Figure 13. Translating MOT lessons into AHA! concepts.

Beside these obvious, content-related translations, also some translations based on the internal structure in MOT and AHA! have to be performed, such as *Name* and *Id* translations. This may sound all a little bit technical and complicated, but it is only the easier part of the translation.

Translating adaptive strategies, especially generic instructional strategies, of the type that can be reused, is the most difficult task. For instance, a test on a value of a generic attribute will have to be added to each and every concept in the translated AHA! concept map. There is also a positive side of this – it is a proof of the compression power of a generic adaptive rule, which can imply great numbers of instance adaptive rules. In particular, the translation of an adaptive strategy affects the *action*, *assignment* and *attribute* tables of the AHA! database the selected concept map is placed in.

Each generic adaptation language construct in the adaptation strategy has to be translated into a number of IF-THEN rules for AHA!, and then applied to all concepts in a given AHA! concept database. To illustrate this process, as well as the problems that can occur during it, we select a very education-oriented construct from MOT, *specialize* (and its counter-part, *generalize*; see Figure 14) and discuss the translation. These constructs use the tree structure (of both conceptual and lesson layers) in order to go up and down the tree, respectively [10].

The way we would want the translation of:

SPECIALIZE(condition)

is:

If *condition* Then ***show*** *child(current_concept)*

[2] This only means adding a header and a footer to the attribute and saving it into a file with unique name, <file-name>.xhtml.

This could only be implemented as such in AHA! if the children of each concept would appear as objects included into a page associated with that concept. This doesn't make sense if we want to represent more than one hierarchical level, or if these concepts have been already translated into independent AHA! concepts, as described above.

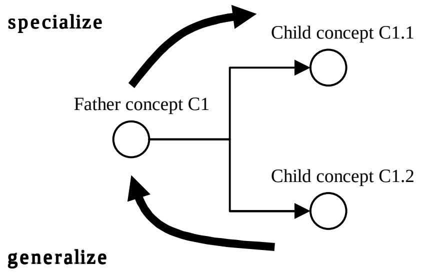


Figure 14. Generalization versus Specialization

```
<?xml version="1.0"?>
<!DOCTYPE concept SYSTEM 'concept.dtd'>
<concept>
  <name>C1.1</name>
  <description></description>
  <expr></expr>
  <attributes>
    <attribute>
      <name>access</name>
      <description>triggered by page access</description>
      <default>false</default>
      <type>3</type>
      <actions>
                <action>
                  <expr>C1.1.suitability</expr>
                  <trigger>true</trigger>
                  <truestat>
                    <assignment>
                      <variable>C1.1.visited</variable>
                      <expr>100</expr>
                    </assignment>
                  </truestat>
                  <falsestat />
                </action>
                ...
      </actions>
      <readonly>true</readonly>
      <system>true</system>
      <persistent>false</persistent>
    </attribute>
    <attribute>
      <name>suitability</name>
      <description>the suitability of this page</description>
      <default> C1.condition && C1.visited==100</default>
      <type>3</type>
      <actions />
      <readonly>true</readonly>
      <system>false</system>
      <persistent>false</persistent>
    </attribute>
   ...
  </attributes>
  <resource>file:/<path>/C1_1.xhtml</resource>
</concept>
```

Figure 15. Specialization rule for child concept C1.1.

So, an alternative, quite curious[3] solution has to be found. Each (child) concept in the new AHA! concept map has to be attached a rule specifying that it is ready to be used if the condition is satisfied and the father concept has been accessed. That means, for the child concept C1.1 in Figure 14, the behavior in Figure 15 has to be attached. The opposite has to happen in order to generate the *generalize* relation.

This is only an example of one adaptation language construct. As can be seen in Figures 7, 8, adaptive strategies, or adaptive procedures (Figure 10) can contain many more such constructs. The translation is done from the authoring interface via a frame window, as shown in Figure 6. The information contained in one adaptive strategy has to be distributed over several concept behavior descriptions in AHA!. The actual translation is done into MySQL database tables, but we have shown the XML translation in Figure 15 because of ease of reading. Moreover, AHA! provides a very handy functionality of translating in both directions between the MySQL version of the concept behavior and the XML version.

6. DISCUSSION AND CONCLUSIONS

In this paper we have presented two different views upon introducing learning styles in adaptive hypermedia systems: the adaptive hypermedia *engine pull* and the adaptive hypermedia *author push*. To illustrate these two views, we exemplified them with two systems: AHA!, a well-known adaptive hypermedia system [1], with its Graph Author tool, and MOT, a high-level adaptive hypermedia authoring system [19].

We believe that it is important to study these two perspectives, as the one tells us what authors might *want* to see their educational adaptive hypermedia do, whereas the other one tells us what such systems *can* do at present.

Another complementarity these two systems show is given by the type of authoring they allow: the *schema* level authoring, as in MOT, and the *instance* level authoring, as in AHA! (possible also in MOT but not shown in this paper).

It is interesting to address authoring at the different levels, the schema as well as the instance level, as authors themselves have different goals and understanding levels [10]. Some authors may prefer to make all the necessary specifications by hand, which gives them full control over the adaptation, whereas others may want to give higher level specifications, leaving the system to perform the rest for them automatically.

The paper also showed that the distinction only exists in the authoring tools. Structures authored with AHA!'s Graph Author or with MOT can both be translated to concept structures and adaptation rules used by the AHA! engine, or to other adaptive engines. (In the ADAPT project a compiler from MOT to WHURLE [3] is being developed for instance.)

As we are no psychologists, we do not recommend any particular instructional strategy for a particular learning style. We only can implement various instructional strategies as specified by the cognitive science literature and provide authors with tools that

[3] Curious because it works in a different direction than the original specialize relation.

allow them to define adaptive strategies and specify which instructional strategies should correspond to which learning style.

From the end-user side perspective, we assume that it is always important to provide them with different teaching strategies while using an application. So an option for them is to try different ones and select the one which corresponds better for them. However, an unresolved issue is how to ensure that the transition between learning styles or teaching strategies is smooth, i.e. that the learner continually feels at ease with the way that both previously visited and new material is presented (using the new style).

7. ACKNOWLEDGMENTS

This work is supported by the NLnet Foundation and by the ADAPT project (101144-CP-1-2002-NL-MINERVA-MPP). Our research uses the results of a great number of studies and findings, presented in the reference list. Information about AHA! (and the software) can be found at [1]. Information about MOT (and the software) can be found at: [19].

8. REFERENCES

[1] AHA!. http://aha.win.tue.nl.

[2] LSAS. http://www.archives.ecs.soton.ac.uk/users/nb99r/intro_short/frame.htm.

[3] Brailsford, T.J.; Stewart, C.D.; Zakaria, M.R. Moore, A. (2002). Autonavigation, Links and Narrative in an Adaptive - Based Integrated Learning Environment. 11th Intl. World Wide WebConference (2002), Hawaii, May 2002.

[4] Brusilovsky, P. Adaptive hypermedia. User Modeling and User Adapted Interaction,11(1/2), (2001), 87-110.

[5] Brusilovsky, P. Developing adaptive educational hypermedia systems: From design models to authoring tools. In: T. Murray, S. Blessing and S. Ainsworth (eds.): Authoring Tools for Advanced Technology Learning Environment. Dordrecht: Kluwer Academic Publishers, 2003.

[6] Carver, C.A., Howard, R.A., and Lavelle, E. Enhancing student learning by incorporating learning styles into adaptive hypermedia. In Proceedings of ED-MEDIA '96 World Conf. on Educational Multimedia and Hypermedia (Boston, USA, 1996), 118-123.

[7] Cristea, A.I. Evaluating Adaptive Hypermedia Authoring while Teaching Adaptive Systems. SAC, Track ELS'04, ACM.

[8] Cristea, A.I., Floes, D., Stash, N., and De Bra, P. MOT meets AHA!. In Proceedings of PEG'03 Conference (St. Petersburg, Russia, July 2003).

[9] Cristea, A.I., and De Bra, P. Towards Adaptable and Adaptive ODL Environments. In Proceedings of AACE E-Learn'02 Conference (Montreal, Canada, October 2002), 232-239.

[10] Cristea, A.I., and Calvi, L. The three Layers of Adaptation Granularity. UM'03. Springer.

[11] Dagger, D., Wade, V., Conlan, O.An Architecture for Candidacy in Adaptive eLearning Systems to Facilitate the Reuse of Learning Resources. In Proceedings of AACE ELearn'03 Conference. (Phoenix, November 07-11, 2003).

[12] De Bra, P., Aerts, A. and Rousseau, B. Concept Relationship Types for AHA! 2.0. In Proceedings of the AACE ELearn'2002 Conference (Montréal, Canada, 2002), 1386-1389.

[13] Dunn, R., and Dunn, K. Teaching students through their individual learning styles: A practical approach. Reston, VA: Reston Publishing, 1978.

[14] Felder, R.M. and Silverman, L.K. Learning and teaching styles in engineering education. Journal of Engineering Education, 78(7), (1988), 674-681.

[15] Gilbert, J.E. and Han, C.Y. Adapting instruction in search of 'a significant difference'. Journal of Network and Computer applications, 22, (1999).

[16] Honey, P. and Mumford A. The Manual of Learning Styles, Peter Honey, Maidenhead, 1992.

[17] Kolb, D. A. Experiential learning experience as the source of learning and development, New Jersey, Prentice Hall, 1984.

[18] Kwok, M. and Jones, C. Catering for different learning styles, Association for learning Technology (ALT-J) 3, 1, (1985), 5-11.

[19] MOT. http://wwwis.win.tue.nl/~acristea/mot.html.

[20] Paredes, P. and Rodrigues, P. Considering sensing-intuitive dimension to exposition-exemplification in adaptive sequencing. In Proceedings of the AH2002 Conference, (Malaga, Spain, 2002), 556-559.

[21] LOM standard. http://ltsc.ieee.org/wg12/.

[22] Grigoriadou, M., Papanikolaou, K, Kornilakis, H. and Magoulas, G. INSPIRE: an intelligent system for personalized instruction in a remote environment. In Proceedings of 3rd Workshop on Adaptive Hypertext and Hypermedia (Sonthofen, Germany, 2001), 13-24.

[23] SCORM standard. http://www.adlnet.org/index.cfm?fuseaction=scormabt.

[24] Stern, M. and Woolf, P. Adaptive content in an online lecture system. In Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-based systems (Trento, Italy, 2000), 291-300.

[25] Triantafillou, E., Pomportsis. A, and Georgiadou, E. AES-CS: Adaptive Educational System base on cognitive styles. In Proceedings of the AH2002 Workshop (Malaga, Spain, 2002), 10-20.

[26] iWeaver. http://www3.cti.ac.at/icl/archive/presentation/wolf.pdf

[27] Wu, H. A. Reference Architecture for Adaptive Hypermedia Applications, doctoral thesis, Eindhoven University of Technology, The Netherlands, ISBN 90-386-0572-2.