Adaptation languages as vehicles of explicit intelligence in Adaptive Hypermedia

Natalia Stash*

Faculty of Mathematics and Computer Science,
Eindhoven University of Technology,
Postbus 513, 5600 MB Eindhoven, The Netherlands
E-mail: nstash@win.tue.nl
*Corresponding author

Alexandra I. Cristea

Department of Computer Science,
University of Warwick,
Coventry CV4 7AL, UK
E-mail: acristea@dcs.warwick.ac.uk

Paul De Bra

Faculty of Mathematics and Computer Science,
Eindhoven University of Technology,
Postbus 513, 5600 MB Eindhoven, The Netherlands
E-mail: debra@win.tue.nl

Abstract: This paper deals with a new challenge in Adaptive Hypermedia (AH) and web-based systems: finding the *adaptation language* to express, independently from the domain model or platform, the *intelligent, adaptive behaviour of personalised web courseware*. The major requirements for the ideal language are: *reuse*, *flexibility*, *high level semantics*, and *ease of use*. To draw closer to this ideal language, we compare two such language proposals: LAG, a generic adaptation language, and a new XML adaptation language for *Learning Styles* (LS) in AHA!, LAG-XLS.

Keywords: plug-and-play intelligence; learning styles; LS; user modelling; adaptive hypermedia; AH; authoring of Adaptive Hypermedia.

Reference to this paper should be made as follows: Stash, N., Cristea, A.I. and De Bra, P. (2007) 'Adaptation languages as vehicles of explicit intelligence in Adaptive Hypermedia', *Int. J. Continuing Engineering Education and Life-Long Learning*, Vol. 17, Nos. 4/5, pp.319–336.

Biographical notes: N. Stash received her PhD from the Eindhoven University of Technology (TU/e), The Netherlands. She is currently a researcher in the CHIP project (Cultural Heritage Information Personalisation) between the Rijksmuseum (Amsterdam), TU/e and Telematica Institute (Enschede). Her research interests include AH, semantic web technologies, LS.

A.I. Cristea received her PhD from UEC, Tokyo, Japan and is currently an Associate Professor at University of Warwick, where she heads the Intelligent and Adaptive Systems group. Her research interests include AH and authoring, UM, semantic web, NN, AI, ITS, web-based educational systems. She co-authored over 100 papers and co-chaired workshops on AH authoring.

P. De Bra is a full Professor at TU/e; he heads the Database and Hypermedia group. He is one of the pioneers of AH, and is (co-)author of over 100 publications on databases, hypermedia, web-technology and adaptive systems. He organised numerous workshops on AH and was PC chair of the AH conference, Malagà, 2002 and general chair, Eindhoven, 2004.

1 Adaptation language as an intermediate platform

1.1 Benefits on an intermediate platform

Creation and authoring of adaptive, 'intelligent' courseware can be a cumbersome process (Cristea and de Mooij, 2003). To create a personalised, rich learning experience for each user, not only the actual content of the lesson has to be prepared, but much more. Catering for different user needs means creating (labelled) alternatives of the same content, which ensures multiple paths through that content. This content organisation is often called in the AH literature the creation of the *Domain Model* (Wu, 2002). Adaptive dynamics design also embraces specifications of what the user expected. Often, this is done in the form of user attributes, specified in what is usually called a *User Model* (UM) (Brusilovsky, 2001). Moreover, in the educational field there is a serious need for a separate *Pedagogical Model* (Dagger, 2004), which establishes adaptation and interaction types for the different kind of learners, according to pedagogic strategies (Coffield et al., 2004). Finally, machine constraints have to be considered, via, e.g., a *Presentation Model* (Cristea and Mooij, 2003). All these are connected via an *Adaptation Model* (Wu, 2002). Therefore, authoring of personalised courseware can be a difficult and costly process. There are different ways of striving towards alleviating what can be called the 'authoring problem'. Two ways of dealing with it are:

- To consider the difficulty of the first-time authoring process unavoidable and to concentrate on improving *reuse* capabilities. In this way, the cost could be reduced by reuse of previously created material.
- To lighten the authoring burden, by moving away from the platform dependent authoring style, common especially in AH, towards *platform independent* authoring.

In this paper we concentrate on the first solution, *reuse*. The solving of one issue actually offers solutions of the second, but in this paper, this issue is not followed-up. The reusable items can be the *static parts* of the authored courseware (such as the content of the domain model) and the *actual dynamics*. Most of existing standards address only static reuse. Examples of such e-learning standards are: *Learning resources*: metadata: IEEE-LOM (Learning Object Metadata) (IEEE-LOM, 2002), Dublin Core (http://dublincore.org), ADL-SCORM (Reusable learning content as 'instructional objects', http://www.adlnet.org/index.cfm?fuseaction=scormabt); *Data exchange*: IMS-Content Packaging (CP) IMS-CPS, *Data formats*: IMS-QTI (Question and Test Interoperability), *Education Modelling Languages* (http://eml.ou.nl/eml-ou-nl.htm), learning paths specifications: IMS-LD (Learning Design) (IMS-LD, 2003).

This paper looks instead of the static reuse at what we consider the most challenging and difficult task for the reuse topic: reuse of *intelligent, adaptive behaviour*. The approach taken here is to look at *defining Adaptation Languages*, as vehicles for the intelligent behaviour of the AH. This would be equivalent to *exchanging not only the ingredients, but the recipes as well*. This extraction and separate expression of semantically relevant, reusable, explicit 'artificial intelligence' of AH systems can also feedback to AH systems as analyser of the level of 'intelligence' they can provide. As requirements we enforce that the adaptation behaviour described by the adaptation languages should be reusable and the language extensible – the latter as it may be necessary to be able to create new strategies that need the addition of new elements. Ideally, the language should use or extend the emerging web standards, which will enhance reusability and compatibility with current implementations.

1.2 Elements of course dynamics in AH

In order to extract the main elements of personalised, intelligent course dynamics, we work on a concrete case of representing personalisation based on LSs. We have chosen LS in order to show possible adaptation to some higher-level traits rather than just the properties of the subject domain. LS and their effects on learning have been examined most carefully in Coffield et al. (2004) and Holodnaya (2002). These reviews show that there is no *commonly accepted point of view* on the usefulness or effectiveness of LS application in learning environments. However, firstly, we consider that it is good for the learner to be aware of his LS, in order to know what his strengths and weaknesses are. Secondly, experiments that do not show any significant difference in individuals' performance for matched vs. mismatched LS are usually done on skilled learners. Other experiments show that less skilled learners perform better in LS-matched conditions. Thirdly, researchers suggest that mismatching can be advantageous, as it allows individuals to develop new skills, by forcing them to adopt unfamiliar techniques. Therefore we conclude that LS-based strategies are useful in a variety of situations and combinations thereof. Hence we consider that it is important to provide authors, instructors and learners with various *instructional strategies* (Coffield et al., 2004), including LS-based ones.

Our review of research on and application of LS in AH (Stash et al., 2004) shows that existing systems can provide adaptation to the learner in terms of *content adaptation* (Brusilovsky, 2001), *navigation paths* (Brusilovsky, 2001) or usage of *multiple navigational tools* (de La Passardiere and Dufresne, 1992). These adaptation types limit the possible response of the system to accommodate the different LS. The most frequently used elements of instructional strategies we have found in Adaptive Web-based Education literature are

- *Selection of media items* to accommodate different learner preferences; this can also be extended to different LS. For instance, the same information (or the same concept) can be presented in various ways, by using alternative media types (Brusilovsky, 2001) – audio, video, image, text, etc. Depending on the learner's style a certain item (or group of items) may be included into the final presentation. E.g., in LS terms, we can say that *verbalisers* (Riding and Buckle, 1990), who prefer textual information, may be presented with text and possibly spoken audio; whilst *imagers*, who prefer pictorial information, can be shown images, diagrams, graphs, charts or other items about the same concept (Riding and Rayner, 1995). The selection process can be applied not only to media items, but also to other types of items.

- *Ordering information or providing different navigation paths*. The order in which information items are processed can be based on learner needs. E.g., some learners prefer to learn things by *doing something actively* first whilst others prefer to *collect data first* (corresponding to the *active* and *reflective* LS, respectively). Moreover, some learners tend to learn through a linear, step-by-step, logical and systematic process, whilst others want to see the big picture before they tackle the details (corresponding, respectively, to *sequential* and *global* LS (Felder and Soloman, 2000)).
- *Providing learners with navigational support tools.* Depending on the learner preferences, different learning tools can be provided. In terms of catering for LS, for example, *field-dependent* (Witkin et al., 1977) learners can be provided with a concept map, graphic path indicator, advanced organiser, etc., in order to help them organise the structure of the knowledge domain. Alternatively, *field-independent* learners might be provided with a control option showing a menu from which they can choose to proceed with the application in any order (Triantafillou et al., 2002).

There are fewer systems which attempt to provide, along with various instructional strategies, some mechanisms for *inferring* the learner's preferences based on his/her actions and selections. For example, MANIC (Stern and Woolf, 2000) uses a Naïve Bayes Classifier to reason about the learner's preferences in terms of explanations, examples and graphics.

1.3 A view on adaptive learning strategies

To distinguish between the different types of strategies, we need, beside the previous list of elements of instructional strategies, a high-level classification based on their overall semantics. From the analysis of the literature, we extract a classification based *application range*, as follows

- *Instructional strategies* – Define how the adaptation is performed. Namely the adaptation rules specified in the strategy are used to adjust the presentation to the learner with a particular learning preference (Berlanga and Garcia, 2003), style (Stash et al., 2006) or need (Brusilovsky, 2001). We argue that it is very important to provide several instructional strategies for an application so that the learners or tutors can select the most appropriate.
- *Instructional meta-strategies – inference* or *monitoring strategies* – are applied in order to infer the learner's preferences during his/her interaction with the system (Stash et al., 2004, 2006). These strategies can not completely replace the existing psychological LS questionnaires; however they can be used as a simplified, unobtrusive way to infer the learner preferences corresponding to these styles via their browsing behaviour.

The first type of strategy is more common, but the second type is novel and requires some clarification. A meta-strategy can, for example, track the learner's preferences by observing his/her interactions with the system (Stash et al., 2006). It can track some repetitive patterns in the learner's behaviour, like accessing particular types of information (if a choice is available). It can observe that the user has a preference for textual information, which is typical for a learner with *verbaliser* style, or, on the contrary, that the user has a preference for the pictorial representations (*imagers* or *visualisers*). It can also trace the navigational paths: browsing through the learning material in breadth-first order – typical for the learners with *field-dependent* or *holist* style – vs. navigating in depth-first order, which might indicate a learner with *analytic* style (Coffield et al., 2004). Meta-strategies of this type update some UM parameters that can be used later on for selecting a particular instructional strategy. These parameters can indicate what the system 'thinks' the learner's preferences are. In most existing systems that provide LS adaptation, information about LS and preferences is not updated during the interaction. However, LS preferences might actually change, depending on various circumstances (Coffield et al., 2004) (for instance on the mood, time of day, subject, etc.). Meta-strategies could trace if the preferences specified by the learner when he begins working with the system stay the same or change. In case the learner's behaviour is different than initially specified, a strategy corresponding to another LS might be suggested. Other examples of UM parameters which can be influenced by the actions specified in the meta-strategies are: level of difficulty of the material presented to the learner, link colours, etc. These actions occur when the learner accesses the concepts of an application. Therefore, an adaptive meta-strategy is a '*strategy about strategies*', that can switch, explicitly or implicitly, between adaptive strategies.

According to the type of adaptation provided, we can refine the classification of adaptation strategies by analysing the *external* (interactive) *actions* occurring is shown in Table 1.

Table 1 Refined classification of *external actions* in adaptive strategies

Basic actions on items	Selection
	Showing the content of an item
	Showing a link to an item
Hierarchical actions on items	Actions on child items
	Actions on parent item
Actions on groups of items (e.g., siblings)	Ordering
	Performing 'actions on items' on each group item
Actions on the *overall environment*	Changing the layout of the presentation

These are actions, which directly determine changes in what the users sees. Similar to meta-strategies, instructional strategies also perform *internal actions* (mainly UM updates). These actions can be classified according to traditional UM classification and are therefore not further explained here.

In the following, we will show how we have used our analysis of the state of the art and of the standards as well as an existing adaptation language, LAG, to create a new language, based on web technologies.

2 LAG: model, language and implementation

The LAG model is a specification of the Adaptation Model, as defined by the LAOS model (Cristea and Mooij, 2003). LAOS is a generic model for authoring of AH, detailing a *Domain Model*, a *UM*, and a *Goal and Constraints Model* (which becomes the *Pedagogical Model* for educational applications), a *Presentation Model* (dealing with the different machine-oriented ways of presenting the same information: e.g., different colours, formats, etc.) and an *Adaptation Model*. For the purpose of this paper, we only focus on the Adaptation Model, the sub-model that allows reuse of dynamics, as opposed to current standards which are mainly focused on static material reuse. The Adaptation Model is the one representing the Artificial Intelligence component of the Adaptive System.

2.1 LAG model and language: review

To enable reuse of dynamics in personalisation and adaptation, the adaptation model used was a 3-layer model, LAG (Cristea and Calvi, 2004). The details are skipped. In short, LAG consists of an *Adaptation Assembly Language*, corresponding to the typical IF-THEN rules in AH; at intermediate level, of a semantic *Adaptation Language*; and at the highest level, of *Adaptive Strategies* or *Adaptive Procedures*. These strategies/ procedures[1] are containers for the adaptation program (which details, in machine readable adaptation language, how the adaptation is performed). In addition, each strategy has a description (semantic label) in natural language, which can be directly used by authors to select a specific, ready-made strategy for their course. In this way, course *content* creation and the creation of *adaptation dynamics* for that course are kept separate, and can be performed by differently specialised authors (roles), at different times.

As an instantiation of the *Adaptation Language* in the LAG model, the *LAG Language* (Cristea and Calvi, 2004) was introduced. This language uses for syntax the LAG grammar (Figure 1), and is the basis of an Intermediate Platform specification for adaptation dynamics. Concretely, the LAG Language provides the building blocks for the creation of Adaptation Strategies. Figure 1 shows the new, extended version of the LAG grammar, improved after authoring usability tests (Cristea and Cristea, 2004), as well as conversion and reuse tests (into two delivery systems, AHA! (http://aha.win.tue.nl) and WHURLE (Moore et al., 2001)).

Figure 1 The extended LAG grammar

```
PROG  DESCRIPTION VARS
    INITIALIZATION  IMPLEMENTATION
DESCRIPTION  "// DESCRIPTION" "text"
VARS  ATTRIBUTE | (VARS)+ "," VARS
INITIALIZATION  "initialization("STATEMENT ")"
IMPLEMENTATION
              "implementation("STATEMENT ")"
STATEMENT  IFSTAT | WHILESTAT |
    FORSTAT | BREAKSTAT | GENSTAT |
    SPECSTAT | COMMENT | (STATEMENT) *
    STATEMENT | ACTION
IFSTAT  "if" CONDITION "then ("STATEMENT")"
WHILESTAT  "while" CONDITION "do
              ("STATEMENT")" [TARGETLABEL]
FORSTAT  "for" RANGE "do ("STATEMENT")"
              [TARGETLABEL]
BREAKSTAT  "break" SOURCELABEL
GENSTAT  "generalize("CONDITION ")"
SPECSTAT  "specialize("CONDITION")"

ACTION  ATTRIBUTE OP VALUE
COMMENT  "//" "text" | (COMMENT)*
                              COMMENT
CONDITION  "enough("(PREREQ",")+
                       VALUE ")" | PREREQ
RANGE  "integer"
PREREQ  ATTRIBUTE COMPARE
                                    VALUE
LABEL  "text"
TARGETLABEL  "text"
SOURCELABEL  "text_label_a"
ATTRIBUTE  GENCONCEPT |
               SPECCONCEPT
GENCONCEPT  "CM_type.concept.attr"
              | "CM_type.concept.attr_z"
SPECCONCEPT
               "CM_x.concept_y.attr_z"
OP  "=" | "+=" | "-=" | ".="
COMPARE  "==" | "<" | ">" | "in"
VALUE  "text"
```

The figure describes the components of an adaptive strategy *prog*. Each strategy has four main parts: *description*, *variable declarations*, *initialisation* and *implementation*. The description is a comment for the human reader (the author who has to decide if to apply this strategy). The variables are a new addition, to prevent overlaps and clashes if multiple strategies are applied on the same course. The two phases, of initialisation and implementation, are also new. The *initialisation* should set all the variables in use during the strategy, before the actual interaction of the strategy with the user (learner) occurs. It also establishes what learning items have to be shown to the user from the very beginning. The *implementation* part contains the user interaction and activity description. Initialisation and implementation are built from statements. These building blocks are the basis of the current version of the LAG language. The adaptation language also allows assembly language statements, such as IF-THEN statements. However, it also contains more general programming statements, such as WHILE, FOR, and BREAK statements, and comments. The most specific statements are the SPECIALIZE and GENERALIZE statements, that allow the user to go down, or up the learning item hierarchy respectively – depending upon the fulfillment of certain conditions. These statements use the *structure* of the learning material, therefore have greater semantics for authors familiar with the learning material. The conditions are either prerequisites, or combinations of ENOUGH prerequisites. The value in the latter construct is a number, establishing how many of the prerequisites have to be fulfilled.[2] In such a way, more complex AND-OR combinations of conditions can be obtained.

The details of the grammar have been simplified a little. However, it is important to remark that the ATTRIBUTES used in initialisations, actions and comparisons can be of two main types: GENERIC or SPECIFIC. The specific attributes refer to an *instance* of the learning material, whereas the generic attributes refer to materials of a given *type*. Therefore, strategies can be written general enough to be able to be applied to any given set of learning materials, instead of belonging to a specific course.

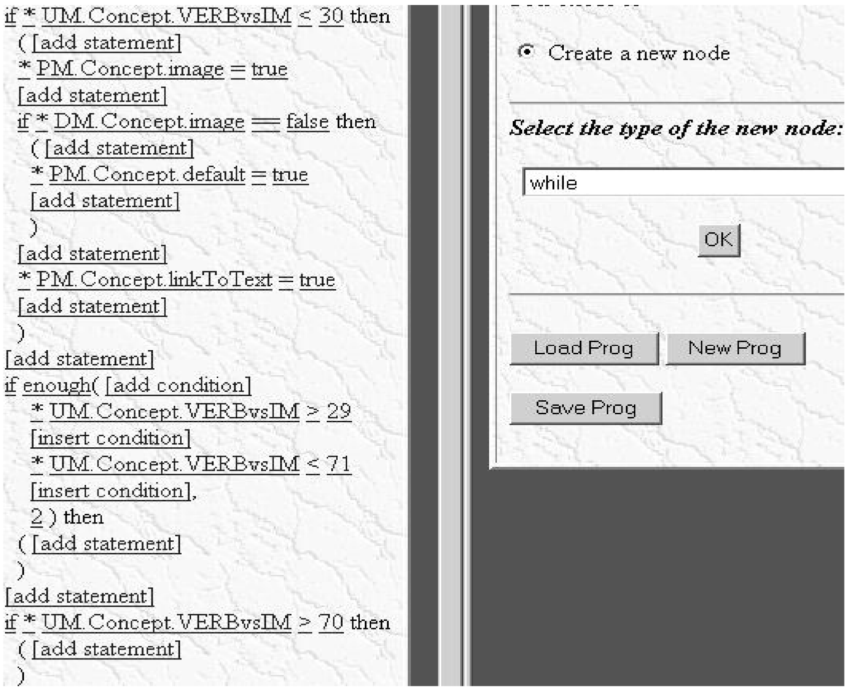
From the strategy classifications in Section 1.3, LAG can create both *strategies* and *meta-strategies*, as shown in the following section. From the point of view of actions, LAG supports *selection*, *showing of content of an item*, *hierarchical actions*, *actions on group*s (except for ordering) and *actions on the overall environment*. *Ordering* is part of the Goal and Constraints Model in LAOS, and *links to items* can only be displayed if they are represented in the Domain or Goal and Constraints Models.

2.2 (Authoring of) Learning Styles (LS) with LAG

The LAG grammar was used as a basis of the MOT-adapt interface (Moore et al., 2001). This interface is depicted in Figures 2 and 3.

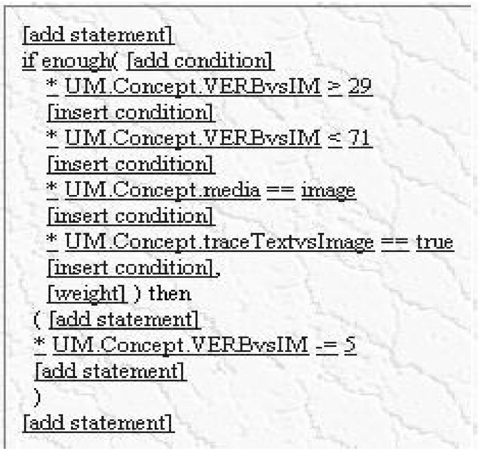
Figure 2 shows an *adaptive strategy* written in LAG for the *Verbalisers vs. Imagers* LS. The strategy specifies that verbalisers are presented with more textual information, whilst imagers receive more graphic information, such as pictures, diagrams, charts. The value of the VERBvsIM attribute is an integer between 0 and 100. A value between 30 and 70 indicates an unknown LS or a learner with no strong preference. Values above 70 indicate a *verbaliser*, values below 30 indicate an *imager*. The strategy is simplified and uses IF-THEN constructs only, to enable easy comparison with the LAG-XLS language presented in Section 3. *Showing the content of an item* in LAG is via action statements: ‘PM.Concept.item=true’;[3] e.g., ‘PM.Concept.image = true’ means ‘show an image’. UM attributes are similar: ‘UM.Concept.VERBvsIM <30’ means that the user is considered an imager. The strategy is detailed for imagers only, the other case being symmetrical.

Figure 2 The LAG grammar: imager strategy



```
if * UM.Concept.VERBvsIM ≤ 30 then
  ( [add statement]
  * PM.Concept.image = true
  [add statement]
  if * DM.Concept.image == false then
    ( [add statement]
    * PM.Concept.default = true
    [add statement]
    )
  [add statement]
  * PM.Concept.linkToText = true
  [add statement]
  )
[add statement]
if enough( [add condition]
    * UM.Concept.VERBvsIM ≥ 29
    [insert condition]
    * UM.Concept.VERBvsIM ≤ 71
    [insert condition],
    2 ) then
  ( [add statement]
  )
[add statement]
if * UM.Concept.VERBvsIM ≥ 70 then
  ( [add statement]
  )
```

Figure 3 LAG grammar: imager vs. textual meta-strategy (extract)



```
[add statement]
if enough( [add condition]
    * UM.Concept.VERBvsIM ≥ 29
    [insert condition]
    * UM.Concept.VERBvsIM ≤ 71
    [insert condition]
    * UM.Concept.media == image
    [insert condition]
    * UM.Concept.traceTextvsImage == true
    [insert condition],
    [weight] ) then
  ( [add statement]
  * UM.Concept.VERBvsIM -= 5
  [add statement]
  )
[add statement]
```

Figure 3 shows an example of an *adaptive meta-strategy* written based on LAG, which detects which type of concepts the user has preference for: *images* or *text*, then modifies the UM accordingly. The adaptation language constructs and variables are similar to the ones in Figure 2.

Concluding, we can say that LAG allows reusable dynamic representations at different levels: at *adaptation language level*, by reusing the language constructs, and at *adaptation strategy level*, by reusing adaptive procedures as new language constructs, but also by reusing whole adaptive strategies (by applying them to different domain maps and user maps, or exporting them to other systems).

3 LAG-XLS: a new XML Learning Style (LS) adaptation language

LAG-XLS started with the purpose of taking over these advantages of dynamic reuse, whilst adding new research results, summarised in the review on the most frequently used instructional methods to support LS (presented in Section 1.2). LAG-XLS instantiates the Adaptation Language layer of the *LAG model* as well, but with different goals. In LAG-XLS we try to express the first two methods: *selection of media items* (or selection of a particular type of information in general) and *ordering information* – in a simple and straightforward manner. Moreover, the refined classification of actions in Table 1 is applied directly. We have based LAG-XLS on the *LAG language*, and have tried to alleviate some of its problems, whilst at the same time simplifying parts of it. This is based on our desire to identify more specific language constructs aimed at LS strategies, as well as at being completely AHA! compatible. We initially decided to create an XML based language, with the aim of aligning it with semantic web research (http://www.w3.org/2001/sw/). Reusability is achieved in LAG-XLS by specifying each strategy as a separate XML file. XML (E*X*tensible *M*arkup *L*anguage (http://www.w3.org/2000/xp/)) is a cross-platform, software and hardware independent tool for representing and transmitting data. XML elements for learning adaptive strategies are not yet defined in the literature, so we endeavoured to invent and describe our own elements.

LAG-XLS bases *selection* and *ordering* of concepts on the attributes and values of their sub-concepts,[4] as follows. The names of the attributes and their values indicate how these sub-concepts represent the parent concept. For instance, if the media attribute is audio, the sub-concept will represent an audio version of the concept. Another goal was that of expressing monitoring strategies. To achieve this, the adaptation language for AHA! contains elements specifying UM updates.

The resulting LAG-XLS language, corresponding to various strategies (extracted from what was previously implemented in 'adaptation assembly' form only, but also from literature review, and informed by the refined classification in Table 1) is presented in Figure 4. The meaning of the DTD *elements* and *attributes* is explained below.

- *strategy*: is the root element of a file corresponding to a strategy, attribute *name* – the name of the strategy
- *description*: is the strategy meaning; e.g., the corresponding learner model for which this strategy has been created
- *if*: a statement to specify if-then-else rules (currently we have only *if* statements within the *strategy* element, however we are thinking about applying other statements as well, like *for*, *while*, etc., as in LAG)
- *condition*: appears within an *if* statement; a Boolean expression which can contain user-related information; e.g., about the user's LS

- *then*: defines actions to be performed when the *condition* is satisfied
- *else*: an element defining an alternative set of actions.

The following elements are used to define *how* the adaptation is performed:

- *select*: selecting a concept representation from a number of existing ones to be included into the final presentation
- *sort*: sequencing different concept representations depending on the user's LS, and reordering them from most to least relevant.

Figure 4 LAG-XLS DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT strategy (description, if*)>
<!ATTLIST strategy name CDATA #REQUIRED>
<!ELEMENT description (#PCDATA)>
<!ELEMENT if (condition, then)>
<!ELEMENT condition (#PCDATA)>
<!ELEMENT then (action*, select?, setDefault*, sort?)>
<!ELEMENT action (UMvariable, expression)>
<!ATTLIST action attributeName CDATA #REQUIRED>
<!ELEMENT UMvariable (#PCDATA)>
<!ELEMENT expression (#PCDATA)>
<!ELEMENT select (showContent*, showContentDefault*, showLink*)>
<!ATTLIST select attributeName CDATA #REQUIRED>
<!ELEMENT setDefault (expression)>
<!ATTLIST setDefault attributeName CDATA #REQUIRED>
<!ELEMENT sort (linkTo*)>
<!ATTLIST sort attributeName CDATA #REQUIRED>
<!ELEMENT showContent (#PCDATA)>
<!ELEMENT showContentDefault (#PCDATA)>
<!ELEMENT showLink (linkTo, comment)>
<!ELEMENT linkTo (#PCDATA)>
```

The 'select' and 'sort' elements have an attribute 'attributeName'. The value is provided by the author depending on the aspects of the concepts he wants to include or reorder in the final presentation. For example, we have a concept which has several children representing it via different types of media. All the children concepts have an attribute 'media'. The value of this attribute for different concepts can be 'audio', 'video', 'text', 'image', etc. In the final presentation for various strategies (links to) media items can be explicitly included or not; similarly (links to) media items can be ordered in different ways:

- *showLink*: showing a link to the concept representation
- *showContent*: showing the content of the concept representation
- *showDefaultContent*: showing a default content specified by the author when no other representation is found for a particular concept
- *action*: specifies how the UM is updated; attribute *UMupdate* shows whether it is an absolute or relative update

- *UMvariable*: indicates which UM variable should be updated, namely which attribute of which concept
- *expression*: is the value used for UM update.

To exemplify the use of LAG-XLS, we follow the previous example from Section 2.2, and write a strategy for the *verbaliser* vs. *imager* LS. Due to of lack of space we present only the part of the XML file corresponding to the *imager* LS. To indicate that the user is either a verbaliser or imager we use a similar UM attribute as in the LAG example, Section 2.2, 'VERBvsIM', for the AHA! 'personal' concept.[5] In AHA! the LSs related attributes of this concept can be initialised via the registration form. The strategy in Figure 5 uses the XML adaptation language elements: *description*, *select*, *showContent*, *showLink*. It also uses traditional AH elements such as IF-THEN constructs. The meaning of the strategy is that if the user is an imager (personal.VERBvsIM <30)[6] then, for each concept which can be represented by different media types,[7] an 'image' representation is included in the presentation. If no 'image' representation exists, then the default representation provided by the author is used. The author can specify that links to other concept representations are included. In Figure 5 a link to a textual representation is inserted using the 'showLink' element.

Figure 5 Strategy of verbaliser vs. imager

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE strategiesList SYSTEM "strategy.dtd">
<strategy name="VerbalizerVersusImager">
  <description>Strategy "Verbal" vs. "Visual" style: Felder-Silverman Learning Model</description>
  <if>
   <condition>personal.VERBvsIM &lt; 30</condition>
   <then>
    <select attributeName="media">
     <showContent>image</showContent>
     <showContentDefault>default</showContentDefault>
     <showLink>text</showLink>
    </select>
   </then>
  </if>
  <if>
   <condition>personal.VERBvsIM &gt; 29 &amp;&amp; personal.VERBvsIM &lt; 71</condition>
   <then>…</then>
  </if>
  <if>
   <condition>personal.VERBvsIM &gt; 70</condition>
   <then>…<then>
  </if>
</strategy>
```

Next, we present a short example of an instructional meta-strategy, corresponding to the LAG meta-strategy in Figure 3. Here, the author specifies actions which are performed when the user accesses an AHA! concept (like increasing or decreasing the confidence of the system that the learner has a particular LS). We present only the part of the XML file indicating a decrease in the system's confidence that the user is a verbaliser (and an increase in the confidence that (s)he is an imager). In this strategy we use two new AHA! variables: *personal.VERBvsIM.initial* and *personal.traceTextvsImage*. These can be added by the author and initialised through the registration form. The first variable stores the initial value of the 'VERBvsIM' attribute. The second variable indicates whether the user wants the system to infer his preferences. For example, the user does not know his LS and wants the system to trace it. He might still let the system trace it even if he explicitly specified his LS. Whilst tracing, *personal.traceTextvsImage* is set to true. During the actual learner-system interaction, the user's repetitive accesses to pictorial representations increase the system's confidence that the user is a *imager*, indicated by the expression var:–5. *Var* means that the value can be changed by the author while applying the strategy to a particular application. In the strategy in Figure 6, the default is –5. The system traces the user's behaviour until the value of 'VERBvsIM' reaches a meaningful threshold (30 or 70); then the value of the attribute personal.traceTextvsImage will be set to false and tracing will stop. Afterwards, an instructional strategy corresponding to the new value of the 'VERBvsIM' will be suggested to the user.

Figure 6 Meta-strategy of verbaliser vs. imager

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE strategiesList SYSTEM "strategy.dtd">
 <strategy name="TextvsImagePreference">
  <description>Inferring the preference for textual or pictorial information</description>
  <if>
   <condition>personal.VERBvsIM.initial &gt; 29 &amp;&amp; personal.VERBvsIM.initial &lt; 71
&amp;&amp; concept.media == "image" &amp;&amp; personal.traceTextvsImage</condition>
   <then>
    <action UMupdate="relative">
     <UMvariable>VERBvsIM</UMvariable>
     <expression>var:-5</expression>
    </action>
   </then>
  </if>  …
 </strategy>
```

If the learner is not satisfied with an instructional strategy he can always inspect his UM and make necessary corrections. AHA! provides a tool that allows authors to create forms to let the learners change values of attributes of concepts in their UM. It is thus possible to create a form that lets a learner change their 'VERBvsIM' value.

This is an example of an XML adaptation strategy which can be reused by various authors. For their own applications, authors might create different versions of the verbaliser vs. imager strategy or the strategy for tracing the learner's preference for textual or pictorial information. They could use a different attribute of different type indicating the user's style (instead of 'VERBvsIM'); they might also specify a different range of values for the attribute and different kinds of adaptation using 'showLink', 'showContent' elements. They could specify as well a different set of actions for inferring the learner's preferences, limited only by the DTD.

LAG-XLS allows authors to apply generic adaptation rules. Moreover, the default values of the parameters in each rule can be replaced by the author.

Similarly to the LAG adaptation language, LAG-XLS can deal with *specific* as well as *generic* concepts. The examples presented so far only show dealing with generic concepts, specified by the variable 'concept'. While applying the strategy to an application, this name will be replaced with the specific concept names. Specific concepts can also be directly used in strategies: *NameSpecificConcept.Attribute = Value.*

Currently, a user friendly authoring tool for LAG-XLS is under development. It will allow authors to create strategies using the predefined set of elements (specified in the DTD). Authors should first use this tool to create adaptive strategies – resulting in a separate server-side XML file for each strategy. Skilled authors could also manually create or edit XML files corresponding to strategies, as shown in the examples. The created files will be stored into the author's private directory. A set of 'standard' strategies, reusable by all authors, is available and will be extended. If authors want to let others use some of their strategies, they would have to add them to the list of standard strategies.

4 Applying LAG-XLS to AHA!

To visualise strategies not only from the author's, but also from the delivery (learner's) point of view, we show how the Figure 5 strategy 'VerbaliserVersusImager' is converted for AHA! The author can create the Domain and Adaptation model for AHA! courses using a high-level authoring tool called Graph Author (De Bra et al., 2002). A new option added to it allows authors to choose which strategies to apply to a particular course, and in which order (when applying several strategies, order can be important). Authors might need to rewrite some information (e.g., parameters specified with 'var'). Otherwise, default values are applied. During saving, the AHA! concept relationships graph is translated into AHA! low-level (assembly) adaptation rules. Applied strategies may influence desirability of concepts and actions to be performed when concepts are accessed. Additional application pages (in XHTML format) might also be generated.

The 'VerbaliserVersusImager' strategy is applied to all AHA! concepts in the given course which have sub-concepts with an attribute 'media'. This has as effect the display of the content of the appropriate sub-concept, depending on the value of the 'media' attribute ('image', 'default' or 'text'), and a link to inappropriate sub-concepts. For the imager an 'image' should be included into the presentation. If an 'image' is not found then the system will look for a 'default'. As inappropriate sub-concepts are added as links, the learner can still follow a link to a 'text'.

In Figure 7 we show a simplified part of the parent concept structure after application of a strategy. This parent concept represents the conversion of an AHA! concept from the LAG-XLS language (Figure 6) to the AHA! low-level assembly language, which the AHA! system can deliver. It shows that the 'VERBvsIM' attribute value of the concept 'personal' influences the 'showability' attribute, which in turn determines the fragment displayed. Files generatedfile2.xhtml and generatedfile3.xhtml are needed because of some extra steps in the conversion. The 'text' concept is an AHA! object concept. Resources associated with this type of concept can only be seen if included into pages. Therefore, a new page resource file (e.g., generatedfile1.xhtml) that includes it has to be created, representing a viewable version of the 'text' concept, as follows:

```
<!DOCTYPE html SYSTEM '/aha/AHAstandard/xhtml1-strict.dtd'>
<html xmlns='ttp://www.w3.org/1999/xhtml'><body>
<object name='objectText' type='aha/text'/></body></html>
```

The goal of this resource file is to add a header wrapper to the AHA! object concept. The resource file uses an 'object' tag for conditional inclusion of objects. The specified type 'aha/text' does not mean that the object is a text; it can be any media item. It is used only as an indication that the object should be processed by the AHA! engine. Afterwards, a resource representing the AHA! parent concept has to be also generated – a page resource, if the AHA! parent concept is a page concept; or a fragment, if it is an object concept. The first case is that of *adaptive link destinations*: i.e., when the learner follows a link to a parent concept, the displayed content varies with the UM state. Therefore, the same link to a concept will point to different resources, depending on the UM. The second case results in *adaptation of the content*. This happens if the parent concept is a part of some other page. This page will contain different contents, depending again on the UM.

Figure 7 Example part of the generated structure for the AHA! concept

```
<concept>
 <name>conceptname</name><expr>true</expr>
 <attribute><name>access</name>
 <action>
   <if> personal.VERBvsIM &lt; 30</if>
   <then>conceptname.showability := 0</then>
  </action>
  <action>
    <if> personal.VERBvsIM &gt; 70</if>
    <then>conceptname.showability := 1</then>
  </action> …
  </attribute>…
  <attribute><name>showability</name>
    <casegroup>
    <defaultfragment> generatedfile4.xhtml</defaultfragment>
    <casevalue><value>0</value>
      <returnfragment>generatedfile2.xhtml</returnfragment>
    </casevalue>
    <casevalue><value>1</value>
      <returnfragment>generatedfile3.xhtml</returnfragment>
    </casevalue>
    </casegroup>
  </attribute>
</concept>
```

5 Empirical evaluation of LAG and LAG-XLS

Both LAG and LAG-XLS have been tested in practice with students. LAG has been tested in three different settings, in a 2004 Computer Science (CS) regular curriculum course on AH, in a 2005 User-System Interaction (USI) 2-week course on Adaptive Systems and User Modelling for Master Students, both at TU/e, and at a 1-week summer school course on Authoring of AH at Joensuu University, Finland. LAG-XLS has been tested during the CS course. For all settings, the students had to work both as *authors* and *end-users* of the AH material. For the LAG evaluations, students were asked to fill in SUS questionnaires and a generic questionnaire on a Likert scale. For LAG-XLS, they had a generic questionnaire to fill in, as well as the Felder-Solomon "Index of Learning Styles Questionnaire" (Felder and Soloman, 2000). Due to lack of space, these results are not detailed here. In general, students reacted positively to the idea of generic adaptation languages, the freedom of applying different intelligent strategies, LS, etc., easily on the same or different course material. However, they also expressed some dissatisfaction with some of the implementation and installation issues of the programs they worked with. In the LAG language case, these issues were taken into consideration directly, and different generations of students experienced different stages of the project. For the LAG-XLS case, students considered the experiment of creating adaptation strategies based on LS, and then comparing them to their own, pleasant, but not that easy. It is also noteworthy to say that students from CS, for instance, preferred creating their own strategies and declared so in the questionnaires, but USI students preferred reusing strategies created by others. This shows that this approach of allowing authors with different abilities access to adaptivity at different levels (some as re-users and others as creators) is a valid one.

6 Discussion and conclusion

Before we conclude on the two adaptive languages extracting artificial intelligence features of AH, as described in this paper, we first analyse the few comparable approaches found in the literature. Recently, similar attempts at defining a reusable representation for the system 'intelligence' and dynamics of web-based adaptive education environments have been researched and can be classified into the following categories, as follows

- *Adaptation languages.* In Berlanga and Garcia (2003), the authors define adaptive rules based on a collection of sets employing the IMS Learning Design (IMS-LD, 2003). These rules are only at the level of assembly language of adaptation (according to the classification in Cristea and Calvi (2004)), i.e., IF-THEN rules, but are enriched with extra semantics. For this, they use semantically labelled actions (such as show, hide, show-menu, sort-ascending, number-to-select, etc.). One problem with this approach is that it mixes the user adaptation (such as some material being *not recommendable* for a user) with the actual presentation of this adaptation (*hide it* from user). This problem is inherited from the strict adherence to the IMS-LD standard, which does not make this distinction. In the AH literature (Brusilovsky, 2001), however, the presentation of an item which is undesirable can vary from *hiding* to *colour-code marking* (e.g., 'Red' is undesirable). This type of presentation depends on the degree of control the learner can have within the learning environment. Moreover, the IMS-LD standard is especially aimed at collaboration, and not at personalisation.
- *Workflow models.* The COW platform in Vantroys and Peter (2003) as well as the WFMS in Cesarini et al. (2004) use workflow modelling for dynamics representation. However, in COW no personalisation or adaptation is envisioned. WFMS has a form of non-flexible adaptation, comparable with the conditional fragment inclusion technique in early AH (Brusilovsky, 2001).
- *Task composition models.* In Carro et al. (1999), tasks are modelled and alternative paths are created via AND and OR relations. This alternation seems to be more dynamic than the Simple Sequencing Protocol (2003). The problem is that the language used for task definition is very domain dependent.

LAG has already addressed many of these problems, as it is a higher level language that allows for an increased level of semantics. User adaptation and presentation are kept separate. The adaptivity degree allowed is extremely flexible (adjustable) and the language is not domain dependent. LAG has been evaluated in real life settings

(Cristea and Cristea, 2004). One important drawback is that it does not reflect the current web-standards.

The newly proposed LAG-XLS adaptation language aims at alleviating this last problem, as it is based on XML which is a W3C standard. The language is extensible and its XML syntax ensures web-readability and the capacity to export to different systems. LAG-XLS has also been evaluated in real life settings and the results reported in Stash et al. (2006). The focus of the new language is however slightly different from LAG, which is a more generic adaptation language. LAG-XLS specifically targets users' LS and the adaptive strategies corresponding to them, restricted by the DTD definitions and based on the action definitions in Table 1.

Currently we have defined and are experimenting in LAG-XLS with a number of *instructional* strategies other than the one represented in this paper, such as Active vs. Reflective, Auditory vs. Visual, Holist vs. Analytic, Field-Dependent vs. Field-Independent, Verbal vs. Visual learners as well as other *monitoring* meta-strategies, like inferring preferences for textual or pictorial information or reading in breadth- or depth-first order. We are thinking about other types of strategies the authors might need for their adaptive applications and the extension of the adaptation language to allow more complex rules and evaluate the usage of Web Ontology Language (OWL).

Both LAG and LAG-XLS instantiate the *LAG Model Adaptation Language*. This paper therefore demonstrates that separating the specific dynamics required for the complex issue of LS adaptive response is possible, and therefore paves the way for exportable adaptive strategies on a global scale and their integration into web standards. We have demonstrated this by comparing two adaptation languages, starting with what problems they solve, what their underlying model is, how they differ from other approaches, and what are their positive and negative aspects. Moreover, by making the 'intelligence' in the AH systems explicit, not only can these AH systems be analysed as to the extent of 'intelligence' they can represent; but also, in this way, the adaptive model is only weakly connected to the delivery engine, and can therefore be easily replaced with other alternative approaches of machine intelligence representation, such as fuzzy logics, neural networks, etc. In this way, the artificial intelligence part of the AH systems is clearly delimited and defined, and plug-and-play technology becomes applicable. Existing educational hypermedia can therefore be reused in new, adaptive and intelligent ways – however more research is necessary for establishing the requirements of merging at both syntactic and semantic levels.

Acknowledgements

This work is performed within the PROLEARN network, and supported previously by the NLnet Foundation, the Minerva ADAPT project (101144-CP-1-2002-NL-MINERVA-MPP) and the Minerva ALS (Adaptive Learning spaces) project (229714-CP-1-2006-1- NL-MPP).

References

Berlanga, A. and Garcia, F.J. (2003) 'Towards reusable adaptive rules', *Workshop on AH and Collaborative Web-based Systems*, ICWE'04.

Brusilovsky, P. (2001) 'Adaptive hypermedia', *User Modeling and User Adapted Interaction*, Vol. 11, Nos. 1–2, pp.87–110.

Carro, R.M., Moriyón, R., Pulido, E. and Rodríguez, P. (1999) 'Teaching tasks in an adaptive learning environment', in Bullinger, H. and Ziegler, J. (Eds.): *HCI Communication, Cooperation and Application Design*, Vol. 2, pp.740–744.

Cesarini, M., Monga, M. and Tedesco, R. (2004) 'Carrying on the e-learning process with a workflow management engine', *Proceedings of the 2004 ACM symposium on Applied Computing*, Nicosia, Cyprus, March 14–17.

Coffield, F., Moseley, D., Hall, E. and Ecclestone, K. (2004) *Learning Styles and Pedagody in Post-16 Learning: A Systematic and Critical Review*, http://www.lsda.org.uk/files/pdf/1543.pdf.

Cristea, A.I. and Calvi, L. (2004) 'The three layers of adaptation granularity', *Proc. of the User Modeling Conference UM2003*, pp.4–14.

Cristea, A. and Cristea, P. (2004) 'Evaluation of adaptive hypermedia authoring patterns during a Socrates programme class', *Journal of Advanced Technology for Learning*, ACTA Press, Vol. 1, No. 2, pp.115–124.

Cristea, A. and de Mooij, A. (2003) 'LAOS: layered WWW AHS authoring model and its corresponding algebraic operators', *Proceedings of WWW'03, Alternate Education Track*, Budapest, Hungary, ACM, 20–24 May.

Dagger, D. (2004) 'Developing adaptive pedagogy with the Adaptive Course Construction Toolkit (ACCT)', *2nd International Workshop on A3EH, AH'04*, http://wwwis.win.tue.nl/~acristea/AH04/workshopAH.htm.

De Bra, P., Aerts, A. and Rousseau, B. (2002) 'Concept relationship types for AHA! 2.0', *Proceedings of AACE ELearn'2002 Conference*, Montréal, Canada, pp.1386–1389.

de La Passardiere, B. and Dufresne, A. (1992) 'Adaptive navigational tools for educational hypermedia', *Computer Assisted Learning*, Springer-Verlag, Berlin, pp.555–567.

Felder, R.M. and Soloman, B.A. (2000) *Learning Styles and Strategies*, http://www.engr.ncsu.edu/learningstyles/ilsweb.html.

Holodnaya, M.A. (2002) *Cognitive Styles: about the Nature of Individual Mind* (in Russian) Per Se, Moscow.

IEEE-LOM (2002) 'IEEE-LOM 1484.12.1-2002.1', *Standard for Learning Object Metadata* http://ltsc.ieee.org/wg12/.

IMS Global Learning Consortium (2003) 'Simple sequencing protocol', *Version 1.0 Final Specification*, March, http://www.imsglobal.org/simplesequencing/index.cfm.

IMS-LD (IMS Global Learning Consortium) (2003) 'IMS learning design information model', *(IMS-LD), Version 1.0 Final Specification*, 20 January, http://www.imsproject.org/learningdesign/index.cfm.

Moore, A., Brailsford, T.J. and Stewart, C.D. (2001) 'Personally tailored teaching in WHURLE using conditional transclusion', Short Paper, *12th ACM Hypertext Conference*, Arhus, Denmark, August 14–18.

Riding, R.J. and Buckle, C.F. (1990) *Learning Styles and Training Performance*, Training Agency, Sheffield.

Stash, N., Cristea, A. and De Bra, P. (2004) 'Authoring of learning styles in adaptive hypermedia: problems and solutions', *Proceedings of WWW'04, Alternate Education Track*, ACM, New York, USA, 17–22 May.

Stash, N., Cristea, A. and De Bra, P. (2006) 'Learning styles adaptation language for adaptive hypermedia', *Proceedings of AH'2006 Conference*, Dublin, Ireland, pp.323–327.

Stern, M. and Woolf, P. (2000) 'Adaptive content in an online lecture system', *AH'00 Proceedings*, Trento, Italy, pp.291–300.

Riding, R.J. and Rayner, S. (1995) 'The information superhighway and individualized learning', *Educational Psychology*, Vol. 15, No. 4, pp.365–378.

Triantafillou, E., Pomportsis, A. and Georgiadou, E. (2002) 'AES-CS: adaptive educational system base on cognitive styles', *AH2002 Workshop*, Malaga, Spain, pp.10–20.

Vantroys, T. and Peter, Y. (2003) 'COW, a flexible platform for the enactment of learning scenarios', *CRIWG 2003, Springer*, LNCS 2806, pp.168–182.

Witkin, H.A., Moore, C.A., Goodenough, D.R. and Cox, P.W. (1977) 'Field-dependent and field-independent cognitive styles and their educational implications', *Review of Educational Research*, Vol. 47, No. 1, pp.1–64.

Wu, H.A. (2002) *Reference Architecture for Adaptive Hypermedia Applications*, Doctoral Thesis, TU/e, The Netherlands, ISBN 90-386-0572-2.

Websites

ADL, SCORM, http://www.adlnet.org/index.cfm?fuseaction=scormabt.

AHA!, http://aha.win.tue.nl.

Dublin Core Metadate Initiative, http://dublincore.org.

EML, http://eml.ou.nl/eml-ou-nl.htm.

W3C Semantic Web, http://www.w3.org/2001/sw/.

W3C.XML Protocol specification, http://www.w3.org/2000/xp/.

Notes

[1]Procedures are new language constructs extending the language.

[2]The idea behind it is simple yet semantically significant: it is based on computer games, where a player has to collect a number of items to advance between levels. This number may be fixed, but the choice of which items to select is up to him.

[3]PM stands for *Presentation Model* in LAOS.

[4]In AHA! there can be different types of concepts, e.g., *abstract*, *page* or *object* (fragment) concepts. Abstract concepts do not have a resource associated with it. Page concept can have one or more associated resources. Fragment concepts should be included into pages; they can have multiple resources, however they represent alternative versions of a *part* of a page. These resources are well-formed documents, to be scanned by the AHA! engine for other recursively included objects. Therefore they do not have a header and cannot be viewed separately.

[5]A pseudo-concept created when a user first logs into the system, storing user information such as name, login, password. As all concepts in AHA!, it can have arbitrary attributes. It can be used to specify attributes reflecting the learning style.

[6]The 'strange' escape sequences &amp; &gt; and &lt; in the XML file are needed because the XML parser will translate them to &, &gt; and &lt;. Without the escaping the XML parser would *interpret*, instead of *translating* them.

[7]Children of this concept have an attribute 'media'.