The next generation Authoring Adaptive Hypermedia: Using and Evaluating the MOT3.0 and PEAL tools

Jonathan G K Foss and Alexandra I Cristea
Department of Computer Science
University of Warwick
Coventry, CV4 7AL. UK
+4424 7657 3797

Email: {J.G.K.Foss, A.I.Cristea}@warwick.ac.uk

ABSTRACT

Adaptive hypermedia allows for customization to the needs of the user. The authoring process however is not trivial, and is often the main hurdle to overcome in order to bring this useful paradigm to a greater number of users. In this paper, we discuss the major problems occurring in authoring of adaptive hypermedia, and propose a set of generic authoring imperatives, to be consulted by any system implementing creation tools for customization of content. Based on these imperatives, in this paper we extensively illustrate and discuss recent extensions and improvements we have implemented in the My Online Teacher (MOT) adaptation authoring tool set, including the MOT3.0 content authoring and labeling tool and the PEAL adaptation strategy author. Furthermore, we evaluate, compare and discuss two long term uses of the MOT tool set, first in 2008 and the second in 2009.

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, user issues*; H.3.3 [**Data**]: Data Structures - *distributed data structures*, *graphs and networks*; K.3.1 [**Computers and Education**]: Computer Uses in Education - *distance learning.*

General Terms

Measurement, Design, Reliability, Experimentation, Human Factors, Standardization, Theory.

Keywords

adaptive hypermedia, authoring tools, LAOS, LAG, MOT.

1. INTRODUCTION

Adaptive educational hypermedia [1] can provide different information to learners based on various combinations of factors, such as their learning goals, preferences, background, explicit or implicit needs, learning place, tool used in learning (e.g., desktop computer, mobile device [2]), context, environment, network parameters, etc. (this list is not intended to be exhaustive). This is achieved by applying an adaptation strategy [3] to the learning material, and thus carefully (and automatically) selecting the content, information and type of interaction that the learner requires. To do this, the author (often, the teacher, but in some cases, a dedicated course designer) must first divide the content into standalone fragments, annotated with sufficient, semantically rich meta-data [4], in order to be able to be reused in various combinations. Then, the teacher would have to select the personalization strategy (adaptive pedagogical strategy [5] [6]) that best suits her students. To ensure the popularity of adaptive educational hypermedia, it is essential that content creation appeals to a wide variety of content authors, and so this process must be as simple and as straightforward as possible. My Online Teacher (MOT) [7] aims to simplify the authoring process, and encourage non-technical content authors to create complex and interesting courses, with enough adaptation and personalization so as to be useful, but without great authoring overhead.

The overarching goal of our research is thus to find the problems and bottlenecks in the authoring process of adaptive material, and propose, implement & test solutions to them. Major problems with authoring are:

- The *authoring process is very complex*; authors cannot always tackle all its facets. Thus, beside its novelty factor, such authoring is genuinely hard. Solutions to this are: divide the authoring process into many sub-tasks, which are individually simpler to perform; divide these sub-tasks to different roles (not all authors need then to author all the facets, just the ones appropriate to their expertise; e.g., content experts author content, pedagogical experts author pedagogical strategies, etc.). This leads to a separation of concerns, in its many aspects: modular view on the authoring process itself, different roles for authors, etc.
- The *authoring process for adaptive content is new and alien to most authors*. It is a given that most authors have never been exposed to authoring for adaptation. Whilst many authors have authored content, authoring for adaptation requires even for content a significant amount of supplementary information, which authors are not used to have to provide (such as meta-data which will be used in adaptation). To deal with these issues of novelty, solutions are to apply standards wherever possible, provide help and reuse of existing materials, whilst smoothing the learning curve.

Based on the overall goal and the current problems highlighted above, this paper transforms the solutions to these problems into a set of authoring imperatives. Based on these imperatives, this paper briefly revisits the main issues improved in MOT3.0 [8] content authoring and labeling tool, and in the PEAL [9] adaptation strategy authoring tool. Moreover, this paper also presents an in-depth evaluation. Unfortunately, in the past, too many systems, especially in the area of adaptive hypermedia, were built only till the proof-of-concept point, and real, long-term use was often considered superfluous. The paper thus also discusses both systems in the light of two long-term evaluation and usage case studies, performed in two consecutive years. This long-term real-life use is important to demonstrate that the gap between idea and practice has been bridged.

The remainder of the paper is organized as follows. In section 2, the MOT toolset is briefly sketched. Section 3 presents related research. The main authoring imperatives are defined in section 4. Sections 5 and 6 describe the new authoring tool set, evaluated in section 7. Section 8 outlines future research, and section 9 draws conclusions.

2. MY ONLINE TEACHER (MOT)

MOT (My Online Teacher) [10] is a set of authoring tools for adaptive (educational) hypermedia, which has been developed over the past 10 years in various minor and major reiterations. MOT follows the five-layer LAOS [11] authoring framework, which adheres to the 'separation of concerns' principle, especially with regard to separation of content and adaptation authoring. The content authoring part of MOT focuses on the first two layers of LAOS; the domain model and the goal model. The domain model allows content to be authored in a hierarchical structure of concepts. This type of structure is often encountered in educational applications, and thus is useful as the basis of adaptive educational content. Each concept contains a title and any number of other attributes, forming the different content alternatives for a concept. The goal model describes the learning outcomes of a particular lesson, and consists of a hierarchical structure of sublessons, which points to content in the domain model. In particular, each sublesson is a link to a specific attribute in the domain model. Sublessons can also be annotated with labels and weights. These labels and weights are used when applying adaptation to the content. MOT allows the author to export domain maps and goal maps to a CAF [7] file, which is an XML file format. The adaptation description (the personalized adaptive pedagogical strategy deciding how the content created in the content authoring part is to be used) is defined in the adaptation authoring part of MOT. Adaptation is specified in various ways, either edited in text editors, or via online tools, such as the MOT Adapt online tool[1], developed early on as part of the MOT tool set. The adaptation specification can then be exported as files written in the LAG [9] language. CAF and LAG files can then be imported into an adaptive (educational) hypermedia delivery system such as AHA! [12] or ADE [13].

3. RELATED RESEARCH

MOT didn't appear in a void. By the time it started being developed, adaptive hypermedia was an established principle. However, there were significant problems with the authoring tools. One problem was that they were not flexible enough, as in the case of Interbook [14], which allowed for easy authoring, but couldn't build much more than a prerequisite structure. Another problem with such authoring tools was that they were not easy enough for non-experts to use. AHA! [12] is an adaptive hypermedia system that provides a set of authoring tools to allow the author to create complex adaptation structures. However, the authoring tools (including a 'Concept Editor', a 'Graph Editor' and a 'Form Editor') are not trivial tools. In other adaptive systems developed, such as WHURLE [15], both the adaptation is limited (only adaptation to 3 types of learners possible), and the authoring is difficult (editing of XML files). The TANGOW [16] system presented some interesting, however highly specialized emotion-based adaptation. The WOTAN [17] tool showed promise in terms of graphical authoring, but suffers from a higher level of complexity, and also has issues with scalability.

4. AUTHORING IMPERATIVES

From various past researches and implementations, we have gradually refined a set of main authoring principles, as well as a set of recommendations for usability. This section describes the main principles of authoring, with respect to the problems outlined in section 1.

4.1 Complexity Imperatives

The following imperatives are designed to address the problem of complexity within adaptive hypermedia authoring systems.

1) *Separation of concerns* (to ensure reusability, and flexibility): Separate minimally adaptive content from static content; and, preferably, separate other elements of adaptation (such as user model or presentation model).

2) *Use of frameworks*: Frameworks usually already implement principle 1, the separation of concerns. Thus they are a straightforward way of respecting the principle. Frameworks also are a preliminary form of standards, as systems obeying the same framework can more easily build interfaces between them. Using frameworks (or models) thus ensures compatibility, some form of separation of concerns, as well as, a more rigorous methodology (apart from the early trial-and-error approaches to building adaptation). Frameworks, finally, are a straight-forward way of conveying lessons learnt from previous research.

3) *Use of standards*: Ideally, for proper reuse, standards should be used where possible. However, the adaptation process of adaptive hypermedia is not yet standardized. E-learning standards should be used where possible for adaptive systems.

4.2 Support Imperatives

Since authoring for adaptive hypermedia is new, and authors may be unfamiliar with the complexity of the authoring process, it is important that the system caters to the author's needs.

1) *Adaptive functionality*: The authoring system should adapt to the needs of the author, suggesting relevant features (this being independent of the adaptation to the learner).

2) *Simple access* to (content) pieces that need to be reassembled for different types of adaptation: The system should allow the author access to: Simple reordering; Simple labeling; Simple search; Simple copying & pasting; Simple linking; Simple editing (e.g., of content, strategies, etc.).

[1] http://e-learning.dsp.pub.ro/motadapt/adaptation.cgi

3) *Shallow learning curve*: The system should be easy to learn, and should allow beginner authors to use a wide variety of features in simple ways.

4) *Familiarity*:

The system should be familiar to the author, thus use as many conventions as possible (functionality, formats, etc.) which are familiar to the author. In particular, the system should provide:

a) *Consistency with existing applications and systems*: The functionality and representation should imitate, where possible, types familiar to the author (e.g., Microsoft Word, Internet Explorer and Mozilla Browsers, etc.).

b) *Interoperability with familiar course creation formats*: The system should be able to extract information from more traditional learning resources, such as presentation files. Where possible, the system should be able to automatically convert these files into adaptive material.

c) *Interoperability with familiar course creation standards*: The system should be compatible with standards used in many other (e-)learning systems, including SCORM[2], LOM, IMS-CP, IMS-QTI and IMS-LD[3].

d) *Interoperability with familiar course creation systems*: The system should be able to import (or export, possibly with some loss of adaptivity) to Learning Management Systems (LMS), which are a popular way of delivering and creating online course material at present.

5. MOT3.0

MOT3.0 was created as a radical overhaul of the content and labeling component of the MOT authoring toolset, to better respond to the above authoring imperatives, as well as to address functional and usability issues of its predecessor, MOT1.0.

5.1 Separation of concerns

MOT3.0 (similarly to MOT1.0) by itself is dedicated to authoring and labeling the content only. Adaptation strategies need to be edited via a different tool. Thus, from the start, MOT3.0 is obeying principles of separation of concerns. Moreover, it also distinguishes between domain map authoring, and goal map authoring. Domain maps allow the bundling of domain concepts into a hierarchical structure with attributes that describe the domain content. Goal maps allow adaptation parameter attributes to be specified. Here, these attributes are weights and labels, which, in an educational context, can be used as pedagogical labels. Also, the tool allows *separation of roles* of authors: an author working with MOT is a content author, and could be different from the adaptation author.

5.2 Use of frameworks

As with MOT1.0, MOT3.0 is based on the LAOS authoring framework. Using a framework is a first step to allow for reusability and compatibility with other systems, which may not need to know the exact internal workings of a tool. This ensures such tools can share the overall principles and models.

[2] http://www.scorm.com/

[3] http://www.imsglobal.org/

5.3 Use of standards

To ensure interoperability with other, more established, educational systems, MOT3.0, (unlike MOT1.0), is able to import content from SCORM[2], IMS-CP and IMS-QTI[3]. This allows content that has been authored in a Learning Management System (LMS) such as Moodle or Sakai to be reused within an adaptive course. Of course, this means that labels and metadata describing adaptation characteristics need to be added. This can be performed in the tools, after the import. Thus, whereas import is possible without information loss, export to such standards would lose the information about adaptation metadata, which is not expressible via the standards. Hence, MOT3.0 allows for enhancement of linear content with adaptation metadata.

5.4 Adaptive functionality

As with MOT1.0, MOT3.0 can suggest domain concepts that are related to another domain concept. However, MOT3.0 extends this functionality to allow these relatedness computations to be based on the similarity of keywords in the 'keyword' attribute (if existent) of two concepts. Additionally, the system is able to compute the similarity of concepts by comparing keywords with the content of other attributes (and thus checking for the appearance of these special keywords in the plain text or (X)HTML of any other attribute). MOT3.0 can also calculate similarities by comparing all the attributes of a given concept to all other attributes in the database.

To calculate the strength of a relation between two concepts, the following formula is used:

$$W_S = \frac{|K_S \cap K_T|}{|K_S|} \times 100\%$$

where K_S is the set of keywords in the *source* concept, K_T is the set of keywords in the *target* concept, and W_S is the weight to express the strength of the relation. If the system is comparing non-keyword attributes, the textual content of the attribute is divided into individual words, before the above formula is applied. This formula has distinct resemblance with the *recall* formula in Information Retrieval theory, where the source keywords would correspond to the relevant documents, and the target keywords would correspond to the retrieved documents, and thus is an appropriate way of retrieving related concepts. This formula could be extended to take into account multisets (multiple occurrences of the same keyword, which would be especially useful when searching through natural language text). The number of times a desired keyword appears could strengthen or weaken the overall relatedness between the two concepts.

5.5 Simple access

An adaptive (educational) hypermedia authoring system must emulate an adaptive delivery engine, in that it needs to adapt to the user – here, the author or teacher. It is therefore necessary that the interface can easily update content based on new information about the author's current purpose. Thus, to provide a smoother user interface, instead of a frames-based layout that was used in other adaptive systems (e.g. AHA! and MOT1.0), an Ajax interface was built. This allows data to be synchronized with the server without the need for frequent page refreshes.

Additionally, MOT3.0 adds a search facility, which is also implemented using Ajax. The search facility is employed in many

aspects of MOT3.0. Unlike in MOT1.0, authors can now search for a concept when copying and linking between domain maps, and when editing goal maps.

5.6 Familiarity

Both MOT1.0 and MOT3.0 aim for a shallow learning curve. Functions are kept elementary, even if functionality is complex. Where possible, MOT3.0 provides a development environment that is familiar to most users. An essential feature in authoring adaptive hypermedia is the ability to use content fragments, and reorder them in a simple and straightforward manner. MOT3.0 uses the JavaScript control jstree[4] to display domain map and goal map hierarchies. This allows the author to quickly rearrange a hierarchy in the browser, and then use Ajax to save the new structure to the server. Trees in MOT1.0 were generated as static HTML by server-side Perl scripts, and were much more cumbersome to maneuver.

Simple editing is essential for adaptive authoring systems. To make it easier for non-technical authors to use HTML within their courses, MOT3.0 adds the JavaScript based WYSIWYG[5] CKEditor[6], allowing authors to input text either in a rich text (X)HTML editor, or in source mode. This replaces MOT1.0's simple editing window, without any formatting, which only allowed for simple text or (X)HTML input.

Many authors will already have created educational resources in a non-adaptive format. It is important to encourage them to reuse these resources, rather than force them to recreate the content. MediaWiki[7] is a collaborative authoring tool, which allows authors to create and format articles using the WikiText markup language. WikiText allows articles to be separated into sections using subheadings. MOT3.0 is now able to automatically download a named article from any MediaWiki (e.g., Wikipedia[8]), and parse the article's WikiText. There, subheading levels are denoted using a number of '=' signs. For each subheading, a new concept is generated, with a text attribute which contains an HTML version of its content. From the level of the subheading, MOT3.0 is able to infer a concept hierarchy. This allows the automatic generation of a domain map and a goal map that resembles the structure of the MediaWiki article. This feature is new to MOT3.0, and was not available in MOT1.0.

Another type of linear resource to be exploited are presentations. Many educators are likely to have already created some (non-adaptive) educational material in the form of presentation slides. To allow authors to reuse this material, MOT3.0 introduces a new import script to allow static content from slides to be incorporated into an adaptive system. Presentation files, such as Microsoft PowerPoint files or OpenOffice.org Impress[9] files, can be uploaded to the MOT server. A script then automates OpenOffice.org to convert the file into a series of HTML and JPEG files. These files are then converted into a domain map. For each slide, a new concept is created, and each concept is allocated four attributes: title – extracted from the title of the slide; text – an HTML version of the textual content of the slide; image – a JPEG image of the slide; and notes – the presenter's notes.

Currently, the import script generates a domain map with a flat hierarchy, consisting of a root concept with a child concept for each slide. Future work will investigate ways of automatically inferring relationships between slides, and therefore creating more detailed domain map hierarchies. Again, this functionality is completely new in MOT3.0, and was not available in MOT1.0.

6. PEAL

In addition to the development of MOT3.0, a new environment has been build for creating adaptation strategies via the LAG language [7], called PEAL [9].

6.1 Separation of concerns

PEAL is designed to create adaptation strategies, and thus separates adaptation from content. Moreover, whilst MOT3.0 is to be used by the content author, PEAL is designed to be used by the adaptation specification author. This means that PEAL needs a more programming-savvy author. However, due to the separation of concerns facilitated by the LAOS framework, the adaptation programmer can be a different person from the content author. Although LAG strategies are not aimed to be written by non-technical authors, they can be reused by content authors without any such programming knowledge.

To allow interoperability with other tools, and thus follow the separation of concerns principle, the following two issues must also be addressed:

1) *Frameworks*: The PEAL tool is based on the LAOS framework, and moreover, on the LAG framework [18] for adaptation.
2) *Standards*: Since there are no standards for adaptation, PEAL uses the LAG adaptation language [7], and thus promotes reuse. This means, any system that can import from the LAG language can use PEAL as an adaptation strategy authoring tool.

6.2 Adaptive functionality

PEAL helps the authors to complete their work in various semi-automatic ways, as follows.

1) *Status bar suggestions:*

   PEAL continually monitors the validity of the code, and suggests missing pieces of code by displaying a message in the status bar.

2) *Code completion:*

   Whilst typing the program code, PEAL allows the user to select the correct statements, operators and variables from a list of suggestions (see Fig. 3).

3) *Strategy Wizard*:

   PEAL provides a strategy wizard that allows the author to define and initialize variables that will be used within the strategy.

---

[4] http://www.jstree.com/

[5] What You See Is What You Get (used for editors)

[6] http:// ckeditor.com/

[7] http://www.mediawiki.org/

[8] http://www.wikipedia.org/

[9] http://www.openoffice.org/product/impress.html

6.3 Simple access

To assist the adaptation language author in accessing edited adaptation strategies and snippets, PEAL provides the following features for online storage:

1. *Strategies* – strategies can be saved online, in either a *private* or *public* space. Public strategies can be seen and used by all authors. Private strategies are to be seen, edited and used by the current author only.
2. *Code fragments* – fragments of code can be saved for reuse within other strategies, thus effectively building a code library that extends the language. This library is available to all authors.

These features are intended to assist programmers in creating their strategies. Additionally, the public storage space, and the reusable fragments allow for collaboration and reuse between authors.

6.4 Shallow learning curve

PEAL is aimed at a different type of author than MOT1.0 and MOT3.0. An author in PEAL needs to be familiar with the process of programming. However, for an author with such background, PEAL does not require a great programmer. PEAL uses the LAG language, which contains a reduced number of programming instructions, no variable typing, and in general, all simplifications possible, in order to keep the learning curve shallow. Additionally, the PEAL system aims to be a familiar environment for authors, as described below.

6.5 Familiarity

In addition to the features enumerated above, PEAL also provides *Syntax highlighting* where LAG keywords are highlighted, to allow the author to quickly identify elements of code. This and the features listed above allow PEAL to simulate features that authors who have programmed before will recognize from programming environments. Thus, although they are writing adaptation strategies, which they may not be familiar with, the environment provides the familiar features via the background and support. Other such features are the various buttons used for saving, opening files, calling the wizard, declaring a strategy private or public, etc., which are based on commonly used icons.

7. EVALUATIONS

The MOT authoring toolset has been used in two consecutive years, 2008 and 2009, between October and December, as the set of systems on which part of the coursework for the course on "Dynamic Web-based Systems" is performed at the Computer Science Department of the University of Warwick, The course is taught to a mix of Computer Science MEng and MSc students. They learn about adaptive web-systems, and thus are required to create some such web-systems, via our set of tools, and effectively become authors of their own adaptive presentations. The content they should use is also related to adaptive and dynamic web-based systems. Thus they learn about the topic whilst applying it, both in terms of most appropriate material, as well as in terms of applying the right combination of meta-data and creating the right adaptation strategies for the personalized presentation of the material. The 2008 class was composed of 23 students, whereas in 2009 the number grew to 34. In-between the two runs of the course, the MOT authoring tool set was updated radically. In 2008, for the content creation and labeling, students worked with MOT1.0, and in 2009 with MOT3.0[10]. Also, for the adaptation strategy creation, students worked with the LAG language in both years, but in 2009, additionally, they had the support of the PEAL system. Students performed all their activities in groups, in both years. However, the size of groups was different: in 2008 there were 4-5 students per group, and in 2009 there were 2-3 students per group. The reduction of group size in 2009 was done based on the feedback and suggestions of the 2008 cohort, who felt that more intense collaboration is easier in smaller groups.

In-between the two long-term runs we present here, the MOT3.0 system (presented in section 5), had been briefly evaluated in March 2009, with two classes of students at the Politehnica University of Bucharest, who compared the system with its previous version. These evaluations were of a relatively short duration (a few days of interaction with MOT3.0 and MOT1.0, then completing questionnaires), and concentrated only on the content authoring. Importantly, these evaluations highlighted that students preferred (with confidence of 95%) reordering hierarchies for domain maps and goal maps and navigation via the menu in MOT3.0. They also showed preference for browsing domain maps and goal maps; creating new domain maps and goal maps; copying concepts and linking between concept maps; and using the HTML editor to edit attributes in MOT3.0. They also liked the structure of imported Wikipedia articles, and the idea that they could import them. However, this approval for the Wikipedia importer was not statistically significant.

There were, however, two features where there was a preference shown for MOT1.0. These features were adding/deleting concepts, and manually creating goal maps. After these evaluations, this feedback was taken into account, and the system was updated. One of the main changes replaced the Javascript component dhtmlxTree, with jstree, to make adding and deleting concepts easier, and to assist the author with rearranging goal maps.

Students in both years, 2008 and 2009, had to perform the following:

- Coursework 1.1: *Simple steps with the content authoring tool (MOT1.0 or MOT 3.0)*. The students were *taught how to use the tool via the following scenarios:*
  - *Exploring Domain Maps and Goal Maps* (Browsing, Views, Searching);
  - *Working with Domain Maps (creating,* editing, changing hierarchy, *coping concepts, linking* concepts, defining concept relations, deleting)*;*
  - *Working with Goal Maps* (converting domain maps to goal maps, adding sublessons, changing the hierarchy, assigning labels and weights, exporting goal maps as CAF files, deleting);
  - *Importing a Wikipedia page;* (MOT3.0 only)
  - *Importing a Presentation* (MOT3.0 only)

[10] MOT2.0 exists but has a different purpose, as it is related to Web 2.0 and social interactions, not extensively targeted in any of the other versions of MOT

- Coursework 1.2: Selecting one-two topics on which to create adaptive content for. The topics were to be related to the overall area of the course, and some suggestion list was given (including papers on this topic [19]). However, students were also encouraged to find their own topic of interest related to the module.
- Coursework 1.3: Complex labeling, usage and creation of strategies (in a text editor for the class of 2008, and the PEAL tool for the class of 2009), using the LAG language: Students were required to perform the complete process of creating static content, labeling it, and applying at least one strategy from the strategy pool. They then converted the output to an adaptive lesson, by uploading their content and strategies to AHA! [12].
- Coursework 1.4: integration of used techniques with multiple strategies and usable course content: Here students were asked to add at least two more strategies to their courses and at least one of these strategies should be entirely designed by themselves, using a text editor (2008) and PEAL (2009).
- Students then gave a (marked) presentation of their topics of choice, and they were encouraged to include their adaptive course in their presentations.
- Finally, at the end of the course, the students were asked to present a (marked) portfolio, consisting of four adaptive courses, created with: one or two content descriptions (domain maps) based on their chosen topics; three or four goal maps (here, appropriate labels, weights, etc. were important); and four adaptive strategies.

Other minor differences were that courseworks 1.1-1.4 were marked courseworks in 2008 and unmarked in 2009. Overall, however, the students had to perform similar quantities of work with the authoring systems, and present at the end about four adaptive courses. Whilst working on the coursework, students were encouraged to provide feedback on the suite of authoring systems that they were using. The feedback methods were direct feedback (during seminars) and a course forum with threads for each of the authoring tools.

7.1 Usage of MOT

7.1.1 Analysis of Created Content

Figure 1 shows an example of a domain map created by one of the groups in 2008 using MOT1.0. Domain maps such as the one in the figure described the content for a specific domain of choice (here, Privacy Enhanced Personalization). In 2008, each group created only one domain map, which they then used to apply different (up to four) adaptation strategies.

For comparison, Figure 2 shows an example of a domain map created by one of the groups in 2009 using MOT3.0.

Both implementations have the same basic functionality – in fact, the implementation from 2009 has some extra functions – however, as can be seen by comparing the figures, the implementation of 2009 is simpler in appearance, and thus distracts the author less from the task at hand.

In 2009, each group of students created multiple domain maps. In fact, some groups created up to four domain maps, to represent their two topics. Some groups created a primary domain map, containing the information about their topic, with a secondary domain map, that was used for defining settings. For instance, one group created a course about healthcare, with a primary domain map providing healthcare information (not shown here, as it is similar in structure with the example map from 2008, in Figure 1), and a secondary domain map containing a series of concepts to store questions (see Figure 2).

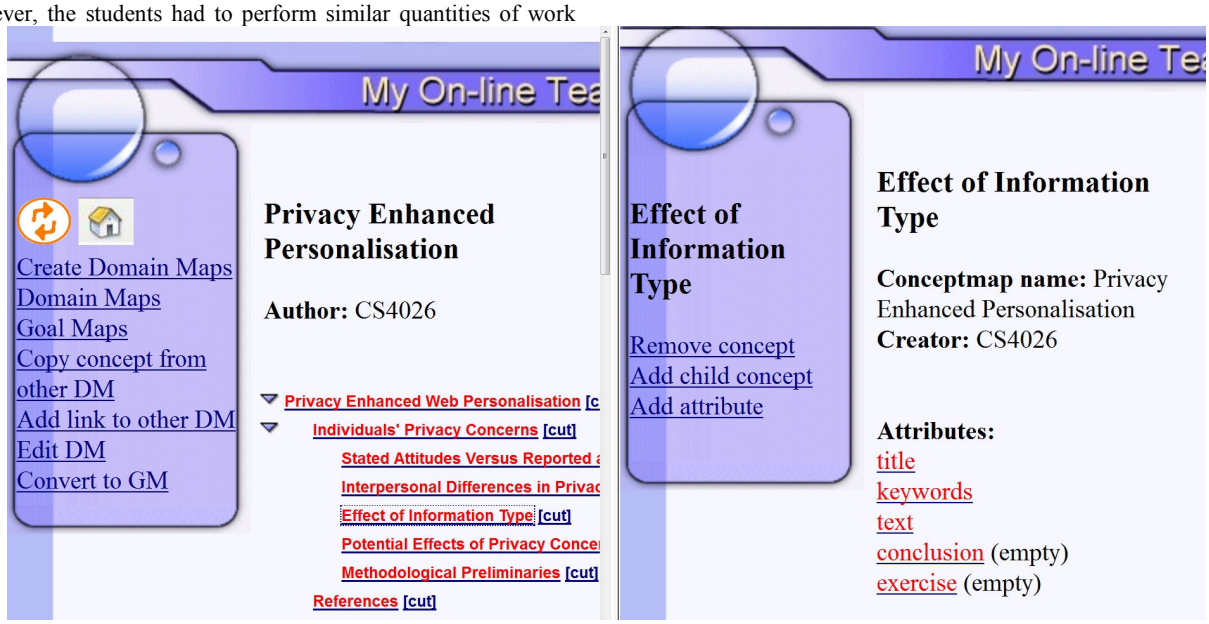

Figure 1. A domain map hierarchy in MOT1.0

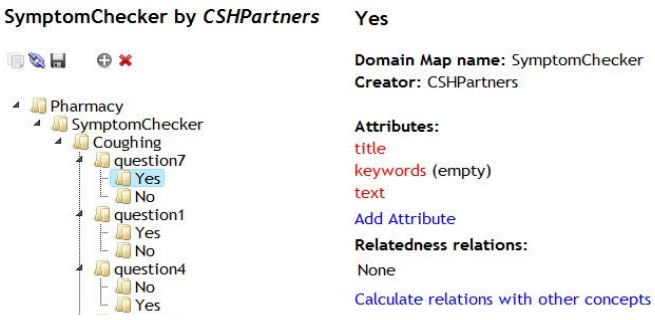

Figure 2. A domain map hierarchy in MOT3.0

Each question contains a 'Yes' or a 'No' concept, with medical advice stored in the text attribute of the answer concept. This was a rather ingenious way of dealing with different types of adaptation. This shows that, although the students had the same amount of time to perform their work, the students in 2009, possibly due to the simpler and clearer environment, managed to create more complex solutions for the same problem.

7.1.2 Quantitative Feedback

Here we analyze, from a quantitative point of view, the differences in terms of use of MOT1.0 versus MOT3.0 by the two groups of students in 2008 and 2009. This means we analyze the *domain maps* and *goal maps* students have created, and the features of these maps. The students in 2009 submitted in total 29 *domain maps*, whilst the group in 2008 submitted only 4. However, this difference is also due to the fact that the 2009 class was larger, that their groups in which they were divided were of smaller size, and that their assignment changed in that it asked for more domain maps (thus resulting in an average of 3-4 per group). So a direct comparison of the quantity of the output is not appropriate here. However, the ultimate goal of the students was the same, which is, to create about 4 adaptive presentations. The students in 2008 opted for doing this by creating one domain map per group, and then reusing it with about 4 strategies. However, the students in 2009 did this via a greater number of domain maps. For instance, some 2009 students created courses that used one main domain map, with several smaller maps (e.g. to store 'settings' information, as explained previously). This therefore suggests that creating domain maps became easier with the new toolset.

To overcome bias introduced by these smaller domain maps, the 'Best Domain Map per group' row in Table 1 is calculated for MOT3.0, by only considering the largest domain map created by each group. Similarly, the 'All Domain Maps per group' is calculated by combining all the domain maps created by a group, and evaluating average characteristics for each group. For MOT1.0, as the students only created one domain map per group, the values per 'All DMs', 'Best DM per group' and 'All DMs per group' are identical, and thus not repeated.

Table 1. Created domain map averages

DM (Domain Map) Production	Average Number of Concepts	(Average) Attributes Number per Concept	Average Depth	Average Attributes Number per DM	Average Custom Attributes Number	Average Distinct Custom Attributes Number	Percentage of attributes containing HTML
MOT1.0: For All DMs	*22*	*2.86*	*2.86*	*62.75*	*1*	*1*	*25.09*
MOT3.0: For All DMs	22.24	**2.45**	3.23	**54.59**	7.03	1.86	34.68
MOT3.0: For Best DM per group	27.1	**2.4**	3.3	65	6.3	1.5	32
MOT3.0: For All DMs per group	65.6	**2.43**	3.17	158.3	20.4	5.4	34.68

The table shows that the domain map hierarchies created were reasonably large, with each domain map containing a remarkably close average of about 22 *concepts* (slightly larger in 2009, when applying MOT3.0, but not significantly so). Interestingly, this is similar to the number independently aimed at in [20], but larger than the findings in [21], who obtained an average of 16.9. However, the production in terms of *average number of concepts (for all DMs)* has slightly increased from MOT1.0 to MOT3.0, especially with respect to the best domain maps per group. Also, the production of concepts *for all DMs per group* grew (from 22 in 2008, to 65.6 in 2009). However, as the table shows, the *average attributes number per concept* was slightly lower in 2009, for all categories, when compared to the 2.86 in 2008. As we have seen, some groups created auxiliary domain maps, with information stored in a single attribute for each concept, which influences the average number per concepts per domain map. However, also the best DM per group presents a lower number of attributes per concept (2.4 in 2009). One added functionality was that of importing content from Wikipedia, which generates a lot of content, however produces only a low number of attributes per concept (2: text and title). Even if students have added additional attributes (which they have) the size of the maps may have been too large to add a significant number of supplementary attributes. To alleviate these difficulties, in the future the MOT3.0 tool will allow for automatic addition of user defined attributes to the standard attribute set. This can serve to have a consistent structure, but it does not add content. For the latter, we can employ automatic tools for content enrichment, as proposed in [22].

Moreover, when we compare the *average* (non-empty) *attributes number per DM,* instead of *per concept*, we see that, in general, an increase is visible (if we compare the *best DM per group*, or *all DMs per group*, instead of the simple average). This shows that the information content in the lessons designed by the students has increased at the level of the groups, thus supporting the statement that, in the same amount of time, it was possible for students to produce more material with MOT3.0 than with MOT1.0.

As the table also illustrates, the *average depth* of a domain map in 2009 was 3.23, thus slightly larger than the maps created in 2008

(2.86), showing that the domain maps created using MOT3.0 were more structured.

There is also an increase in the *average custom attributes number*, and the *average distinct custom attributes number*. This shows that students have added their own definitions of attributes, and reused these definitions within their domain maps. This increase in custom attributes is statistically significant, for the confidence interval of 95%, with P=0.018 and T=2.5, as established by a two-sample T-test.

The *percentage of attributes containing HTML* content has risen from 25% in 2008 to around 35% in 2009. This suggests that the WYSIWYG CKEditor has encouraged authors to use HTML within their attributes.

A similar analysis was applied to the *goal maps* created by the two classes of 2008 and 2009. For the comparison of goal map production, again, averages were calculated, instead of computing the overall productivity, due to the difference in student numbers. Table 2 shows some selected results of the analysis.

The percentage of labeled and weighted sublessons is much larger. In particular, the difference between the *percentage (of) weighted* sublessons used in 2009 (of 30.49) and that used in 2008 (of 10) is statistically significant for the confidence interval of 95%, with p=0.04 (based on a T-test). Similarly, the difference between the *percentage (of) labeled* sublessons used in 2009 (of 34.27) and that used in 2008 (of 9.62) is also statistically significant (as per T-test with 95% confidence interval, p=0.00). This clearly points to the fact that adding weights and labels has become much easier in MOT3.0, as opposed to MOT1.0. This is probably due to the fact that multiple weights and labels can be set at the same time in MOT3.0, whereas his had to be done one at a time in MOT1.0,

However, the *average number of sublessons* for each goal map is smaller in MOT3.0, compared to MOT1.0. This difference could be explained by the slight reduction in the number of attributes per concept in 2009, as sublessons follow often the attribute per concept structure.

Table 2. Created goal map averages

GM (Goal Map) Production	Average Number of Sublessons	Percentage Labeled	Percentage Weighted
MOT1.0: All GMs	*123*	*6.74*	*7.01*
MOT3 0; All GMs	**81.09**	41.73	37.13
MOT1.0: Best GM per group	*114.2*	*18,91*	*18.56*
MOT3.0: Best GM per group	**103.1**	50.92	58.29
MOT1.0: All GMs per group	*315*	*9.13*	*9.05*
MOT3.0: All GMs per group	315.4	42.04	37.35

7.1.3 Qualitative Feedback

The students provided a lot of feedback during their 3 months exposure to the tools. A spiral model for software development [23] was used during this long-term usage, reiterating development, evaluations, and new objectives. This allowed functionality to be developed and tested quickly.

Whilst many users appreciated the idea of a drag & drop tree structure, they raised issues about the implementation. It was suggested that the tree should automatically save its structure whilst the user re-orders it (one currently needs to click a 'save' button to confirm the changes). There were also some (minor) incompatibilities identified between MOT3.0 and AHA!, especially regarding HTML entities, that were exported by MOT3.0, but not displayed by AHA!. Some users also stated that the Wikipedia importer system should be extended to allow images to be imported, too. Overall, from all discussions, we can say that the students understood the ideas behind authoring content.

7.2 Usage of PEAL

7.2.1 Analysis of Created Content

Overall, the students in 2008 implemented only very slightly changed strategies based on the pool of strategies we provided for them. In 2009, students were, to some extent, more innovative. Examples of such original strategies from the 2009 students include strategies based on the classic beginner–intermediate–advanced strategy, which contains 3 levels of knowledge, and only shows concepts according to the user's current level. Students extended this to a more general strategy catering to any number (≥1) of knowledge levels, or a strategy where 75% knowledge is enough to progress to the next level. Another example is an extension to the visual-verbal strategy, with an extra textual preference, while verbal is interpreted as a preference for spoken text.

Another example of innovative thinking is illustrated by a number of strategies that were created around the notion of a learning goal. For instance, a strategy represented the learning goal as the sum of knowledge levels of visited concepts, and decreased knowledge for revisiting concepts. Another group created a 'Mixed Revision' strategy, designed to help students revise. In this strategy, text attributes are initially hidden, but keyword attributes are shown. If the user revisits a particular concept (suggesting that s/he is unfamiliar with the concept), the text is shown for further revision. Another interesting strategy was one for device adaptation. The description noted that while the students implemented it as an adaptable strategy (i.e., user-driven), with settings that allowed the user to select the current device, they commented that they would have liked to be able to get this information (automatically) from the AHA! delivery system. This suggests that options for device adaptation would be perceived as a useful extension to the delivery engine.

Figure 3 shows a screenshot of PEAL, whilst editing a LAG strategy, as created by one of the student groups of 2009.

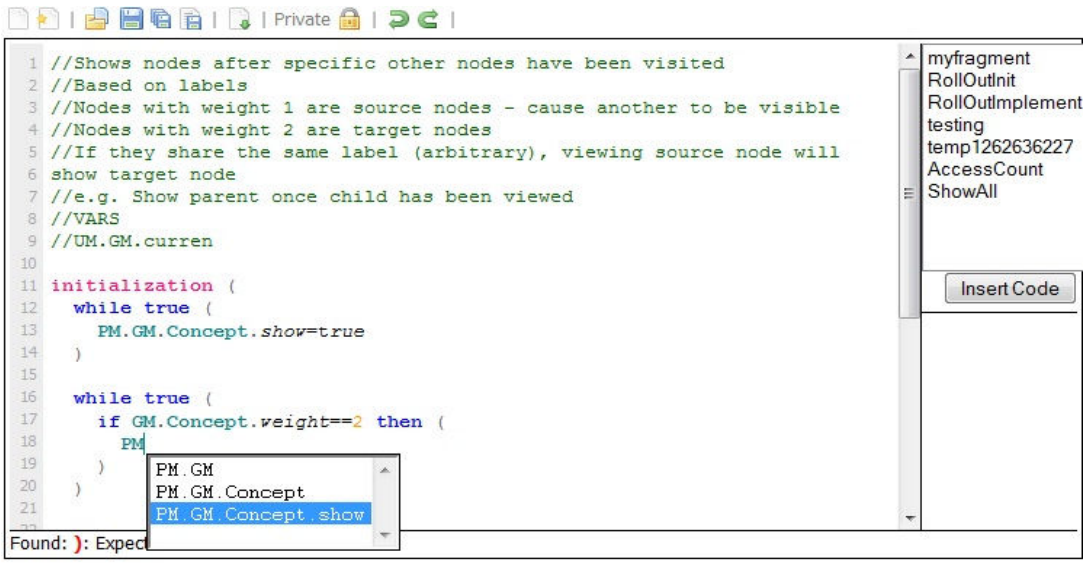

Figure 3. Editing a LAG Strategy using PEAL

7.2.2 Quantitative Feedback

It is also possible to analyze the adaptation rules submitted by the students, from a quantitative point of view. In 2008, the students used a text editor of their choice, whereas in 2009 they were allowed the use of PEAL. Table 3 shows that the strategies created by the 2009 students were significantly (P=0.00 T=3.13) longer than those created by the 2008 cohort. There is also a very large standard deviation for the strategies created in 2009 (STDEV= 36.4) with one strategy containing 199 lines of code. The average size of the strategy description actually reduced slightly, but the variance was high (e.g., strategies with no description were also submitted). Beside the fact that students were not marked for including these comments, the code may have seemed self-explanatory to computer science students. In the follow up version of PEAL, the description will be separate from the main program, and will become an obligatory field, as lay person authors need this information in order to know if to apply a strategy or not.

Table 3. Created Strategies

Adaptation strategy editing tool:	**Average number of lines of code**	**Average size of strategy description**
Text editor: 2008	30.67	7.83
PEAL: 2009	64.18	7.56

7.2.3 Qualitative Feedback

The template strategy pool[11] we put at the disposal of the students in both 2008 and 2009 was considered useful to base strategies on. Many students opted for tweaking existing strategies to their particular scenario. Especially popular were slight modifications on the depth-first-search and breadth-first-search strategies, to which students added conditions to show certain labeled concepts before their natural order.

The overall quality of strategies created was clearly higher than in previous years. While other factors, such as the quality of students and the evolving training material may have all contributed, it seems reasonable to say that the PEAL editor is responsible for at least part of this improvement.

However, some issues were identified with the PEAL tool in its current state. One student requested that PEAL should provide a line number when warning about invalid code, since the current method leads to confusion if a script contains more than one problem. There were also comments about the lengthy procedure required to import the CAF and LAG files into AHA!.

8. FUTURE WORK

Using students in their last years of study to evaluate authoring tools is helpful because much of their work consists of gathering and presenting material, in a similar way to teachers. Clearly, such authors also need to be (and have been) involved in the evaluation process. Still, for comprehensive, long-term evaluations with iterative development, student evaluations are useful, and have highlighted a number of usability issues that needed to be addressed.

Future research will investigate in particular other ways of extracting content from non-adaptive file formats. The MediaWiki importer script will also be extended, to allow keywords and images to be imported. Another way to extend the importer would be to crawl between articles in the wiki. The linked article could then be included in the original domain map. This would allow more comprehensive domain maps to be created, however the depth of this import would need to be researched.

PEAL will also be improved to accommodate the suggestions from the evaluations, including extending the accuracy of the warning messages, and streamlining the process of previewing changes. This could be achieved by integrating MOT3.0 and PEAL with the ADE delivery engine. Moreover, work is on its way to create a purely visual version of PEAL, with drag and drop code snippets, similar, to some extent, to the simple authoring techniques employed in MOT3.0.

[11] http://prolearn.dcs.warwick.ac.uk/strategies.html

9. CONCLUSION

This paper defines a set of *adaptation authoring imperatives*, extracted from previous experience in design, implementation and deployment of authoring for adaptation, as well as on related research. Based on this, the paper then contrasts the main features of *two generations of authoring tools for adaptation*, as well as compares their *real life deployment*. This long-term comparative deployment of the new toolset versus the previous one has shown improvement in productivity and quality of created material with the new set, thus providing more evidence towards the fact that the overall, the MOT3.0 and PEAL approach is promising. Additionally, such an intensive, long term use has provided useful feedback about the functionality (and stability) required for an adaptive hypermedia authoring system, which will be further exploited in the future. A major lesson learnt is that it is important that such tools are released beyond the proof of concept, as fully developed software to encourage the widespread use of adaptive hypermedia.

10. ACKNOWLEDGEMENTS

This research is partially supported by the GRAPPLE IST project IST-2007-215434.

11. REFERENCES

[1] Brusilovsky, P. 2004. Adaptive Educational Hypermedia: From generation to generation (Invited talk). In Proc.of 4th Hellenic Conference on Information and Communication Technologies in Education (Athens, Greece 2004), 19-33.

[2] Kravcik, M., Kaibel, A., Specht, M., and Terrenghi, L. 2004. Mobile Collector for Field Trips. Educational Technology & Society, 7, 2, 25-33.

[3] Henze, N. and Nejdl, W. 2002. Knowledge Modeling for Open Adaptive Hypermedia. In 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems, (May, Malaga, Spain, 2002).29-31.

[4] Nilsson, M., Palemér, M., and Naeve, A. 2002. Semantic Web Metadata for e-Learning – Some Architectual Guidelines. In Proceedings of the 11th World Wide Web Conference (WWW2002) (Hawaii, USA, 2002).

[5] Dagger, D., Wade, V., and Conlan, O. 2004. Adaptive Pedagogy with the Adaptive Course Construction Toolkit (ACCT). In A3EH: 3rd International Workshop, AH 2004 (Eindhoven, The Netherlands, 2004).

[6] Melis, E., Andrès, E., Büdenbender, J., Frishauf, A., Goguadse, G., Libbrecht, P., Pollet, M., and Ullrich, C. 2001. ActiveMath: A web-based learning environment. International Journal of Artificial Intelligence in Education, 12, 4 (2001), 385-407.

[7] Cristea, A. I., Smits, D., and De Bra, P. 2007. Towards a generic adaptive hypermedia platform: a conversion case study. Journal of Digital Information (JoDI), 3, 8 (2007).

[8] Foss, J. G. K. and Cristea, A. I. 2010. Transforming a linear module into an adaptive one: tackling the challenge. In Intelligent Tutoring Systems, ITS 2010. (Pittsburgh, USA, 2010).

[9] Cristea, A. I. Smits, D., Bevan, J., and Hendrix, M. 2009. LAG 2.0: Refining a reusable Adaptation Language and Improving on its Authoring. In 4th ECTEL International Conference (Nice, France, 2009), Springer LCNS, 7-21.

[10] Ghali, F., and Cristea, A. I., 2009. MOT2.0: A Case Study on the Usefulness of Social Modeling for Personalized E-Learning Systems. In The 14th Int. Conference of Artificial Intelligence in Education (AIED'09) (2009), IOS Press.

[11] Cristea, A., and de Mooij, A., 2003. LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators. In The 12th Int. WWW'03 Conference. Alternate Track on Education (Budapest, Hungary, 2003).

[12] De Bra, P., Smits, D., and Stash N. 2006. The Design of AHA! In Proceedings of the ACM Conference on Hypertext and Hypermedia (Odense, Denmark, 2006). 133-134

[13] Scotton, J., Moebs, S., McManis, J., and Cristea, A. I. 2009. A Case Study on Merging Strategies for Authoring QoE-based Adaptive Hypermedia. In A3H: 7th Int. Workshop, 4th ECTEL International Conference (Nice, France, 2009).

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

[15] Brailsford, T., Moore, A., Stewart, C., Zakaria, M. R., Choo, B. S., and Davies, P. 2001. Towards a Framework for Effective Web-based Distributed Learning. In Poster, 10th International WWW Conference (Hong Kong, 2001).

[16] Carro, R., Pulido, E., and Rodriguez, P. 1999. Task-based Adaptive learNer Guidance on the WWW. In Computer Science Report (Eindhoven, The Netherlands 1999), 49-57.

[17] Freire, M., and Rogriguez, P. 2006. A Graph-Based Monitoring Tool for Adaptive Hypermedia Course Systems. In Proceedings of the International Conference on Adaptive Hypermedia (AH) (Dublin, Ireland, 2006), 279-282.

[18] Cristea, A. I., and Calvi, L. 2003 The three Layers of Adaptation Granularity. In Int. Conf. on User Modelling (UM'03) Springer, (Pittsburgh, US, 2003) 145-155.

[19] Brusilovsky, P., Kobsa, A., and Nejdl, W. 2007. The Adaptive Web: Methods and Strategies of Web Personalization. Springer, (May 16 2007).

[20] Rice, D. C., Ryan, J. M., Samson S. M. 1998. Using Concept Maps to Assess Student Learning in the Science Classroom: Must Different Methods Compete? Journal of Research in Science Teaching, Vol. 35, No. 10, 1103–1127 (1998).

[21] Temple, B., and Marshall H. 1996. Using Concept Maps to Evaluate Teaching and Learning. Original ultiBASE STAR Report, (June 1996).

[22] Hendrix, M. and Cristea, A. I. 2008. A spiral model for adding automatic adaptive authoring to adaptive hypermedia. Journal of Universal Computer Science (JUCS). Special Issue on Authoring of Adaptive and Adaptable Hypermedia, Springer, Vol. 14, No. 17, (September 2008).

[23] Boehm, B. A Spiral Model of Software Development and Enhancement (May 1986) ACM SIIGSOFT Software Engineeering Notes, 11(4), 14-24.