Reusing Adaptation Strategies in Adaptive Educational Hypermedia Systems

Joshua Scotton, Alexandra I. Cristea
Department of Computer Science
The University of Warwick
Coventry CV4 7AL, UK
{jscotton, acristea}@dcs.warwick.ac.uk

***Abstract*—Due to the difficulty and thus effort and expenses involved in creating them, personalization strategies in learning environments have to demonstrate a higher return-on-investment (ROI), if they are to be a viable component of the learning setting of the future. One feature that can increase this ROI is the reusability of adaptation strategies in Adaptive Educational Hypermedia Systems. This research looks into various ways of enhancing this reusability. Using multiple *modular adaptation strategies* (MAS) with a controlling *meta-strategy* is proposed as a more efficient way of authoring adaptation strategies. This renders the reuse of adaptation strategies faster and easier for course authors. A method for semi-automatically breaking down complex adaptation strategies into smaller modular adaptation strategies is described. Potential problems with using multiple strategies are described and ways to solve them are discussed. Finally, some evaluation points are illustrated, conclusions are drawn and further research areas are identified.**

Keywords - LAG; Adaptive Educational Hypermedia; Technology Enhanced Learning; Meta-strategies; Adaptation; Reusability; Adaptation Strategies

I. INTRODUCTION

A limiting factor with current Adaptive Educational Hypermedia (AEH) systems is the *reusability* of the adaptation strategies applied within the systems. Often, the adaptation strategies involved are very specific to the course for which they were written, even though they describe adaptation behaviors which are applicable to multiple courses. Course creators may often lack the time or the skills needed to create new adaptation strategies from scratch, and therefore any improvement in the reusability of adaptation strategies is a major help in the authoring process of AEH courses. Writing functional adaptation strategies is not trivial, and we don't expect every teacher or educator to be able to master it. In previous research, we have advocated the separation of concerns principle [1] [2], which states, amongst others, that *adaptive behavior of a course* and the *content of a course* should be able to be authored separately. Besides the obvious implications of reuse, this separation also permits the two parts to be authored by different roles, i.e., by authors of different experience. Whilst subject knowledge is essential when authoring the course content (as performed by the *content author*), for authoring personalized adaptation strategies, a combination between knowledge of pedagogy and some elementary programming knowledge is important. The latter is done by the *adaptation author*, which is our primary target in this paper.

Therefore, this paper tackles the *challenge of reusable personalization strategies for learning* in AEH systems using the LAOS authoring framework [3] and the LAG adaptation strategy language [4] for illustration, and will describe ways in which the authoring of AEH courses can be made easier by improving the reusability of adaptation strategies. The paper then describes potential problems with reusing adaptation strategies and how these may be avoided. Finally points for evaluation are described and areas for further research are identified.

II. A CASE STUDY

The following scenario illustrates the need for reuse of adaptation strategies in AEH systems:

Professor X creates a new course on Computer Science for international students and wants to adapt the content to:

- Change the content depending on network conditions so that those with slow internet connections will not experience delays in accessing the course materials. This would be called a Quality of Service strategy; and
- Slowly present additional information to students, during their progress in the course. At every revisit, content is added for each concept they are learning, but other content parts are removed. This would be called a RollOut Strategy.

Professor X then writes a large adaptation strategy which succesfully implements those adaptation behaviours and runs the course successfully.

Sometime later, Professor Y creates a course on Modern Art and wants to reuse Professor X's adaptation strategy, but only the RollOut adaptation behaviour. Not being from a Computer Science background, Professor Y finds it very difficult to extract the code relating to that adaptation strategy and wishes that the process of authoring and reusing adaptation strategies was made simpler somehow.

III. RELATED WORK

The creation of AEH courses can be a time-consuming process, and multiple methods have been proposed to reduce the amount of time and effort needed to create such courses.

As well as developing *more effective authoring tools* (MOT [5], GAT [6], NetCoach [7], AHA! Graph Author [8] etc.) to enhance the course creation process, research has

focused on improving the reusability of the content and adaptation specification of adaptive courses.

This has involved developing *multiple adaptation frameworks* including ACE [9], AHAM [10], LAOS [3] and the GRAPPLE framework[1].

Reuse of adaptation specifications can also increase the efficiency of authoring AEH. However, the opportunities for reuse rely on the type of adaptation representation within an AEH system [4]. The reuse of adaptation specifications becomes simpler, as the *adaptation representation becomes more generalised and abstracted from the content domain* [2], due to the fact that less modification of the adaptation specification is needed to apply it to a new content domain.

'*Assembly-level*' adaptation languages [4] such as those used in AHA! [11], Interbook [12] and WHURLE [13] are at a disadvantage in this respect, as the adaptation specification is closely linked to the content domain. Thus it cannot be separated from it, nor reused.

Higher-level adaptation languages [4], such as LAG and LAG-XLS [14], where the adaptation specification can be completely generic, are in a much better position to be reused.

However, even though whole adaptation strategies using these adaptation languages can be reutilized across multiple content domains, *reusing parts of strategies* and *combining adaptation strategies* is still problematic.

IV. Reuse of Adaptation Strategies

The case study highlights a common problem with reusing adaptation strategies, where only part of the original strategy is needed. This is particularly a problem for course authors who lack the skills or time to write a new adaptation strategy, and rely on reusing strategies created by others.

Our proposed solution is for strategy authors to create smaller, *modular adaptation strategies* (MAS) which can be combined in different permutations using *meta-strategies* within an AEH system, to acheive the desired adaptation behaviours.

A. Modular Adaptation Strategies and Meta-Strategies

Modular adaptation strategies are strategies that provide specific adaptation behaviors, which can then be used as building blocks in the overall pedagogical strategy for an AEH course. This overall course adaptation strategy would be described by a *meta-strategy*, which specifies when and in which order the modular adaptation strategies are applied to the course content.

An author wishing to reuse these modular strategies would only need to create a new meta-strategy, instead of having to potentially rewrite the whole adaptation strategy.

If modular adaptation strategies are used, then it would be possible to create a meta-strategy authoring tool which provides a list of modular and whole adaptation strategies, from which the author selects the relevant (modular) adaptation strategies that he or she wishes to use in the AEH course.

[1] www.grapple-project.org/

B. Creating modular strategies from existing strategies

However, modular adaptation strategies and meta-strategies don't solve the problem of reusing parts from pre-existing adaptation strategies which haven't been created in this way.

In order to be able to reuse existing strategies, they would first need to be 'broken down' into smaller modular adaptation strategies. As adaptation strategies describe adaptation behaviours, it is logical to identify those and then create modular strategies for each.

Ideally this would be an automated process, however research into automatic identification of adaptation behaviours is still ongoing and hence, to begin with, this would have to be a semi-automated process.

The proposed method for this consists of the following steps:

1) Manually add *semantic markup* to the original adaptation, to label and describe the adaptation behaviour.
2) Use this semantic markup to aid software in automatically creating the reusable *modular strategies* from the original strategy.
3) Automatically create the meta-strategies controlling the created meta-strategies. This creation process is based on the current author's needs and goals.

V. Manual Semantic Markup

In order to aid software in recognizing different personalization behaviors being described in an adaptation strategy, semantic markup can be added, to simplify this process. As shown in the code below, the markup is limited to describing the adaptation task being carried out by a section of code using the syntax:

```
<task name="Task Name"
      description="Task Description">
   Adaptation Code
</task>
```

The semantics of the markup is to make a piece of code reusable for other personalization strategies or meta-strategies. Next time when a strategy is authored, this piece of annotated code will be directly available, as we shall show in the following examples.

This markup process can be undertaken using a standard text editor, or could be carried out using new strategy authoring tools, or by extending existing tools, such as the PEAL [4] strategy authoring tool.

A. Markup in a text editor

We first describe the editing directly in a text editor, which requires a higher level of programming knowledge for the adaptation author. The proposed syntax can be demonstrated by a simple example using the following code from the RollOut adaptation strategy [15].

RollOut adaptation strategy: This strategy slowly rolls out (and hides) concept fragments, based on how often a concept has been accessed. Fragments have the label "*showatmost*" if they should disappear after a while (with a weight indicating the number of visits required till

disappearance) and the label "*showafter*" if they should show up after another number of visits (again, the weight indicates the number of visits).

We use for illustration the LAG language [4], although the annotation mechanism can be used for any other adaptation language as well. The LAG language uses two main interaction paradigms: (1) the description of the concepts and fragments that should be visible to a student the first time he visits a course (the *initialization*), and (2) the description of the adaptive interaction between student and system, which is run in a continuous loop, as long as the student is learning (the *implementation*) .

The code snippet below shows how markup can be added to existent LAG code (tags describing tasks are added to the original code). This can be done by Professor X for his strategy in the case study:

```
initialization(
while true (
  <task name="AccessCount" description="Set a counter
  for each concept, to count accesses to it">
    UM.GM.Concept.beenthere = 0
  </task>
  <task name="ShowAll" description="Show all
concepts">
    PM.GM.Concept.show = true
  </task>
)
  <task name="RemoveShowAfter" description="Remove
  in the initialization concept fragments with label showafter">
   while GM.Concept.label == showafter (
    if GM.Concept.weight > 1 then (
      PM.GM.Concept.show = false
    ) else (
      PM.GM.Concept.show = true
    )
  )
  </task>
)
implementation (
  <task name="AccessCount" description="Count Accesses to
concept">
  if UM.GM.Concept.access == true then (
    UM.GM.Concept.beenthere += 1
  )
  </task>
  <task name="ShowAfterShowAtMost" description="Remove
  concepts with label showatmost for which the Accesses to the concept
  are above the weight of that concept, and show concepts with label
  showafter for which the Accesses to the concept are above the weight of
  that concept">
   if enough(
      UM.GM.Concept.beenthere >= GM.Concept.weight
      GM.Concept.label == showatmost, 2) then (
    PM.GM.Concept.show = false )
   if enough(
      UM.GM.Concept.beenthere >= GM.Concept.weight
      GM.Concept.label == showafter, 2) then (
    PM.GM.Concept.show = true
  )
  </task>
  <task name="NetworkState" description="Show concepts
  which are appropriate for the current Network state.">
   if    (UM.GM.networkState == GM.Concept.label)
   then (PM.GM.Concept.show = true)
  </task>
)
```

The markup process has divided the code into five different reusable tasks, three in the initialization part of the code and two in the implementation part, each with their own *name*, and with *description* information. As can be seen in the following, the description information can be used later on, when reusing that particular code fragment. Please note that in our example, all code has been marked, but it is possible that an author only decides to reuse part and not all of his adaptation code (thus marking only a part of it).

B. *Markup in the PEAL adaptation strategy authoring system*

Alternatively, Professor X could use PEAL for the task markup. The PEAL authoring system can already store pieces of code for further reuse, as illustrated in Fig. 1.

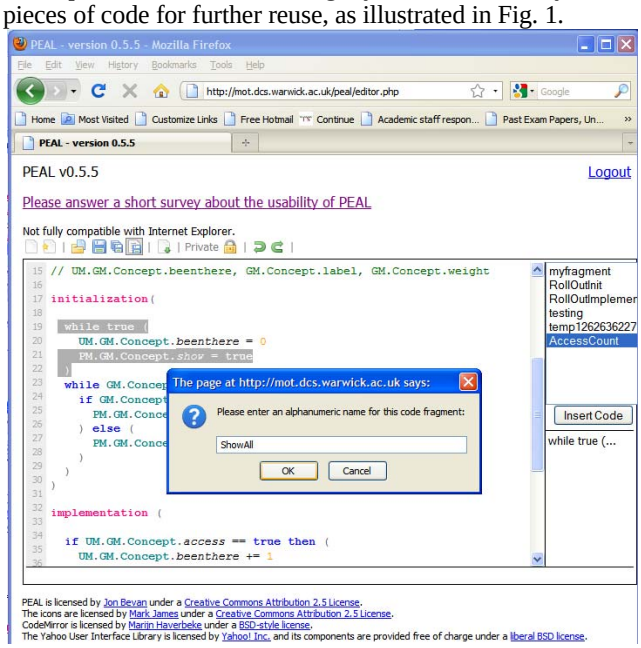

Figure 1. Saving a code fragment called 'ShowAll' in PEAL

As PEAL helps the author with colored syntax highlighting and hints, it is aimed at adaptation authors with less programming experience (importantly, however, *not* non-programmers). As the figure shows, PEAL has already saved the task 'AccessCount' (available in the right upper frame, and with a code preview in the lower frame), and the author is just saving the task 'ShowAll', by simply selecting the desired part of the code and saving it as a code fragment. Thus, an author can select from all existing fragments of code stored by himself, or by his colleague. Please note that this is additional to being able to reuse whole strategies saved in PEAL by any of his colleagues that used the common storage space (PEAL uses two types of spaces: *private* and *common*; private is available to the author only, and common is available to all). PEAL currently does not add, however, the description information, which makes it less usable (less readable) by non-programmers (as an author needs to read and understand the code preview in order to decide upon including a code fragment or not).

VI. Automatizing Modular Strategy Creation

The RollOut strategy shown above could now be automatically split up into modular adaptation strategies (MAS). A new MAS is created for each adaptation task described, as well as a default MAS, for any unmarked code.

During the process, the position of each marked block of code, and the conditions under which it can be executed, need to be added. This can be either done automatically by the system, for instance, by adding from a standard block of conditions, or by copying the conditions from the surrounding original code. Alternatively, this can be manually added by the author. An example of automatic system deduction is shown for the *'ShowAll'* task, which is located in the *initialization* section of the LAG strategy, inside a *while* block with a condition of 'true'. Hence an automatic MAS to be created by the system for this task is as follows:

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

One possible problem with fully automating it in this way is how far do we go upwards in deducing the higher-level conditions? Consider the following strategy code fragment:

```
if (PM.GM.Concept.show == true) then (
  if (GM.Concept.label == 'beg') then (
    <task name="UpdateBegCount">
    UM.begCount += 1
    </task>
  )
)
```

Do we include both, one or no conditions from the two if statements in the MAS? All are, arguably, equally useful. We would recommend that any tool automating the creation of MAS should warn the author if different options are available. For simplifying the process for beginner authors, a default option needs to be proposed.

VII. .Automatizing Meta-Strategies

Once the modular adaptation strategies (MAS) have been created, they can be used in meta-strategies, running the MAS as pieces of regular code. In particular, if a strategy has been completely divided into a number of modular adaptation strategies, an equivalent meta-strategy, representing the original adaptation strategy, can be automatically generated.

The proposed *meta-strategy LAG code* is similar to that of a normal LAG strategy and indeed can use any standard LAG constructs (such as 'if' and 'while' loops). The command to invoke a Modular Adaptation Strategy inside a Meta-strategy is as follows:

strategy [MAS name] *[Code block to execute from MAS]*

The execution order of the MAS would be determined by the order of the markup tags from the original strategy. A meta-strategy has two top-level code blocks (*initialization* and *implementation*) just like a standard LAG strategy, and the order of execution of the modular strategies can be different in each. Also, a MAS strategy does not necessarily need to appear in both code blocks. For instance, the *ShowAll* MAS has an empty implementation block and hence will only be used inside the meta-strategy initialization block.

The meta-strategy equivalent to the *RollOut* adaptation strategy example can therefore be extracted from the overall strategy, for Professor Y, as:

```
initialization (
  strategy AccessCount initialization
  strategy ShowAll initialization
  strategy RemoveShowAfter initialization
)
implementation (
  strategy AccessCount implementation
  strategy ShowAfterShowAtMost implementation
)
```

VIII. Problems with Reusing Multiple Adaptation Strategies

Although using multiple strategies can increase the reusability of the strategy, it can introduce new problems. Whilst these problems could occur when multiple strategies are authored concurrently for the same course, they most commonly occur when strategies are reused from different sources.

Displaying the desired adaptation behaviors when used in isolation does not guarantee that multiple adaptation strategies will not produce unforeseen behaviors when used together. The following examples illustrate the type of problems that can occur when using multiple adaptation strategies:

a) **Execution Order**: Some combinations of adaptation strategies will work correctly in one particular order but not when the order of execution is reversed or changed.
   An example of this can be taken from the case study: if the Multimedia Mix strategy is run first and displays a video, then the Quality of Service strategy may decide to hide the video if the network connection is poor. This would lead to a blank page being shown to the student which is an undesirable result. A solution to this is to have a pre-checking stage in the adaptation meta-strategy creation, with some potential (arbitrary) content. If no content is visible, than the strategy should roll back a step, and show the previous content. This can be inbuilt in the strategy creation, or, alternatively, in the delivery system.

b) **Variable Clashes**: If multiple adaptation strategies read and/or write to the same variable, then this could result in incorrect consequences.
   For example consider two strategies that both have the following line in the strategy file:

   ```
   UM.GM.Concept.beenthere += 1
   ```

   An AEH system using both strategies may report that the user has accessed the concept six times when the user has actually visited the concept only three times. This is incorrect and may impact other

areas of the course. A solution to this is to use a *parameterized MAS*, which declare what variables they are using. In this way, a system can automatically detect potential clashes.

c) **Type Conflicts**: Multiple strategies use the same variable to store different types of value. For example, consider one strategy which stores a Boolean using:

```
UM.GM.Concept.accessed = True
```

Another strategy would have a problem as it will expect an Integer when it accesses the same variable with the following code:

```
if (UM.GM.Concept.accessed > 2) (...
```

It is possible to highlight some potential problems from those described above at the strategy authoring stage. For example, the Type Conflict problem could be easily spotted by analyzing the variables within the strategies being used (or with a *parameterized MAS*, as previously proposed). The author could also be warned about some forms of variable clashes at this stage as well.

IX. Evaluation

Future evaluations of the ideas from this paper would need to address the following questions:

- Is it easier to reuse content from modular strategies than from single, complex strategies?
- Does using meta-strategies and modular strategies provide a less error prone way to author strategies than using a single, complex strategy?
- Is the authoring process faster when reusing content from modular adaptation strategies than from a single strategy?
- Can the method be expanded to include tools supporting the authoring of strategies?

X. Conclusion and Further Research

The case study illustrates the need for an easier way to reuse adaptation strategies in the authoring process of AEH courses. A number of ideas and methods have been discussed in this paper which will simplify and speed up this process. We have discussed possible issues with MAS combinations, such as variable clashes and type conflicts, and have given some suggestions on how to deal with these issues.

An interesting research area is that of how these problems can be automatically identified and avoided, either in the AEH systems or during the authoring process.

While modular adaptation strategies and meta-strategies simplify the reuse of adaptation strategies, it would be useful, additionally, to research authoring tools that simplify the creation of meta-strategies for authors without prior programming experience. For example, an authoring tool where authors drag and drop from a pool of *modular adaptation strategies*, to create the *meta-strategy* could potentially make it easier for such authors.

Furthermore, research is needed in the *semantic markup of adaptation behaviours* within adaptation strategies, including the extension and evaluation of authoring tools that allow for such semantic markup to be added. After this is completed research will be needed into automating this process, so that adaptation strategies can be broken down into modular adaptation strategies automatically.

References

[1] H. Wu, G. J. Houben, and P. De Bra, "Supporting User Adaptation in Adaptive Hypermedia Applications," in *Proceedings InfWet2000*, Rotterdam, The Netherlands, 2000.

[2] A. I. Cristea, "What can the Semantic Web do for Adaptive Educational Hypermedia?," *Educational Technology & Society*, vol. 7, no. 4, 2004.

[3] A. I. Cristea and A. de Mooij, "LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators," in *Proceedings of the WWW 2003 Conference*, 2003.

[4] A. I. Cristea, D. Smits, J. Bevan, and M. Hendrix, "LAG 2.0: Refining a reusable Adaptation Language and Improving on its Authoring," in *Proceedings of Fourth European Conference on Technology Enhanced Learning ECTEL 2009*, Nice, France, 2009.

[5] A. I. Cristea and A. De Mooij, "Adaptive Course Authoring: MOT, My Online Teacher," in *Proceedings of ICT-2003, IEEE LTTF International Conference on Telecommunications, "Telecommunications + Education" Workshop*, Tahiti Island in Papetee - French Polynesia, 2003.

[6] E. L. M. Ploum, "Authoring of Adaptation in the GRAPPLE Project," TECHNISCHE UNIVERSITEIT EINDHOVEN, 2009.

[7] G. Weber, H. Kuhl, and S. Weibelzahl, "Developing Adaptive Internet Based Courses with the Authoring System NetCoach," in *Hypermedia: Openness, Structural Awareness, and Adaptivity*. Springer Berlin / Heidelberg, 2002, pp. 222-223.

[8] P. De Bra, N. Stash, and B. De Lange, "AHA! Adding adaptive behavior to websites," in *Proc. of the NLUUG2003 Conference*, Ede, The Netherlands, 2003.

[9] M. Specht and R. Oppermann, "ACE - Adaptive Courseware Environment," *The New Review of Hypermedia and Multimedia*, vol. 4, 1998.

[10] P. De Bra, G. J. Houben, and H. Wu, "AHAM: A Dexter-based Reference Model for Adaptive Hypermedia," in *Proceedings of the ACM Conference on Hypertext and Hypermedia*, 1999, pp. 147-156.

[11] P. De Bra, D. Smits, and N. Stash, "The Design of AHA!," in *Proceedings of the ACM Hypertext Conference*, Odense, Denmark, 2006.

[12] J. Eklund and P. Brusilovsky, "InterBook: An Adaptive Tutoring System," *UniServe Science News*, vol. 12, no. March 1999.

[13] A. Moore, C. D. Stewart, M. R. Zakaria, and T. J. Brailsford, "WHURLE - an adaptive remote learning framework," in *International Conference on Engineering Education (ICEE-2003)*, Valencia, Spain, 2003.

[14] N. Stash, "Learning Styles Adaptation Language for Adaptive Hypermedia," in *Proceedings of AH'2006 Conference*, Dublin, Ireland, 2006.

[15] Adaptation Strategies. [Online]. http://prolearn.dcs.warwick.ac.uk/strategies.html