APLIC Report

This progress report describes work performed within the APLIC project during the time period 1st of May 2011 – 30th of June 2012. It involves the development, extension and usage with Warwick students and beyond of three systems: ADE, an adaptive course delivery environment and MOT, an authoring environment for the former tool (and other tools), as well as MOT2.0, an adaptive social e-learning tool. It also covers exploratory research into social media and adaptive advertising integration into adaptive educational hypermedia courses. The expenses for the period covered in this report include the travel of Joshua Scotton and Dana Al Qudah to Romania for the second experimental stage as well as work undertaken by Jonathan Foss, Joshua Scotton, Dana Al Qudah and Shi Lei.

The Adaptive Display Environment

The *Adaptive Display Environment* (ADE) is an adaptation delivery engine using the LAOS framework for authoring & delivery of adaptive hypermedia (AH). It builds on existing delivery engines by extending the adaptation behaviours that can be used in AH systems as well as increasing the reusability of adaptation specifications and content. A more detailed overview of ADE was included in the APLIC Report covering the period between 1st November 2010 and 30th April 2011.

At the start of the project ADE was a proof-of-concept prototype and work was needed to increase the functionality, speed and UI of the engine to bring it up to the level where it could be used experimentally in the classroom. After comparing favourably with existing systems in early 2011, work was carried out to extend the functionality of the system based on feedback from experiments carried out between November 2010 and April 2011, in particular the feedback from Warwick University's CS411 course that ran in the Autumn term of the 2010/11 academic year. The work focused on extensions to the LAG adaptation strategy language, which is used as the adaption language in ADE. These extensions were then supported by additional implementation work in ADE. For more information on the extensions, please see Appendix C.

My Online Teacher 4 (MOT4)

MOT4 is a tool which is designed to allow authors to create personalized online courses, and is a major redesign of MOT3.1. Amongst the new features, MOT4 allows authors to arrange concepts in a graph-based structure – allowing for more complex domain relationships to be expressed (rather than a tree structure, which only allows for parent relationships). This feature was added based on feedback in a previous evaluation of both MOT3.1 and GAT (see Appendix A). Another new feature is the ability to provide more than one pedagogical label for each content element, thus providing more flexibility for the author. For more details about MOT3.1 – please see the APLIC Report for 1/11/2010-30/4/2011.

MOT2.0

MOT2.0 was first developed by Fawaz Ghali at Warwick and is a personalised social tool for e-learning. For each learning item, learners are able to tag, rate or comment on the article. They are also able to send chat messages to their peers. Additionally, the system can ask learners to take a test, and recommend content according to the questions that the learner got wrong. Additionally, the system can recommend that the learner chats with other learners – according to the speciality of the users.

In the past academic year, MOT2.0 was used with two groups of students (and 'CS411: Dynamic Web-Based Systems'). The first module, 'CS252: Fundamentals of Relational Databases' used MOT2.0 as a revision tool over the Christmas break in preparation for a January exam. The second module, 'CS411: Dynamic Web-based Systems' used MOT2.0 to teach a particular topic over two lectures.

Students were randomly allocated into one of five groups. Each group provided a different set of the social features, to evaluate the learning value of each social feature. In some groups, this included limiting certain features to the more knowledgeable students.

Within each module, few students fully engaged with the learning tool. This means that the results were not as significant as those documented in Fawaz Ghali's thesis. Part of the reason for this might be that social learning has become more widespread since Fawaz's original experiment [1]. Therefore, rather than being reliant on an integrated social learning tool, the students are already communicating and sharing content on other platforms such as Facebook and Twitter. Indeed, verbal feedback indicated that a "Computer Science - Warwick - 2010 to 2013/14" Facebook group was widely used for such social learning/revision. Further research will investigate how to integrate these existing social networks into adaptive learning environments.

GAT (GRAPPLE Authoring Tool)

GAT is a set of authoring tools, which are brought together transparently under one umbrella, and which have been created by the Warwick team, in collaboration with an Italian and Austrian team, during the FP7 project GRAPPLE. It is aimed at visualising, in graphical form, the process of authoring for adaptive personalised courses. It has introduced some interesting new paradigms, such as pedagogical relations, pedagogical sockets, a different low-level adaptation language (GALE), to name but a few. Some of the researchers involved in the creation of the tools supported partially by IATL were also involved in the GRAPPLE project and worked at the GAT toolset. GAT is designed to work with a different adaptation delivery engine, called GALE.

CS411: Dynamic Web-Based Systems

ADE was as used an example of adaptation delivery engines and also as part of the coursework for the CS411 module on Dynamic Web-Based Systems at the University of Warwick, in the 2011-12 winter term (term 2).

This module is a key part of the newly introduced MSc in Cognitive Systems, a joint degree between the Departments of Computer Science and Psychology. The module is also open (as an option) to students from the MSc in Computers and Applications, and the MEng degree at the Computer Science department. 17 students took this module (the decreased number compared to the previous year is due to the smaller overall cohort of MSc students in the two postgraduate taught courses, and it seems due to the fact that the module was given in the second, instead of the first term – hence most students had made their choices already).

[1] 2009, "MOT 2.0: A Case Study on the Usefulness of Social Modeling for Personalized E-Learning Systems ", The 14th International Conference of Artificial Intelligence in Education (AIED'09), IOS Press, June 2009, ISBN: 978-1-60750-028-5. Authors: F. Ghali, A. I. Cristea

Demonstration

As part of the course, ADE was used as the main adaptive delivery engine. Various examples of adaptive courses on ADE were shown, demonstrating many different types of adaptation, similarly to the previous year. However, the types of adaptation possible had increased from the previous year, including more powerful selection capabilities, social adaptation and domain filtering. For a more detailed explanation see Appendix C.

Coursework

Additionally to learning with the help of the ADE tool, as part of the CS411 Dynamic Web-based Systems module, the students learn about personalization – specifically using 'Adaptive Hypermedia'. For the coursework, the students were divided into groups, and asked to choose two relevant topics. They were then asked to create 4 adaptive courses (2 about each topic). This involves the students using MOT4 to author Domain and Goal content about their chosen subjects (based on some given or self-selected conference/journal papers). For each of their courses they were also asked to write an adaptation strategy (in the LAG adaptation language), which describes how/when the content should be shown. They were then asked to upload their content into ADE, so that the course could be viewed (and tested).

Feedback

The coursework was given to the students in week 2 of term 2, and was due in week 2 of term 3. During this period, the students continually provided feedback about the usability and functionality of both MOT4 and ADE. The content that was submitted as coursework is currently being statistically analysed to ascertain the utility of the tools. This content will be compared to the work submitted by previous cohorts, who used previous versions of the tools. This comparison will therefore allow various hypotheses about the authoring process to be evaluated – specifically, demonstrating whether recent changes to the authoring tools have made the authoring process easier. Some initial results are presented in Appendix C. Many of the previous issues raised in the 2010/11 CS411 course relating to ADE and the LAG language did not reappear in the feedback gathered in this period and the strategies that were submitted used functionality that has been added since November 2010. This suggests that the modifications to ADE and the LAG language as a result of the experiments and feedback from this project have been successful so far. A breakdown of the frequency of the constructs used in the strategies is shown in Appendix C.

In late March 2012, a similar evaluation was performed at the University Politehnica of Bucharest, Romania. This involved a class of 31 4th year students who were studying a module about the 'Semantic Web'. The students were given a number of sample adaptation strategies, and were asked to create content in MOT4 about various XML technologies (including XSLT and XQuery). These semantic web topics have distinct parallels with labelling content for adaptation, and the students were therefore asked to provide semantic content labels to their created content, so that their content could be displayed using one of the sample adaptation strategies. One of the main purposes of this evaluation was to demonstrate that content creators can author content according to the rules of a pre-existing strategy. The students were also asked to complete a questionnaire about the usability of MOT4.

A preliminary set of results based on analysis of the created content can be found in Appendix B. Results show that a similar number of concepts per Domain were created by the 2011-2012 cohort, showing that there is no statistical evidence that MOT4 changes the number of concepts that students are likely to create.

One of the main new features of MOT4 is the ability to visualize more complex (non-parent) relationships between concepts. However, for the UK evaluation, nobody used any such relationships. This might be because the support for these non-parent relationships within LAG was not clearly defined. Another reason might be that the number of students participating in the course was significantly fewer than in previous years.

For the Romanian evaluation, one of the sample LAG strategies made use of such relationships (by recommending related content). The analysis of the created content (Appendix B) shows that the Romanian students created more concepts (however this is not statistically significant), but it also shows that 5 groups created 7 or more extra (non-parent) relationships. This might suggest that the graph structure does not discourage users from creating concepts, but actually allows them a greater freedom to express how the concepts are related to each other.

With regard to the labelling of concepts, only one UK group made use of the new 'multiple labelling' feature, whereas in the previous year, three groups had requested that the multiple labelling feature be added. Indeed none of the 2012 UK groups used the new output format, LAF, which was created to support multiple labelling and a wider variety of relationships. Again, a possible explanation for this is that there were fewer groups and therefore a smaller variety of strategies.

MOT4 and GAT

Whilst the main coursework was carried out using MOT4, towards the end of term 2, the CS411 students were asked to create a basic course using GAT (the GRAPPLE Authoring Toolset), which provides a different way of allowing authors to specify adaptation. They were then asked for their opinions about the functionality of GAT compared with MOT4. This provided both quantitative (Likert-scale responses) and qualitative (comments) data about each system.

Due to the small number of students registered for the CS411 module and the fact that the questionnaire was optional, there were only 8 complete responses. The results provided an indication that users prefer the GAT style of building strategies using smaller 'building blocks' (75%) than the process of writing an entire strategy as in MOT, strengthening findings from the previous year (see Appendix A).

Each of these evaluations will be used to identify the most popular features of each system. The results will be published in a PhD thesis and will also be used in a journal/conference paper. The results and conclusions from these evaluations will allow the next generation of authoring tools to be developed – either at Warwick or by the wider Adaptive Hypermedia research community.

Web Application Development at Bucharest University

Work on ADE during 2011 focussed on targeting issues raised in the feedback from the 2010/11 CS411 students. Most of this work had been completed before the experiments conducted at Bucharest University in March 2011 and during the latter part of 2011 an identical set of experiments were conducted in April 2012 with a new version of ADE

The evaluation of ADE was undertaken by 25 students from the Web Application Development course at the Engineering faculty within the Politechnica University of Bucharest, Romania.

Three revision courses, delivered via ADE, were run alongside the tutorials in the course. The courses covered MySQL, Perl and PHP.

Alongside the experiments detailed above, two further research experiments were conducted with students from Bucharest University studying the Semantic Web and Web Application Development courses. These are detailed below.

Social Networking based Adaptive Educational Hypermedia

A participatory design methodology, We!Design was used to help design a social networking based AEH system prototype. The purpose of this was to propose a new system to replace and improve MOT2.0. However, this particular methodology allows for generation of fresh ideas, unencumbered by previous predefined ideas. Six fourth year undergraduate students studying the course of semantic web took part, with Dana Al Qudah conducting the experiment created by Shi Lei and guided by Jonathan Foss in the UK.

The first phase of the experiment was comprised of two design sessions, each of which lasted for approximately 2.5 hours. During the three stages of design session, the students went through the steps of *needs collecting*, *task sequencing* and *prototype designing*, based on their computer science knowledge background and a brief introduction to AEH and SNS (social networking systems).

By the end of the first phase, a low-tech prototype and a requirement list were proposed. In the second phase, the designers synthesized all the proposed requirements into a single application, and categorized and refined these requirements into an ordered list according to the priority to be developed.

Finally, all the students participating in the design sessions were asked to answer a questionnaire, which contained the questions relating to:

1) Their attitudes on existing e-learning systems that they were using or had used before;

2) Their expectation for the features of new SNS-based AEH systems.

The results of this experiment have been reported in a peer reviewed paper.

Adaptive Online Advertising

The aim of the final experiment was to suggest a framework for an adaptive online advertising system. The experiment was conducted with the help of 12 computer science students in their 3rd studying the Web Application Development course.

The session was conducted in three stages over two hours. The first stage was to examine the current knowledge and perceptions of students of e-commerce and online advertising. This was achieved through the distribution of questionnaires where 34 questions were answered. Most of the questions were closed ended questions.

The second stage was a theoretical background seminar about e-commerce, e-advertising and adaptive hypermedia. In the seminar, the different business models on the web and how these models are interrelated and their application in adaptive hypermedia and social networks were discussed.

The final stage was to make the student design their own system that they feel comfortable with which satisfies their needs as end users.

The results are being processed and a paper is under development at the time of this report.

Further Work

This project has allowed us to extend, improve and use academic prototypes for adaptive delivery, as well as authoring tools for adaptive e-learning, on a much wider scale than would have been possible otherwise.

Concretely, future research will continue to refine the authoring and delivery tools for adaptive hypermedia, with particular emphasis on their integration with social networking features. Similarly, adaptive e-business scenarios will also be created, particularly looking at how advertising can be adaptively presented based on social networking profiles.

The systems and work done are planned to be used further at Warwick and beyond, and to inform further research in the area.

List of Terms

ADE	*The Adaptive Display Environment (a delivery engine for adaptive hypermedia developed by Joshua Scotton)*
AEH	*Adaptive Educational Hypermedia*
CAF	*Common Adaptation Format*
GALE	*The GRAPPLE Adaptive Learning Environment*
GAT	*Grapple Authoring Tool*
GRAPPLE	Generic Responsive Adaptive Personalized Learning Environment
LAG	*An adaptive strategy language*
LAOS	*A framework for adaptive hypermedia systems*
MOT	*My Online Teacher (an authoring tool for adaptive content developed by researchers at Warwick University)*
MOT2.0	*This is a personalised social tool for e-learning*
MOT4	*The current version of the MOT authoring toolset*
SNS	*Social Networking Systems*

Appendix A

HyperText 2011 Conference Poster Abstract – see attached

Appendix B

MOT4.0 Analysis – see attached

Appendix C

Summary of extensions to LAG and analysis of use by CS411 Students.

LAG Grammar Updates

The basic structure of a LAG strategy is comprised of two sections, the initialization block and the implementation block, as shown in the example below. Comments are designated by a double forward slash.

```
initialization {
  //code goes here
}

implementation {
  //more code goes here
}
```

The initialization block is run when a user first uses a content domain, the implementation block is executed once per link click action. Code is executed in line order within each block.

Content within the system is accessed through Concept objects, for example `GM.Concepts` refers to all the concepts in the Goal Model layer. Presentation information about concepts are stored in the presentation model layer and can be modified through the `PM.GM.Concepts` construct. For example to display all concepts in all areas of the user interface you could use this: `PM.GM.Concepts.show = true`.

This not only determines whether a given concept should be displayed in the main content portion of a page, but also whether the current concept as well as the parent of the concept should be displayed in the navigational menus. Open-ended interviews and informal feedback during two long term authoring sessions (of three months each, from October to December 2009 and 2010, with two different groups of students studying the Dynamic Web-based Systems course at the University of Warwick, Computer Science department) has shown that such parallel, symmetrical behaviour, whilst easy to understand, can be sometimes undesirable.

To remedy this problem, a new extension to the LAG language is the ability to select the area to display a concept. The following example shows how you can show a concept when it's link is visited, but only display that link in the tree menu and not the todo list area.

```
PM.CONTENT.GM.Concepts.show = true
PM.MENU.GM.Concepts.show = true
PM.TODO.GM.Concepts.show = false
```

If a specific area is not specified, then the resulting behaviour will be that the show variable is set for all areas (CONTENT, MENU, TODO and NEXT). This is to keep backwards-compatibility with previous LAG strategies.

for-each

The singular instance of this can also be used but needs to be used inside a `for-each` loop. This can be used to filter through all concepts in the domain and allow them to be accessed through the singular `GM.Concept` construct. An example below, displays all non-visible concepts in the navigational tree menu.

```
for-each(PM.GM.Concept.show == false) {
  PM.MENU.GM.Concept.show = true
}
```

The for-each block was added recently to combat an inefficiency in the previous implementation of the LAG language. Prior to this, the implementation loop was run for each concept in the domain, which led to very slow processing times when large content domains were used. The for-each loop allows far more efficient selection of domain concepts within the strategy.

Filter Selection

Previously, there has been no quick way in LAG to select, display or modify a group of objects at the same time. For example, to display all the introductions (identified via the Introduction attribute) in the initialization of a course, the previous version of LAG would require the following code:

```
while ( GM.Concept.type == "Introduction") {
  PM.GM.Concept.show = true
}
```

This would loop through all the concepts and check if the type was Introduction before setting the show variables to true.

Here we propose using a shorter and more efficient way to select a group of concepts. We will call them lists as we assume that they are all ordered. To create a list we use a conditional filter to make a selection from a group of similar objects. This filter is based on the predicate element in the XPath syntax. The main idea in using XPath is to use existing standards where possible, to the extent they can be used. We cannot currently express all the types of web adaptation required in Web standard languages (such as XPath, XQuery, OWL, etc.). This ensures both compatibility with other systems, as well as familiarity of use for authors. The syntax for selecting a list is as follows:

```
{PM|UM}{DM|GM}{Concepts}[condition]
```

Using this syntax, to select all the introduction concepts we can then use:

```
  GM.Concepts[type=="Introduction"]
```

Here, LAG understands this to be the group of concepts where `GM.Concept.type = "introduction"`, type inheriting the models from the GM.Concepts. We can then undertake an action on the list, such as displaying all of them:

```
  PM.GM.Concepts[GM.type=="Introduction"].show = true
```

In this example, type needs to be qualified with `GM.` as `PM.GM.Concept.type` does not exist by default. LAG understands this to be set `PM.GM.Concept.show = true` for all concepts where `GM.Concept.type = "introduction"`

Social

With the advent of Web2.0, users have come to expect more social interaction in the systems that they use. We explain in this sub-section how we can implement certain social adaptation behaviours in LAG.

Allowing an adaptation strategy to access other user models introduces a whole variety of different adaptation behaviours. For example, we could select the experts on the current topic:

```
  GM.User[GM.Concept.knowledge==100]
```

Alternatively, we can display extra information to the current user on a topic that more than five users (which may or may not include the current user) are finding difficult (have a knowledge rating of less than 50

```
if GM.User[GM.Concept.knowledge<50].list.size > 5 then {
  PM.GM.Concept.Parent.extraExplanation.show = true
}
```

GM.User can also be used to adapt user models that do not belong to the current user. In the following example, the extra explanation is displayed to all users, if more than five users are finding the current concept difficult:

```
g1 AS GM.User[GM.Concept.knowledge<50].list
if g1.size > 5 then {
  g1.PM.GM.Concept.Parent.extraExplanation.show = true
}
```

User Interface Adaptation

Delivery systems for adaptive educational hypermedia will normally display a content area, navigational menus, course header and system links (such as logout, main menu etc.).

In some systems the layout and style of the interface can be changed on a per system or per course layout (for example, using CSS, in the case of AHA!. However, beyond showing/hiding certain sections of the layout, further adaptation of the layout at runtime is not available in current adaptive web-based systems.

In order for the layout to be dynamically adapted to the user's needs, layout sections need to be explicitly accessed. A simple solution would have been to access different sections of the layout via their LAG language equivalents e.g., PM.Menu, which is currently on the left side; PM.ToDo, which is currently on the right side, etc. However, this would confine the author to use the menus only in their current position, and not be able to move, e.g., the menu to the right side of the screen (this being one of the potential areas influenced by cultural backgrounds). For this reason, we have opted for another highly known paradigm, which most programmers should be familiar with, as will be explained in the following.

First, the viewable portion of the interface is divided into North, West, East, South and Centre sections as shown in Fig. 1. This layout is similar to that of the jQuery UI.Layout plugin and the Java BorderLayout. These sections can be sub-divided a further time using the same layout.

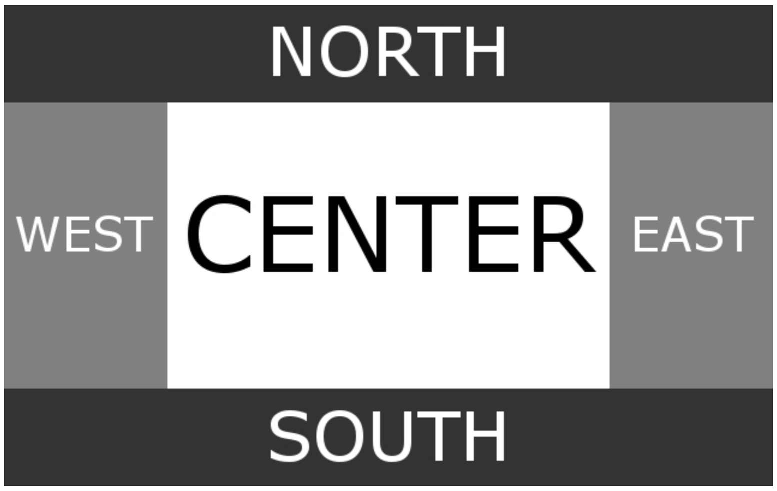

Figure 1: Layout Sections

Each layout section is allocated a section type, denoting the type of content, and this is further used by the delivery engine to properly present the content. The types identified are listed here:

- **Header** This is the course header information.
- **Footer** This type displays the footer information for a course
- **Main** This displays the main content for the page being requested
- **Menu** This displays a navigational tree for the course.
- **Todo** This displays a default to-do list for the course.
- **List** This type displays a list object
- **Text** This displays plain text
- **Image** â“ This displays a picture, the content should be set to a link
- **Progress** The content for this should be a number between 0 and 100 as it is a progress bar.

The content for the header, footer, main, menu and todo types are generated by the delivery system. Although they could be generated manually in an adaptation strategy, these common features can normally be automatically generated, to avoid complicating a strategy with unnecessary detail.

Whereas the type attribute of the layout section determines how this should be formatted, the content attribute determines what should be displayed. The syntax for setting the type and content of a layout section is shown in the following example, which adds a progress bar to the top of the right hand part of a page. The progress bar is set to a user model variable, which would be updated elsewhere in the strategy.

```
Layout[E][N].type = "Progress"
Layout[E][N].content = UM.GM.progress
```

The syntax is similar for the Text and Image types.

The layout for the course is stored on a per user basis for each course, so the layout can be individually adapted, as each user progresses through the course.

Feedback from Authoring Coursework in CS411

A total of 19 unique working strategies were submitted as part of the coursework for the CS411 Dynamic Web Systems course at the University of Warwick. Analysis of the frequency of code constructs in each strategy produced the following table.

Construct/Feature	**2011/12 (19)**	**2010/11 (31)**	**2009/10 (35)**
PM.GM.Concept.access	100.00% (19)	96.77% (30)	100.00% (35)
if	94.74% (18)	100.00% (31)	100.00% (35)
UM.GM variables	89.47% (17)	96.77% (31)	97.14% (34)
Concepts** (plural)	78.95% (15)	N/A	N/A
GM.Concept.label	73.68% (14)	87.10% (27)	91.43% (32)
UM.GM.Concept.accessed*	57.89% (11)	9.68% (3)	N/A
Show/Hide Specific Areas	52.63% (10)	45.16% (14)	40.00% (14)
GM.Concept.weight	47.47% (9)	41.94% (13)	77.14% (27)
enough	21.05% (4)	54.84% (17)	91.43% (32)
Specific Navigational Adaptation**	21.05% (4)	N/A	N/A
Reset the course*	10.53% (2)	0.00% (0)	N/A
Layout Adaptation**	10.53% (2)	N/A	N/A
Type	0.00% (0)	9.68% (3)	14.29% (5)
parent	0.00% (0)	6.45% (2)	2.86% (1)
LIKE*	0.00% (0)	3.23% (1)	N/A
Order	0.00% (0)	6.45% (2)	0.00% (0)
Level	0.00% (0)	0.00% (0)	5.71% (2)

*These constructs/actions are from the new extensions to LAG for the 2010/11 course

**These constructs/actions are from the new extensions to LAG for the 2011/12 course

PM.GM.Concept.access is used to check which concepts are currently being viewed and appeared in every single strategy. It was always used in a for-each loop as a filter during the 2011/12 course.

User Model variables were used in almost all strategies, commonly to store user specific variables, such as user level (beginner, intermediate etc) and also user-concept values such as knowledge and access counts. The amount of variables used dropped as the more powerful selection constructs were available in the 2011/12 course and also accessed was used more often than custom UM counters.

A large majority of the strategies worked by displaying groups of concepts, filtered by their label. This was substantially more than those strategies using the concept weights all the courses, however the label usage has dropped year on year. This is due to more diverse strategies being created, which provides support for the success of the new extensions in LAG. In the 2009/10 course, there were 35 strategies created and the majority were variants of two types of basic adaptation strategy, the roll out strategy (weights) and the beginner, intermediate, advanced strategy (labels). Strategies from the 2010/11 course were more diverse and the 2011/12 course even more so.

Of the latest additions to LAG this year, 10 strategies in the 2011/12 course used code to hide or display complete sections of the user interface and 2 strategies changed the content of the layout to display course completion messages to the users. Also four strategies customised the links in the Todo section, using navigational specific adaptation code to do this.

Selection filters were not used by any strategies in the 2011/12 course, however bulk selection updates were used in 15 strategies, mostly to show or hide concepts in the initalization block.